Open Source Used In AppDynamics_Cloud_Druid_Data_Store_Levitate 22.11.0-518

Cisco Systems, Inc.
www.cisco.com

Cisco has more than 200 offices worldwide.
Addresses, phone numbers, and fax numbers
are listed on the Cisco website at
www.cisco.com/go/offices.

Text Part Number: 78EE117C99-1473551951

This document contains licenses and notices for open source software used in this product. With respect to the free/open source software listed in this document, if you have any questions or wish to receive a copy of any source code to which you may be entitled under the applicable free/open source license(s) (such as the GNU Lesser/General Public License), please submit this form.

In your requests please include the following reference number 78EE117C99-1473551951

Contents

1.1 apiguardian-apiguardian-api 1.1.0

1.1.1 Available under license :

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of

the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works

that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A

PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "{}" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright {yyyy} {name of copyright owner}

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.2 opentelemetry 1.9.1

1.2.1 Available under license :

Note that publicsuffixes.gz is compiled from The Public Suffix List:
https://publicsuffix.org/list/public_suffix_list.dat

It is subject to the terms of the Mozilla Public License, v. 2.0:
https://mozilla.org/MPL/2.0/

1.3 findbugs-jsr305 3.0.2

1.3.1 Available under license :

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

1. "Collective Work" means a work, such as a periodical issue, anthology or encyclopedia, in which the Work in its entirety in unmodified form, along with a number of other contributions, constituting separate and independent works in themselves, are assembled into a collective whole. A work that constitutes a Collective Work will not be considered a Derivative Work (as defined below) for the purposes of this License.
2. "Derivative Work" means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except that a work that constitutes a Collective Work will not be considered a Derivative Work for the purpose of this License. For the avoidance of doubt, where the Work is a musical composition or sound recording, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered a Derivative Work for the purpose of this License.
3. "Licensor" means the individual or entity that offers the Work under the terms of this License.
4. "Original Author" means the individual or entity who created the Work.
5. "Work" means the copyrightable work of authorship offered under the terms of this License.
6. "You" means an individual or entity exercising rights under this License who has not previously violated the

terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

2. Fair Use Rights. Nothing in this license is intended to reduce, limit, or restrict any rights arising from fair use, first sale or other limitations on the exclusive rights of the copyright owner under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

1. to reproduce the Work, to incorporate the Work into one or more Collective Works, and to reproduce the Work as incorporated in the Collective Works;
2. to create and reproduce Derivative Works;
3. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission the Work including as incorporated in Collective Works;
4. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission Derivative Works.
5.

For the avoidance of doubt, where the work is a musical composition:

1. Performance Royalties Under Blanket Licenses. Licensor waives the exclusive right to collect, whether individually or via a performance rights society (e.g. ASCAP, BMI, SESAC), royalties for the public performance or public digital performance (e.g. webcast) of the Work.

2. Mechanical Rights and Statutory Royalties. Licensor waives the exclusive right to collect, whether individually or via a music rights agency or designated agent (e.g. Harry Fox Agency), royalties for any phonorecord You create from the Work ("cover version") and distribute, subject to the compulsory license created by 17 USC Section 115 of the US Copyright Act (or the equivalent in other jurisdictions).

6. Webcasting Rights and Statutory Royalties. For the avoidance of doubt, where the Work is a sound recording, Licensor waives the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions).

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. All rights not expressly granted by Licensor are hereby reserved.

4. Restrictions.The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

1. You may distribute, publicly display, publicly perform, or publicly digitally perform the Work only under the terms of this License, and You must include a copy of, or the Uniform Resource Identifier for, this License with every copy or phonorecord of the Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Work that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Work with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Work as

incorporated in a Collective Work, but this does not require the Collective Work apart from the Work itself to be made subject to the terms of this License. If You create a Collective Work, upon notice from any Licensor You must, to the extent practicable, remove from the Collective Work any credit as required by clause 4(b), as requested. If You create a Derivative Work, upon notice from any Licensor You must, to the extent practicable, remove from the Derivative Work any credit as required by clause 4(b), as requested.

2. If you distribute, publicly display, publicly perform, or publicly digitally perform the Work or any Derivative Works or Collective Works, You must keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or (ii) if the Original Author and/or Licensor designate another party or parties (e.g. a sponsor institute, publishing entity, journal) for attribution in Licensor's copyright notice, terms of service or by other reasonable means, the name of such party or parties; the title of the Work if supplied; to the extent reasonably practicable, the Uniform Resource Identifier, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and in the case of a Derivative Work, a credit identifying the use of the Work in the Derivative Work (e.g., "French translation of the Work by Original Author," or "Screenplay based on original Work by Original Author"). Such credit may be implemented in any reasonable manner; provided, however, that in the case of a Derivative Work or Collective Work, at a minimum such credit will appear where any other comparable authorship credit appears and in a manner at least as prominent as such other comparable authorship credit.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

1. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Derivative Works or Collective Works from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.
2. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

1. Each time You distribute or publicly digitally perform the Work or a Collective Work, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.

2. Each time You distribute or publicly digitally perform a Derivative Work, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.
3. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.
4. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.
5. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

1.4 asm 7.0

1.4.1 Available under license :

<OWNER> = Regents of the University of California
<ORGANIZATION> = University of California, Berkeley
<YEAR> = 1998

In the original BSD license, both occurrences of the phrase "COPYRIGHT HOLDERS AND CONTRIBUTORS" in the disclaimer read "REGENTS AND CONTRIBUTORS".

Here is the license template:

Copyright (c) <YEAR>, <OWNER>

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
Neither the name of the <ORGANIZATION> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS

"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

1.5 profiler 1.0.2

1.5.1 Available under license :

The MIT License (MIT)

Copyright (c) 2013 G4 Code

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.6 commons-logging 1.2

1.6.1 Available under license :

/*
* Licensed to the Apache Software Foundation (ASF) under one or more
* contributor license agreements. See the NOTICE file distributed with
* this work for additional information regarding copyright ownership.
* The ASF licenses this file to You under the Apache License, Version 2.0
* (the "License"); you may not use this file except in compliance with
* the License. You may obtain a copy of the License at
*

* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
Work or Derivative Works thereof in any medium, with or without
modifications, and in Source or Object form, provided that You
meet the following conditions:

(a) You must give any other recipients of the Work or
Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices
stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works
that You distribute, all copyright, patent, trademark, and
attribution notices from the Source form of the Work,
excluding those notices that do not pertain to any part of
the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its
distribution, then any Derivative Works that You distribute must
include a readable copy of the attribution notices contained
within such NOTICE file, excluding those notices that do not
pertain to any part of the Derivative Works, in at least one
of the following places: within a NOTICE text file distributed
as part of the Derivative Works; within the Source form or
documentation, if provided along with the Derivative Works; or,
within a display generated by the Derivative Works, if and
wherever such third-party notices normally appear. The contents
of the NOTICE file are for informational purposes only and
do not modify the License. You may add Your own attribution
notices within Derivative Works that You distribute, alongside
or as an addendum to the NOTICE text from the Work, provided
that such additional attribution notices cannot be construed
as modifying the License.

You may add Your own copyright statement to Your modifications and
may provide additional or different license terms and conditions
for use, reproduction, or distribution of Your modifications, or
for any such Derivative Works as a whole, provided Your use,
reproduction, and distribution of the Work otherwise complies with
the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
any Contribution intentionally submitted for inclusion in the Work
by You to the Licensor shall be under the terms and conditions of
this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify
the terms of any separate license agreement you may have executed

with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate

comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
Apache Commons Logging
Copyright 2003-2014 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

1.7 org.apache.datasketches:datasketches-memory 1.1.0-incubating

1.7.1 Available under license :

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,

"control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or,

within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all

other commercial damages or losses), even if such Contributor
has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
the Work or Derivative Works thereof, You may choose to offer,
and charge a fee for, acceptance of support, warranty, indemnity,
or other liability obligations and/or rights consistent with this
License. However, in accepting such obligations, You may act only
on Your own behalf and on Your sole responsibility, not on behalf
of any other Contributor, and only if You agree to indemnify,
defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following
boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

datasketches-memory
Copyright 2015-2019 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

1.8 opentelemetry-java---io.opentelemetry:opentelemetry-sdk-metrics 1.9.1-alpha

1.8.1 Available under license :

/*
* Copyright The OpenTelemetry Authors
* SPDX-License-Identifier: Apache-2.0
*/

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a

copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object
form, that is based on (or derived from) the Work and for which the
editorial revisions, annotations, elaborations, or other modifications
represent, as a whole, an original work of authorship. For the purposes
of this License, Derivative Works shall not include works that remain
separable from, or merely link (or bind by name) to the interfaces of,
the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity
on behalf of whom a Contribution has been received by Licensor and
subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
copyright license to reproduce, prepare Derivative Works of,
publicly display, publicly perform, sublicense, and distribute the
Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
(except as stated in this section) patent license to make, have made,
use, offer to sell, sell, import, and otherwise transfer the Work,
where such license applies only to those patent claims licensable
by such Contributor that are necessarily infringed by their
Contribution(s) alone or by combination of their Contribution(s)
with the Work to which such Contribution(s) was submitted. If You
institute patent litigation against any entity (including a
cross-claim or counterclaim in a lawsuit) alleging that the Work
or a Contribution incorporated within the Work constitutes direct

or contributory patent infringement, then any patent licenses
granted to You under this License for that Work shall terminate
as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
Work or Derivative Works thereof in any medium, with or without
modifications, and in Source or Object form, provided that You
meet the following conditions:

(a) You must give any other recipients of the Work or
Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices
stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works
that You distribute, all copyright, patent, trademark, and
attribution notices from the Source form of the Work,
excluding those notices that do not pertain to any part of
the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its
distribution, then any Derivative Works that You distribute must
include a readable copy of the attribution notices contained
within such NOTICE file, excluding those notices that do not
pertain to any part of the Derivative Works, in at least one
of the following places: within a NOTICE text file distributed
as part of the Derivative Works; within the Source form or
documentation, if provided along with the Derivative Works; or,
within a display generated by the Derivative Works, if and
wherever such third-party notices normally appear. The contents
of the NOTICE file are for informational purposes only and
do not modify the License. You may add Your own attribution
notices within Derivative Works that You distribute, alongside
or as an addendum to the NOTICE text from the Work, provided
that such additional attribution notices cannot be construed
as modifying the License.

You may add Your own copyright statement to Your modifications and
may provide additional or different license terms and conditions
for use, reproduction, or distribution of Your modifications, or
for any such Derivative Works as a whole, provided Your use,
reproduction, and distribution of the Work otherwise complies with
the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
any Contribution intentionally submitted for inclusion in the Work
by You to the Licensor shall be under the terms and conditions of

this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify
the terms of any separate license agreement you may have executed
with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
names, trademarks, service marks, or product names of the Licensor,
except as required for reasonable and customary use in describing the
origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
agreed to in writing, Licensor provides the Work (and each
Contributor provides its Contributions) on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied, including, without limitation, any warranties or conditions
of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
PARTICULAR PURPOSE. You are solely responsible for determining the
appropriateness of using or redistributing the Work and assume any
risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
whether in tort (including negligence), contract, or otherwise,
unless required by applicable law (such as deliberate and grossly
negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special,
incidental, or consequential damages of any character arising as a
result of this License or out of the use or inability to use the
Work (including but not limited to damages for loss of goodwill,
work stoppage, computer failure or malfunction, or any and all
other commercial damages or losses), even if such Contributor
has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
the Work or Derivative Works thereof, You may choose to offer,
and charge a fee for, acceptance of support, warranty, indemnity,
or other liability obligations and/or rights consistent with this
License. However, in accepting such obligations, You may act only
on Your own behalf and on Your sole responsibility, not on behalf
of any other Contributor, and only if You agree to indemnify,
defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following

boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.9 metrics-integration-with-jmx 4.0.5

1.9.1 Available under license :

No license file was found, but licenses were detected in source scan.

Manifest-Version: 1.0
Bnd-LastModified: 1545938260836
Build-Jdk: 1.8.0_191
Built-By: artem
Bundle-Description: A set of classes which allow you to report metrics
 via JMX.
Bundle-License: http://www.apache.org/licenses/LICENSE-2.0.html
Bundle-ManifestVersion: 2
Bundle-Name: Metrics Integration with JMX
Bundle-SymbolicName: io.dropwizard.metrics.jmx
Bundle-Version: 4.0.5
Created-By: Apache Maven Bundle Plugin
Export-Package: com.codahale.metrics.jmx;uses:="com.codahale.metrics,j
avax.management";version="4.0.5"
Implementation-Title: Metrics Integration with JMX
Implementation-URL: http://metrics.dropwizard.io/metrics-jmx
Implementation-Vendor-Id: io.dropwizard.metrics
Implementation-Version: 4.0.5
Import-Package: org.slf4j;version="[1.6.0,2.0.0)",com.codahale.metrics
;version="[4.0,5)",javax.management

Require-Capability: osgi.ee;filter:="(&(osgi.ee=JavaSE)(version=1.8))"
Tool: Bnd-3.3.0.201609221906

Found in path(s):
* /opt/cola/permits/1274704779_1648835825.49/0/metrics-jmx-4-0-5-jar/META-INF/MANIFEST.MF

1.10 apache-commons-lang 2.6

1.10.1 Available under license :

Apache License

Version 2.0, January 2004

http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate

as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify

the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include

the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
Apache Commons Lang
Copyright 2001-2011 The Apache Software Foundation

This product includes software developed by
The Apache Software Foundation (http://www.apache.org/).

1.11 google-guice 4.2.2

1.11.1 Available under license :

Google Guice - Extensions - MultiBindings
Copyright 2006-2018 Google, Inc.

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by

the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object
form, that is based on (or derived from) the Work and for which the
editorial revisions, annotations, elaborations, or other modifications
represent, as a whole, an original work of authorship. For the purposes
of this License, Derivative Works shall not include works that remain
separable from, or merely link (or bind by name) to the interfaces of,
the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained

within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be

liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.12 guava-internalfuturefailureaccess-and-internalfutures 1.0.1

1.12.1 Available under license :

No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2018 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/

Found in path(s):
* /opt/cola/permits/1130987386_1612872111.26/0/failureaccess-1-0-1-sources-jar/com/google/common/util/concurrent/internal/InternalFutureFailureAccess.java
* /opt/cola/permits/1130987386_1612872111.26/0/failureaccess-1-0-1-sources-jar/com/google/common/util/concurrent/internal/InternalFutures.java

1.13 druid-datasketches 0.20.0

1.13.1 Available under license :

Apache Druid
Copyright 2011-2020 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,

and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object
form, that is based on (or derived from) the Work and for which the
editorial revisions, annotations, elaborations, or other modifications
represent, as a whole, an original work of authorship. For the purposes
of this License, Derivative Works shall not include works that remain
separable from, or merely link (or bind by name) to the interfaces of,
the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the

Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,

whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.14 jackson-bom 2.14.0

1.14.1 Available under license :

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object
form, that is based on (or derived from) the Work and for which the
editorial revisions, annotations, elaborations, or other modifications

represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without

modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade

names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier

identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.15 commons-codec 1.15

1.15.1 Available under license :

Apache Commons Codec
Copyright 2002-2020 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (https://www.apache.org/).

src/test/org/apache/commons/codec/language/DoubleMetaphoneTest.java
contains test data from http://aspell.net/test/orig/batch0.tab.
Copyright (C) 2002 Kevin Atkinson (kevina@gnu.org)

==========

The content of package org.apache.commons.codec.language.bm has been translated
from the original php source code available at http://stevemorse.org/phoneticinfo.htm
with permission from the original authors.
Original source copyright:
Copyright (c) 2008 Alexander Beider & Stephen P. Morse.

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object
form, that is based on (or derived from) the Work and for which the
editorial revisions, annotations, elaborations, or other modifications
represent, as a whole, an original work of authorship. For the purposes
of this License, Derivative Works shall not include works that remain
separable from, or merely link (or bind by name) to the interfaces of,
the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but

excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its

distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise,

unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.16 apache-commons-lang 3.8.1

1.16.1 Available under license :

Apache Commons Lang
Copyright 2001-2018 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work

(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses

granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions.

Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]"

replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.17 asm-tree 7.0

1.17.1 Available under license :

<OWNER> = Regents of the University of California
<ORGANIZATION> = University of California, Berkeley
<YEAR> = 1998

In the original BSD license, both occurrences of the phrase "COPYRIGHT HOLDERS AND CONTRIBUTORS" in the disclaimer read "REGENTS AND CONTRIBUTORS".

Here is the license template:

Copyright (c) <YEAR>, <OWNER>

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
Neither the name of the <ORGANIZATION> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS

"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

1.18 guava 31.0.1-jre

1.18.1 Available under license :

No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2020 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/
/**
* Holder for web specializations of methods of {@code Floats}. Intended to be empty for regular
* version.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-
jar/com/google/common/primitives/FloatsMethodsForWeb.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2007 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0

*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/
/*
 * This following method is a modified version of one found in
 * http://gee.cs.oswego.edu/cgi-bin/viewcvs.cgi/jsr166/src/test/tck/AbstractExecutorServiceTest.java?revision=1.30
 * which contained the following notice:
 *
 * Written by Doug Lea with assistance from members of JCP JSR-166 Expert Group and released to
 * the public domain, as explained at http://creativecommons.org/publicdomain/zero/1.0/
 *
 * Other contributors include Andrew Wright, Jeffrey Hayes, Pat Fisher, Mike Judd.
 */

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/MoreExecutors.java
No license file was found, but licenses were detected in source scan.

/*
* Written by Doug Lea with assistance from members of JCP JSR-166
* Expert Group and released to the public domain, as explained at
* http://creativecommons.org/publicdomain/zero/1.0/
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/cache/LongAdder.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/AtomicDoubleArray.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/Striped64.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/cache/Striped64.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/LongAdder.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2020 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*

* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/
/**
* Holder for web specializations of methods of {@code Doubles}. Intended to be empty for regular
* version.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/primitives/DoublesMethodsForWeb.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2011 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/AtomicLongMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/GwtTransient.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2013 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*

* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/MultimapBuilder.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableMapEntry.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/MoreFiles.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2014 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/eventbus/SubscriberRegistry.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/MoreObjects.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/eventbus/Subscriber.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/ListenerCallQueue.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/eventbus/Dispatcher.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/TrustedListenableFutureTask.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/math/Quantiles.java

No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2007 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/Functions.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/Objects.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/FinalizableSoftReference.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/eventbus/AllowConcurrentEvents.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/AbstractFuture.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/eventbus/DeadEvent.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/package-info.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/ExecutionList.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/MultiInputStream.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/HashBiMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/LittleEndianDataInputStream.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/ListenableFuture.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/Charsets.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/eventbus/EventBus.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/Defaults.java

* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/eventbus/package-info.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/Predicates.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/Closeables.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/eventbus/Subscribe.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/CharStreams.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/eventbus/AsyncEventBus.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/Supplier.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/LineBuffer.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/Preconditions.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/Resources.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/LineReader.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/LittleEndianDataOutputStream.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/Throwables.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/Predicate.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/ByteStreams.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/FinalizableReferenceQueue.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/Files.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/FinalizableWeakReference.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/AbstractIterator.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/primitives/Primitives.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/FinalizableReference.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/CountingOutputStream.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/FinalizablePhantomReference.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/EnumMultiset.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/package-

info.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/CountingInputStream.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/Flushables.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/package-info.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/DirectExecutor.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/Interners.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/Suppliers.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/Function.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2012 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/ByteSource.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/AbstractByteHasher.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/reflect/AbstractInvocationHandler.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/reflect/Parameter.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/ServiceManager.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/StandardSystemProperty.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/math/StatsAccumulator.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-

jar/com/google/common/collect/CartesianList.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/cache/LongAddable.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/reflect/TypeToInstanceMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/html/package-info.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/ChecksumHashFunction.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/ByteSink.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/reflect/TypeCapture.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/math/LinearTransformation.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/FilteredKeyMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/reflect/ClassPath.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/LongAddables.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/xml/package-info.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/BaseEncoding.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/escape/package-info.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/SmoothRateLimiter.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/SipHashFunction.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/reflect/MutableTypeToInstanceMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/CharSource.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/reflect/package-info.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/math/Stats.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableRangeMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/math/PairedStatsAccumulator.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/FileWriteMode.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableRangeSet.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-

jar/com/google/common/io/CharSink.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/LongAddable.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/cache/LongAddables.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/RateLimiter.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/math/PairedStats.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/reflect/Invokable.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/reflect/ImmutableTypeToInstanceMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/Closer.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/ListenableScheduledFuture.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2010 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/primitives/package-info.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/annotations/Beta.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/ListeningExecutorService.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/ThreadFactoryBuilder.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/Ascii.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/Equivalence.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/Strings.java

* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/UncaughtExceptionHandlers.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/Atomics.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ContiguousSet.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/SortedLists.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/net/package-info.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/ForwardingBlockingQueue.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/Monitor.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/annotations/package-info.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2007 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/Multiset.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/Iterables.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/SortedSetMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/MapDifference.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/Sets.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/HashMultiset.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-

jar/com/google/common/collect/TreeMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/AbstractIterator.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/package-info.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/MutableClassToInstanceMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/SetMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/Multisets.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/BiMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ForwardingSortedMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ReverseOrdering.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/AbstractMapEntry.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ByFunctionOrdering.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/Ordering.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableSet.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ForwardingMapEntry.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/EnumHashBiMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/Interner.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ForwardingObject.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/Multimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ForwardingListIterator.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/LinkedListMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ExplicitOrdering.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ForwardingMultiset.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ComparatorOrdering.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableList.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-

jar/com/google/common/collect/AbstractMapBasedMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/AbstractMultiset.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ForwardingSortedSet.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/Lists.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/RegularImmutableSet.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/AbstractBiMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ForwardingQueue.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/EnumBiMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ForwardingMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/LinkedHashMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/Iterators.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/NaturalOrdering.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/Multimaps.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ForwardingMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/NullsFirstOrdering.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/NullsLastOrdering.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/UsingToStringOrdering.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/Maps.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ClassToInstanceMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ListMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ConcurrentHashMultiset.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ArrayListMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ForwardingList.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/TreeMultiset.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-

jar/com/google/common/collect/ForwardingSet.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ForwardingConcurrentMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/AbstractListMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/AbstractMapBasedMultiset.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ForwardingCollection.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/SingletonImmutableSet.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/AbstractSortedSetMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/HashMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/LexicographicalOrdering.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/CompoundOrdering.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/AbstractSetMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ForwardingIterator.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/Synchronized.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ReverseNaturalOrdering.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/LinkedHashMultiset.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2017 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-

jar/com/google/common/primitives/ImmutableIntArray.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/ForwardingCondition.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/ForwardingLock.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/primitives/ImmutableDoubleArray.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/AbstractHashFunction.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/primitives/ImmutableLongArray.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2016 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/LinkedHashMultimapGwtSerializationDependencies.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/DirectedNetworkConnections.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/ElementOrder.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/DirectedMultiNetworkConnections.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/ImmutableValueGraph.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/CollectCollectors.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/MutableValueGraph.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/AbstractGraphBuilder.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/StandardMutableGraph.java

* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableMultisetGwtSerializationDependencies.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ArrayListMultimapGwtSerializationDependencies.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/GraphBuilder.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/StandardMutableValueGraph.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/Comparators.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/EndpointPairIterator.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/ValueGraphBuilder.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/MapRetrievalCache.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/ForwardingGraph.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/AbstractValueGraph.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/ValueGraph.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/MoreCollectors.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/UndirectedMultiNetworkConnections.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/GraphConstants.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/ForwardingValueGraph.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/StandardMutableNetwork.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/StandardNetwork.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/RangeGwtSerializationDependencies.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/StandardValueGraph.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/MapIteratorCache.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/AbstractUndirectedNetworkConnections.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/NetworkConnections.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/HashMultimapGwtSerializationDependencies.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/AbstractGraph.java

* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/MultiEdgesConnecting.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/AbstractNetwork.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/UndirectedNetworkConnections.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/NetworkBuilder.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/ForwardingNetwork.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/EdgesConnecting.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/AbstractDirectedNetworkConnections.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/GraphConnections.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/DirectedGraphConnections.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/UndirectedGraphConnections.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/EndpointPair.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2019 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/Internal.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/primitives/Platform.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2007 The Guava Authors

```
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
/**
  * Returns an array containing all of the elements in the specified collection. This method
  * returns the elements in the order they are returned by the collection's iterator. The returned
  * array is "safe" in that no references to it are maintained by the collection. The caller is
  * thus free to modify the returned array.
  *
  * <p>This method assumes that the collection size doesn't change while the method is running.
  *
  * <p>TODO(kevinb): support concurrently modified collections?
  *
  * @param c the collection for which to return an array of elements
  */
```

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ObjectArrays.java
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2012 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/
/*
  * This method was rewritten in Java from an intermediate step of the Murmur hash function in
  * http://code.google.com/p/smhasher/source/browse/trunk/MurmurHash3.cpp, which contained the
```

```
 * following header:
 *
 * MurmurHash3 was written by Austin Appleby, and is placed in the public domain. The author
 * hereby disclaims copyright to this source code.
 */
```

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/SmallCharMatcher.java
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2015 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you
* may not use this file except in compliance with the License.  You may
* obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
* implied.  See the License for the specific language governing
* permissions and limitations under the License.
*/
```

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/Streams.java
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2012 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
```

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/CompactHashSet.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/FilteredMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/DescendingImmutableSortedSet.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/AbstractNavigableMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/FilteredSetMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/FilteredKeySetMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ForwardingImmutableList.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ForwardingImmutableSet.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/CompactLinkedHashSet.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/UnmodifiableSortedMultiset.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/EvictingQueue.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/CompactHashMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/TreeTraverser.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/DescendingMultiset.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableEnumMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/FilteredEntryMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/RegularImmutableAsList.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ForwardingNavigableMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/TransformedListIterator.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/FilteredKeyListMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/TreeRangeMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/CompactLinkedHashMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ForwardingBlockingDeque.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-

jar/com/google/common/collect/AbstractMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/FilteredEntrySetMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/RangeMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ForwardingDeque.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/AbstractSortedKeySortedSetMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/AllEqualOrdering.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ForwardingNavigableSet.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/ForwardingBlockingDeque.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/TransformedIterator.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/SortedMultisetBridge.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ForwardingImmutableMap.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2013 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/reflect/TypeVisitor.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/eventbus/SubscriberExceptionHandler.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/AbstractTable.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/VerifyException.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-

jar/com/google/common/base/Verify.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/thirdparty/publicsuffix/PublicSuffixType.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/FilteredMultimapValues.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/eventbus/SubscriberExceptionContext.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/CharSequenceReader.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/WrappingScheduledExecutorService.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/HashingInputStream.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/Utf8.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/Runnables.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2011 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the
* License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
* express or implied. See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/GeneralRange.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/RegularImmutableSortedMultiset.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableSortedMultisetFauxverideShim.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/AbstractRangeSet.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/SortedIterable.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/Count.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-

jar/com/google/common/collect/ImmutableSortedMultiset.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-
jar/com/google/common/collect/SortedIterables.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-
jar/com/google/common/collect/ForwardingSortedMultiset.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-
jar/com/google/common/collect/RangeSet.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2015 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-
jar/com/google/common/util/concurrent/Platform.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-
jar/com/google/common/io/ReaderInputStream.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-
jar/com/google/common/util/concurrent/InterruptibleTask.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-
jar/com/google/common/util/concurrent/AsyncCallable.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-
jar/com/google/common/hash/LittleEndianByteArray.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-
jar/com/google/common/collect/ConsumingQueueIterator.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-
jar/com/google/common/hash/MacHashFunction.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-
jar/com/google/common/util/concurrent/CombinedFuture.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-
jar/com/google/common/util/concurrent/AggregateFutureState.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-
jar/com/google/common/hash/FarmHashFingerprint64.java
No license file was found, but licenses were detected in source scan.

/*

* Copyright (C) 2018 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-
jar/com/google/common/hash/ImmutableSupplier.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-
jar/com/google/common/collect/JdkBackedImmutableMultiset.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-
jar/com/google/common/util/concurrent/ExecutionSequencer.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-
jar/com/google/common/collect/JdkBackedImmutableSet.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2011 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-
jar/com/google/common/hash/AbstractStreamingHasher.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-
jar/com/google/common/cache/Cache.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-
jar/com/google/common/cache/AbstractLoadingCache.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-

jar/com/google/common/cache/CacheBuilderSpec.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/BloomFilter.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/primitives/UnsignedLong.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/cache/ForwardingLoadingCache.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/DescendingImmutableSortedMultiset.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/net/HostAndPort.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/PrimitiveSink.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/math/IntMath.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/Hasher.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/PairwiseEquivalence.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/cache/LoadingCache.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/primitives/ParseRequest.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/Hashing.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/math/DoubleUtils.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/cache/Weigher.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/primitives/UnsignedInteger.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/AbstractScheduledService.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/AbstractListeningExecutorService.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/primitives/UnsignedInts.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/ExecutionError.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/Funnel.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/reflect/TypeParameter.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/BloomFilterStrategies.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/CycleDetectingLockFactory.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-

jar/com/google/common/net/MediaType.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/Uninterruptibles.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/net/HttpHeaders.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/cache/RemovalCause.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/cache/package-info.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/math/MathPreconditions.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/Ticker.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/Absent.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/ForwardingExecutorService.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/EmptyContiguousSet.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/math/DoubleMath.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/cache/RemovalListener.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/AbstractSortedMultiset.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/Enums.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/AbstractCompositeHashFunction.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/Present.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/Funnels.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/primitives/UnsignedLongs.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/HashingOutputStream.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/Optional.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/math/BigIntegerMath.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/AbstractHasher.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/BoundType.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/TreeRangeSet.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-

jar/com/google/common/cache/RemovalListeners.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/cache/ForwardingCache.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/FutureCallback.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/HashCode.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/math/package-info.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/Murmur3_128HashFunction.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/math/LongMath.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/AbstractNonStreamingHashFunction.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/MessageDigestHashFunction.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/Murmur3_32HashFunction.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/Queues.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/cache/AbstractCache.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/ForwardingListeningExecutorService.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/RegularImmutableMultiset.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/Crc32cHashFunction.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/WrappingExecutorService.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/UncheckedExecutionException.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/HashFunction.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/cache/CacheStats.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/cache/RemovalNotification.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/AsyncFunction.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/RegularContiguousSet.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/ListeningScheduledExecutorService.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/FunctionalEquivalence.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-

jar/com/google/common/reflect/Types.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/cache/CacheLoader.java
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2011 The Guava Authors.
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/
```

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/package-info.java
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2014 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
```

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/TopKSelector.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/Graphs.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/PredecessorsFunction.java

* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/ImmutableNetwork.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/Graph.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/MutableGraph.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/Network.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/MutableNetwork.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/SuccessorsFunction.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/InsecureRecursiveDeleteException.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/ImmutableGraph.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/RecursiveDeleteOption.java
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2009 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
/**
  * Not supported. <b>You are attempting to create a map that may contain a non-{@code Comparable}
  * key.</b> Proper calls will resolve to the version in {@code ImmutableSortedMap}, not this dummy
  * version.
  *
  * @throws UnsupportedOperationException always
  * @deprecated <b>Pass a key of type {@code Comparable} to use {@link
  *     ImmutableSortedMap#of(Comparable, Object)}.</b>
  */
```

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableSortedMapFauxverideShim.java

No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2016 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/
```

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/JdkPattern.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/CommonPattern.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/CommonMatcher.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/PatternCompiler.java

No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2020 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/
/**
* Holder for web specializations of methods of {@code Ints}. Intended to be empty for regular
* version.
*/
```

Found in path(s):

* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/primitives/IntsMethodsForWeb.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2006 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/FuturesGetChecked.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/UncheckedTimeoutException.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/TimeLimiter.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/FluentFuture.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/ImmediateFuture.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/escape/CharEscaper.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/reflect/TypeToken.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/PatternFilenameFilter.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/annotations/VisibleForTesting.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/SimpleTimeLimiter.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/FakeTimeLimiter.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/AppendableWriter.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/Futures.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/AbstractCatchingFuture.java

* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/TimeoutFuture.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/GwtFuturesCatchingSpecialization.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/escape/CharEscaperBuilder.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/AbstractTransformFuture.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/CollectionFuture.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/CaseFormat.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/AggregateFuture.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/GwtFluentFutureCatchingSpecialization.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2009 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the
* License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
* express or implied. See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableSortedAsList.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2005 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either

express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/reflect/Reflection.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2009 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/ForwardingListenableFuture.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/Service.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/cache/CacheBuilder.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/ForwardingFuture.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/MapMakerInternalMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/annotations/GwtCompatible.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/annotations/GwtIncompatible.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/cache/LocalCache.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/AbstractExecutionThreadService.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/AbstractIdleService.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/cache/ReferenceEntry.java

* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/net/InternetDomainName.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/html/HtmlEscapers.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/AbstractService.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/Cut.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/Platform.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/ForwardingFluentFuture.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/escape/Escapers.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/primitives/SignedBytes.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/primitives/UnsignedBytes.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/net/HostSpecifier.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/escape/ArrayBasedCharEscaper.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/Splitter.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/escape/Platform.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/MapMaker.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/DenseImmutableTable.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/net/UrlEscapers.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/ByteArrayDataOutput.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/RegularImmutableTable.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/Callables.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/escape/ArrayBasedEscaperMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/reflect/TypeResolver.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/ByteArrayDataInput.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/escape/ArrayBasedUnicodeEscaper.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/SparseImmutableTable.java

* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/xml/XmlEscapers.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/LineProcessor.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/JdkFutureAdapters.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/SettableFuture.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/ByteProcessor.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2015 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/package-info.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableBiMapFauxverideShim.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/CollectSpliterators.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2018 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,

* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/JdkBackedImmutableBiMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/IndexedImmutableSet.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/BaseImmutableMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/JdkBackedImmutableMap.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2010 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ForwardingSetMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/MinMaxPriorityQueue.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ForwardingImmutableCollection.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/AbstractSequentialIterator.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ForwardingListMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/UnmodifiableListIterator.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ForwardingSortedSetMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-

jar/com/google/common/collect/RowSortedTable.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/SortedMapDifference.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2009 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableClassToInstanceMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/DiscreteDomain.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableSetMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/RegularImmutableSortedSet.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableSortedSetFauxverideShim.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ForwardingTable.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableEnumSet.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableAsList.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/AbstractIndexedListIterator.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/EmptyImmutableSetMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/SingletonImmutableTable.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/TableCollectors.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/RegularImmutableList.java

* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ComparisonChain.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableSortedMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableTable.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ArrayTable.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ComputationException.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/SingletonImmutableList.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2021 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/escape/ParametricNullness.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/ElementTypesAreNonnullByDefault.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/ParametricNullness.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/net/ParametricNullness.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/xml/ParametricNullness.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/ElementTypesAreNonnullByDefault.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/ParametricNullness.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/escape/ElementTypesAreNonnullByDefault.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-

jar/com/google/common/eventbus/ElementTypesAreNonnullByDefault.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/reflect/ParametricNullness.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/primitives/ParametricNullness.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/ElementTypesAreNonnullByDefault.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/primitives/ElementTypesAreNonnullByDefault.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/xml/ElementTypesAreNonnullByDefault.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/ParametricNullness.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/reflect/ElementTypesAreNonnullByDefault.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ElementTypesAreNonnullByDefault.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/eventbus/ParametricNullness.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/math/ParametricNullness.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/net/ElementTypesAreNonnullByDefault.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/cache/ElementTypesAreNonnullByDefault.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/math/ElementTypesAreNonnullByDefault.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/ElementTypesAreNonnullByDefault.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/cache/ParametricNullness.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/html/ElementTypesAreNonnullByDefault.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ParametricNullness.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/ParametricNullness.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/ParametricNullness.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/ElementTypesAreNonnullByDefault.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/html/ParametricNullness.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2020 The Guava Authors
*

```
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/
/**
* Holder for web specializations of methods of {@code Shorts}. Intended to be empty for regular
* version.
*/
```

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/primitives/ShortsMethodsForWeb.java
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2011 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/
/*
 * This method was written by Doug Lea with assistance from members of JCP JSR-166 Expert Group
 * and released to the public domain, as explained at
 * http://creativecommons.org/licenses/publicdomain
 *
 * As of 2010/06/11, this method is identical to the (package private) hash method in OpenJDK 7's
 * java.util.HashMap class.
 */
```

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/Striped.java

No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2017 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/AbstractBaseGraph.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/ClosingFuture.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/BaseGraph.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/Traverser.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2016 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/
/**
* Holder for extra methods of {@code Objects} only in web. Intended to be empty for regular
* version.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/ExtraObjectsMethodsForWeb.java
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2020 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/
```

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/math/BigDecimalMath.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/OverflowAvoidingLockSupport.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/hash/Java8Compatibility.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/math/ToDoubleRounder.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/Java8Compatibility.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/Java8Compatibility.java
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2009 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
```

```
*/
/**
* Outer class that exists solely to let us write {@code Partially.GwtIncompatible} instead of plain
* {@code GwtIncompatible}. This is more accurate for {@link Futures#catching}, which is available
* under GWT but with a slightly different signature.
*
* <p>We can't use {@code PartiallyGwtIncompatible} because then the GWT compiler wouldn't recognize
* it as a {@code GwtIncompatible} annotation. And for {@code Futures.catching}, we need the GWT
* compiler to autostrip the normal server method in order to expose the special, inherited GWT
* version.
*/
```

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/Partially.java
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2020 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
```

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/ServiceManagerBridge.java
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2021 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
```

express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/NullnessCasts.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/NullnessCasts.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/NullnessCasts.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2008 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/Platform.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableMapEntrySet.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableMapValues.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/Tables.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/Collections2.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableMultiset.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/HashBasedTable.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-

jar/com/google/common/collect/ImmutableMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/CollectPreconditions.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/thirdparty/publicsuffix/PublicSuffixPatterns.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableMapKeySet.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/StandardRowSortedTable.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableListMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/RegularImmutableMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/PeekingIterator.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/Table.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableEntry.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/RegularImmutableBiMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/SingletonImmutableBiMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/Serialization.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableSortedSet.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableBiMap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/StandardTable.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/EmptyImmutableListMultimap.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/ImmutableCollection.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/Range.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/TreeBasedTable.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/UnmodifiableIterator.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2008 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at

*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/internal/Finalizer.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/primitives/Booleans.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/thirdparty/publicsuffix/TrieParser.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/primitives/Ints.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/ListenableFutureTask.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/primitives/Chars.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/primitives/Shorts.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/escape/Escaper.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/primitives/Floats.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/Stopwatch.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/FileBackedOutputStream.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/CharMatcher.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/escape/UnicodeEscaper.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/FluentIterable.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/io/MultiReader.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/Converter.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/primitives/Doubles.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/net/InetAddresses.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-

jar/com/google/common/primitives/Longs.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/primitives/Bytes.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/base/Joiner.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/util/concurrent/SequentialExecutor.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/net/PercentEscaper.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2011 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not
* use this file except in compliance with the License. You may obtain a copy of
* the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
* WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
* License for the specific language governing permissions and limitations under
* the License.
*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/SortedMultiset.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/SortedMultisets.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2019 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.

*/

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/graph/IncidentEdgeSet.java
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/CompactHashing.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2008 The Guava Authors
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
/*
 * This method was rewritten in Java from an intermediate step of the Murmur hash function in
 * http://code.google.com/p/smhasher/source/browse/trunk/MurmurHash3.cpp, which contained the
 * following header:
 *
 * MurmurHash3 was written by Austin Appleby, and is placed in the public domain. The author
 * hereby disclaims copyright to this source code.
 */

Found in path(s):
* /opt/cola/permits/1288520115_1647351767.93/0/guava-31-0-1-jre-sources-jar/com/google/common/collect/Hashing.java

1.19 guava-listenablefuture-only 9999.0-empty-to-avoid-conflict-with-guava

1.19.1 Available under license :

Found license 'GNU Lesser General Public License' in '// This library is free software; you can redistribute it and/or // modify it under the terms of the GNU Lesser General Public // License as published by the Free Software Foundation; either // version 2.1 of the License, or (at your option) any later version. // This library is distributed in the hope that it will be useful, // but WITHOUT ANY WARRANTY; without even the implied warranty of // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU // Lesser General Public

License for more details. // You should have received a copy of the GNU Lesser General Public * This grammar is in the PUBLIC DOMAIN'

GNU LESSER GENERAL PUBLIC LICENSE
Version 2.1, February 1999

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free

software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not

covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If

identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding

machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library

facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by

all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our

decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the library's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either

version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public
License along with this library; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the library, if
necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the
library `Frob' (a library for tweaking knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990
Ty Coon, President of Vice

That's all there is to it!
//
// checkstyle: Checks Java source code for adherence to a set of rules.
// Copyright (C) 2001-2020 the original author or authors.
//
// This library is free software; you can redistribute it and/or
// modify it under the terms of the GNU Lesser General Public
// License as published by the Free Software Foundation; either
// version 2.1 of the License, or (at your option) any later version.
//
// This library is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
// Lesser General Public License for more details.
//
// You should have received a copy of the GNU Lesser General Public
// License along with this library; if not, write to the Free Software
// Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
//

1.20 apache-httpcomponents-core 4.4.6

1.20.1 Available under license :

Apache HttpCore NIO
Copyright 2005-2017 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object

form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
Work or Derivative Works thereof in any medium, with or without
modifications, and in Source or Object form, provided that You
meet the following conditions:

(a) You must give any other recipients of the Work or
Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices
stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works
that You distribute, all copyright, patent, trademark, and
attribution notices from the Source form of the Work,
excluding those notices that do not pertain to any part of
the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its
distribution, then any Derivative Works that You distribute must
include a readable copy of the attribution notices contained
within such NOTICE file, excluding those notices that do not
pertain to any part of the Derivative Works, in at least one
of the following places: within a NOTICE text file distributed
as part of the Derivative Works; within the Source form or
documentation, if provided along with the Derivative Works; or,
within a display generated by the Derivative Works, if and
wherever such third-party notices normally appear. The contents
of the NOTICE file are for informational purposes only and
do not modify the License. You may add Your own attribution
notices within Derivative Works that You distribute, alongside
or as an addendum to the NOTICE text from the Work, provided
that such additional attribution notices cannot be construed
as modifying the License.

You may add Your own copyright statement to Your modifications and
may provide additional or different license terms and conditions
for use, reproduction, or distribution of Your modifications, or
for any such Derivative Works as a whole, provided Your use,
reproduction, and distribution of the Work otherwise complies with
the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
any Contribution intentionally submitted for inclusion in the Work
by You to the Licensor shall be under the terms and conditions of
this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify
the terms of any separate license agreement you may have executed
with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

   To apply the Apache License to your work, attach the following
   boilerplate notice, with the fields enclosed by brackets "[]"
   replaced with your own identifying information. (Don't include
   the brackets!) The text should be enclosed in the appropriate
   comment syntax for the file format. We also recommend that a

file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.21 snappy-java 1.1.7.3

1.21.1 Available under license :

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

You must give any other recipients of the Work or Derivative Works a copy of this License; and

You must cause any modified files to carry prominent notices stating that You changed the files; and

You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.
You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.
5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.22 checker-qual 2.10.0

1.22.1 Available under license :

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.23 netty-tomcatnative-openssl-dynamic 2.0.48.Final

1.23.1 Available under license :

No license file was found, but licenses were detected in source scan.

<!--
~ Copyright 2016 The Netty Project
~
~ The Netty Project licenses this file to you under the Apache License,
~ version 2.0 (the "License"); you may not use this file except in compliance
~ with the License. You may obtain a copy of the License at:
~
~   http://www.apache.org/licenses/LICENSE-2.0
~
~ Unless required by applicable law or agreed to in writing, software
~ distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
~ WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
~ License for the specific language governing permissions and limitations
~ under the License.
-->

Found in path(s):
* /opt/cola/permits/1473561893_1669090143.9916968/0/netty-tcnative-2-0-48-final-jar/META-INF/maven/io.netty/netty-tcnative/pom.xml
No license file was found, but licenses were detected in source scan.

Manifest-Version: 1.0
Implementation-Title: Netty/TomcatNative [OpenSSL - Dynamic]
Bundle-Description: A Mavenized fork of Tomcat Native which incorporat
es various patches. This artifact is dynamically linked to OpenSSL
 and Apache APR.
Automatic-Module-Name: io.netty.internal.tcnative
Bundle-License: http://www.apache.org/licenses/LICENSE-2.0.txt
Bundle-SymbolicName: io.netty.tcnative
Implementation-Version: 2.0.48.Final

Built-By: norman
Bnd-LastModified: 1643124869520
Bundle-ManifestVersion: 2
Implementation-Vendor-Id: io.netty
Fragment-Host: io.netty.tcnative-classes
Tool: Bnd-5.1.1.202006162103
Bundle-Name: Netty/TomcatNative [OpenSSL - Dynamic]
Bundle-Version: 2.0.48.Final
Build-Jdk-Spec: 1.8
Created-By: Apache Maven Bundle Plugin
Build-Jdk: 1.8.0_252
Implementation-URL: https://github.com/netty/netty-tcnative/netty-tcna
tive/

Found in path(s):
* /opt/cola/permits/1473561893_1669090143.9916968/0/netty-tcnative-2-0-48-final-jar/META-INF/MANIFEST.MF

1.24 junit-platform-junit-platform-commons 1.7.2

1.24.1 Available under license :

import java.io.File
import java.net.URI

data class License(val name: String, val url: URI, val headerFile: File)
Apache License
==========

Version 2.0, January 2004
<<https://www.apache.org/licenses/>>

Terms and Conditions for use, reproduction, and distribution

1. Definitions

License shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

Licensor shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

Legal Entity shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, control means **(i)** the power, direct or indirect, to cause the direction or management of such entity, whether by

contract or otherwise, or **(ii)** ownership of fifty percent (50%) or more of the outstanding shares, or **(iii)** beneficial ownership of such entity.

You (or Your) shall mean an individual or Legal Entity exercising permissions granted by this License.

Source form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

Object form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

Work shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

Derivative Works shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

Contribution shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, submitted means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as Not a Contribution.

Contributor shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such

Derivative Works in Source or Object form.

3. Grant of Patent License

Subject to the terms and conditions of this License, each Contributor hereby
grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free,
irrevocable (except as stated in this section) patent license to make, have
made, use, offer to sell, sell, import, and otherwise transfer the Work, where
such license applies only to those patent claims licensable by such Contributor
that are necessarily infringed by their Contribution(s) alone or by combination
of their Contribution(s) with the Work to which such Contribution(s) was
submitted. If You institute patent litigation against any entity (including a
cross-claim or counterclaim in a lawsuit) alleging that the Work or a
Contribution incorporated within the Work constitutes direct or contributory
patent infringement, then any patent licenses granted to You under this License
for that Work shall terminate as of the date such litigation is filed.

4. Redistribution

You may reproduce and distribute copies of the Work or Derivative Works thereof
in any medium, with or without modifications, and in Source or Object form,
provided that You meet the following conditions:

* **(a)** You must give any other recipients of the Work or Derivative Works a copy of
this License; and
* **(b)** You must cause any modified files to carry prominent notices stating that You
changed the files; and
* **(c)** You must retain, in the Source form of any Derivative Works that You distribute,
all copyright, patent, trademark, and attribution notices from the Source form
of the Work, excluding those notices that do not pertain to any part of the
Derivative Works; and
* **(d)** If the Work includes a NOTICE text file as part of its distribution, then any
Derivative Works that You distribute must include a readable copy of the
attribution notices contained within such NOTICE file, excluding those notices
that do not pertain to any part of the Derivative Works, in at least one of the
following places: within a NOTICE text file distributed as part of the
Derivative Works; within the Source form or documentation, if provided along
with the Derivative Works; or, within a display generated by the Derivative
Works, if and wherever such third-party notices normally appear. The contents of
the NOTICE file are for informational purposes only and do not modify the
License. You may add Your own attribution notices within Derivative Works that
You distribute, alongside or as an addendum to the NOTICE text from the Work,
provided that such additional attribution notices cannot be construed as
modifying the License.

You may add Your own copyright statement to Your modifications and may provide
additional or different license terms and conditions for use, reproduction, or
distribution of Your modifications, or for any such Derivative Works as a whole,

provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an AS IS BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your

sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.
Apache License
==========

Version 2.0, January 2004
<<https://www.apache.org/licenses/>>

Terms and Conditions for use, reproduction, and distribution

1. Definitions

License shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

Licensor shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

Legal Entity shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, control means **(i)** the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or **(ii)** ownership of fifty percent (50%) or more of the outstanding shares, or **(iii)** beneficial ownership of such entity.

You (or Your) shall mean an individual or Legal Entity exercising permissions granted by this License.

Source form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

Object form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

Work shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

Derivative Works shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

Contribution shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, submitted means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as Not a Contribution.

Contributor shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

* **(a)** You must give any other recipients of the Work or Derivative Works a copy of

this License; and
* **(b)** You must cause any modified files to carry prominent notices stating that You changed the files; and
* **(c)** You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
* **(d)** If the Work includes a NOTICE text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an AS IS BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied,

including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets `[]` replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same printed page as the copyright notice for easier identification within third-party archives.

  Copyright [yyyy] [name of copyright owner]

  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    https://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

Eclipse Public License - v 2.0
==========

THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS ECLIPSE PUBLIC LICENSE (AGREEMENT). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.

1. Definitions

Contribution means:
* **a)** in the case of the initial Contributor, the initial content Distributed under this Agreement, and
* **b)** in the case of each subsequent Contributor:
 * **i)** changes to the Program, and
 * **ii)** additions to the Program;
where such changes and/or additions to the Program originate from and are Distributed by that particular Contributor. A Contribution originates from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor's behalf. Contributions do not include changes or additions to the Program that are not Modified Works.

Contributor means any person or entity that Distributes the Program.

Licensed Patents mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program.

Program means the Contributions Distributed in accordance with this Agreement.

Recipient means anyone who receives the Program under this Agreement or any Secondary License (as applicable), including Contributors.

Derivative Works shall mean any work, whether in Source Code or other form, that is based on (or derived from) the Program and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship.

Modified Works shall mean any work in Source Code or other form that results from an addition to, deletion from, or modification of the contents of the Program, including, for purposes of clarity any new file in Source Code form that contains any contents of the Program. Modified Works shall not include works that contain only declarations, interfaces, types, classes, structures, or files of the Program solely in each case in order to link to, bind by name, or subclass the Program or Modified Works thereof.

Distribute means the acts of **a)** distributing or **b)** making available in any manner that enables the transfer of a copy.

Source Code means the form of a Program preferred for making modifications, including but not limited to software source code, documentation source, and configuration files.

Secondary License means either the GNU General Public License, Version 2.0, or any later versions of that license, including any exceptions or additional permissions as identified by the initial Contributor.

2. Grant of Rights

a) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, Distribute and sublicense the Contribution of such Contributor, if any, and such Derivative Works.

b) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in Source Code or other form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder.

c) Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to Distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.

d) Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement.

e) Notwithstanding the terms of any Secondary License, no Contributor makes additional grants to any Recipient (other than those set forth in this Agreement) as a result of such Recipient's receipt of the Program under the terms of a Secondary License (if permitted under the terms of Section 3).

3. Requirements

3.1 If a Contributor Distributes the Program in any form, then:

* **a)** the Program must also be made available as Source Code, in accordance with section 3.2, and the Contributor must accompany the Program with a statement that the Source Code for the Program is available under this Agreement, and informs Recipients how to obtain it in a reasonable manner on or through a medium customarily used for software exchange; and

* **b)** the Contributor may Distribute the Program under a license different than this Agreement, provided that such license:
 * **i)** effectively disclaims on behalf of all other Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose;

* **ii)** effectively excludes on behalf of all other Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits;
* **iii)** does not attempt to limit or alter the recipients' rights in the Source Code under section 3.2; and
* **iv)** requires any subsequent distribution of the Program by any party to be under a license that satisfies the requirements of this section 3.

3.2 When the Program is Distributed as Source Code:

* **a)** it must be made available under this Agreement, or if the Program **(i)** is combined with other material in a separate file or files made available under a Secondary License, and **(ii)** the initial Contributor attached to the Source Code the notice described in Exhibit A of this Agreement, then the Program may be made available under the terms of such Secondary Licenses, and
* **b)** a copy of this Agreement must be included with each copy of the Program.

3.3 Contributors may not remove or alter any copyright, patent, trademark, attribution notices, disclaimers of warranty, or limitations of liability (notices) contained within the Program from any copy of the Program which they Distribute, provided that Contributors may add their own appropriate notices.

4. Commercial Distribution

Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor (Commercial Contributor) hereby agrees to defend and indemnify every other Contributor (Indemnified Contributor) against any losses, damages and costs (collectively Losses) arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must: **a)** promptly notify the Commercial Contributor in writing of such claim, and **b)** allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those damages.

5. No Warranty

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, AND TO THE EXTENT PERMITTED BY APPLICABLE LAW, THE PROGRAM IS PROVIDED ON AN AS IS BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the

appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

6. Disclaimer of Liability

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, AND TO THE EXTENT PERMITTED BY APPLICABLE LAW, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. General

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. The Eclipse Foundation is the initial Agreement Steward. The Eclipse Foundation may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be Distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to Distribute the Program (including its Contributions) under the new version.

Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved. Nothing in this Agreement is intended to be enforceable by any entity that is not a Contributor or Recipient. No third-party beneficiary rights are created under this Agreement.

Exhibit A - Form of Secondary Licenses Notice

> This Source Code may also be made available under the following Secondary Licenses when the conditions for such availability set forth in the Eclipse Public License, v. 2.0 are satisfied: {name license(s), version(s), and exceptions or additional permissions here}.

Simply including a copy of this Agreement, including this Exhibit A is not sufficient to license the Source Code under Secondary Licenses.

If it is not possible or desirable to put the notice in a particular file, then You may include the notice in a location (such as a LICENSE file in a relevant directory) where a recipient would be likely to look for such a notice.

You may add additional accurate notices of copyright ownership.
Open Source Licenses
==========

This product may include a number of subcomponents with separate
copyright notices and license terms. Your use of the source code for
these subcomponents is subject to the terms and conditions of the
subcomponent's license, as noted in the LICENSE-<subcomponent>.md
files.
[[contributors]]
== Contributors

Browse the {junit5-repo}/graphs/contributors[current list of contributors] directly on GitHub.

1.25 junit-jupiter-junit-jupiter-engine 5.7.2

1.25.1 Available under license :

import java.io.File
import java.net.URI

data class License(val name: String, val url: URI, val headerFile: File)
Apache License
==========

Version 2.0, January 2004
<<https://www.apache.org/licenses/>>

Terms and Conditions for use, reproduction, and distribution

1. Definitions

License shall mean the terms and conditions for use, reproduction, and
distribution as defined by Sections 1 through 9 of this document.

Licensor shall mean the copyright owner or entity authorized by the copyright

owner that is granting the License.

Legal Entity shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, control means **(i)** the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or **(ii)** ownership of fifty percent (50%) or more of the outstanding shares, or **(iii)** beneficial ownership of such entity.

You (or Your) shall mean an individual or Legal Entity exercising permissions granted by this License.

Source form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

Object form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

Work shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

Derivative Works shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

Contribution shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, submitted means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as Not a Contribution.

Contributor shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License

Subject to the terms and conditions of this License, each Contributor hereby
grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free,
irrevocable copyright license to reproduce, prepare Derivative Works of,
publicly display, publicly perform, sublicense, and distribute the Work and such
Derivative Works in Source or Object form.

3. Grant of Patent License

Subject to the terms and conditions of this License, each Contributor hereby
grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free,
irrevocable (except as stated in this section) patent license to make, have
made, use, offer to sell, sell, import, and otherwise transfer the Work, where
such license applies only to those patent claims licensable by such Contributor
that are necessarily infringed by their Contribution(s) alone or by combination
of their Contribution(s) with the Work to which such Contribution(s) was
submitted. If You institute patent litigation against any entity (including a
cross-claim or counterclaim in a lawsuit) alleging that the Work or a
Contribution incorporated within the Work constitutes direct or contributory
patent infringement, then any patent licenses granted to You under this License
for that Work shall terminate as of the date such litigation is filed.

4. Redistribution

You may reproduce and distribute copies of the Work or Derivative Works thereof
in any medium, with or without modifications, and in Source or Object form,
provided that You meet the following conditions:

* **(a)** You must give any other recipients of the Work or Derivative Works a copy of
this License; and
* **(b)** You must cause any modified files to carry prominent notices stating that You
changed the files; and
* **(c)** You must retain, in the Source form of any Derivative Works that You distribute,
all copyright, patent, trademark, and attribution notices from the Source form
of the Work, excluding those notices that do not pertain to any part of the
Derivative Works; and
* **(d)** If the Work includes a NOTICE text file as part of its distribution, then any
Derivative Works that You distribute must include a readable copy of the
attribution notices contained within such NOTICE file, excluding those notices
that do not pertain to any part of the Derivative Works, in at least one of the
following places: within a NOTICE text file distributed as part of the
Derivative Works; within the Source form or documentation, if provided along
with the Derivative Works; or, within a display generated by the Derivative
Works, if and wherever such third-party notices normally appear. The contents of
the NOTICE file are for informational purposes only and do not modify the
License. You may add Your own attribution notices within Derivative Works that
You distribute, alongside or as an addendum to the NOTICE text from the Work,

provided that such additional attribution notices cannot be construed as
modifying the License.

You may add Your own copyright statement to Your modifications and may provide
additional or different license terms and conditions for use, reproduction, or
distribution of Your modifications, or for any such Derivative Works as a whole,
provided Your use, reproduction, and distribution of the Work otherwise complies
with the conditions stated in this License.

5. Submission of Contributions

Unless You explicitly state otherwise, any Contribution intentionally submitted
for inclusion in the Work by You to the Licensor shall be under the terms and
conditions of this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify the terms of
any separate license agreement you may have executed with Licensor regarding
such Contributions.

6. Trademarks

This License does not grant permission to use the trade names, trademarks,
service marks, or product names of the Licensor, except as required for
reasonable and customary use in describing the origin of the Work and
reproducing the content of the NOTICE file.

7. Disclaimer of Warranty

Unless required by applicable law or agreed to in writing, Licensor provides the
Work (and each Contributor provides its Contributions) on an AS IS BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied,
including, without limitation, any warranties or conditions of TITLE,
NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are
solely responsible for determining the appropriateness of using or
redistributing the Work and assume any risks associated with Your exercise of
permissions under this License.

8. Limitation of Liability

In no event and under no legal theory, whether in tort (including negligence),
contract, or otherwise, unless required by applicable law (such as deliberate
and grossly negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special, incidental,
or consequential damages of any character arising as a result of this License or
out of the use or inability to use the Work (including but not limited to
damages for loss of goodwill, work stoppage, computer failure or malfunction, or
any and all other commercial damages or losses), even if such Contributor has
been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.
Apache License
==========

Version 2.0, January 2004
<<https://www.apache.org/licenses/>>

Terms and Conditions for use, reproduction, and distribution

1. Definitions

License shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

Licensor shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

Legal Entity shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, control means **(i)** the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or **(ii)** ownership of fifty percent (50%) or more of the outstanding shares, or **(iii)** beneficial ownership of such entity.

You (or Your) shall mean an individual or Legal Entity exercising permissions granted by this License.

Source form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

Object form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

Work shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

Derivative Works shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

Contribution shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, submitted means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as Not a Contribution.

Contributor shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution

You may reproduce and distribute copies of the Work or Derivative Works thereof
in any medium, with or without modifications, and in Source or Object form,
provided that You meet the following conditions:

* **(a)** You must give any other recipients of the Work or Derivative Works a copy of
this License; and
* **(b)** You must cause any modified files to carry prominent notices stating that You
changed the files; and
* **(c)** You must retain, in the Source form of any Derivative Works that You distribute,
all copyright, patent, trademark, and attribution notices from the Source form
of the Work, excluding those notices that do not pertain to any part of the
Derivative Works; and
* **(d)** If the Work includes a NOTICE text file as part of its distribution, then any
Derivative Works that You distribute must include a readable copy of the
attribution notices contained within such NOTICE file, excluding those notices
that do not pertain to any part of the Derivative Works, in at least one of the
following places: within a NOTICE text file distributed as part of the
Derivative Works; within the Source form or documentation, if provided along
with the Derivative Works; or, within a display generated by the Derivative
Works, if and wherever such third-party notices normally appear. The contents of
the NOTICE file are for informational purposes only and do not modify the
License. You may add Your own attribution notices within Derivative Works that
You distribute, alongside or as an addendum to the NOTICE text from the Work,
provided that such additional attribution notices cannot be construed as
modifying the License.

You may add Your own copyright statement to Your modifications and may provide
additional or different license terms and conditions for use, reproduction, or
distribution of Your modifications, or for any such Derivative Works as a whole,
provided Your use, reproduction, and distribution of the Work otherwise complies
with the conditions stated in this License.

5. Submission of Contributions

Unless You explicitly state otherwise, any Contribution intentionally submitted
for inclusion in the Work by You to the Licensor shall be under the terms and
conditions of this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify the terms of
any separate license agreement you may have executed with Licensor regarding
such Contributions.

6. Trademarks

This License does not grant permission to use the trade names, trademarks,
service marks, or product names of the Licensor, except as required for
reasonable and customary use in describing the origin of the Work and
reproducing the content of the NOTICE file.

7. Disclaimer of Warranty

Unless required by applicable law or agreed to in writing, Licensor provides the
Work (and each Contributor provides its Contributions) on an AS IS BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied,
including, without limitation, any warranties or conditions of TITLE,
NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are
solely responsible for determining the appropriateness of using or
redistributing the Work and assume any risks associated with Your exercise of
permissions under this License.

8. Limitation of Liability

In no event and under no legal theory, whether in tort (including negligence),
contract, or otherwise, unless required by applicable law (such as deliberate
and grossly negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special, incidental,
or consequential damages of any character arising as a result of this License or
out of the use or inability to use the Work (including but not limited to
damages for loss of goodwill, work stoppage, computer failure or malfunction, or
any and all other commercial damages or losses), even if such Contributor has
been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability

While redistributing the Work or Derivative Works thereof, You may choose to
offer, and charge a fee for, acceptance of support, warranty, indemnity, or
other liability obligations and/or rights consistent with this License. However,
in accepting such obligations, You may act only on Your own behalf and on Your
sole responsibility, not on behalf of any other Contributor, and only if You
agree to indemnify, defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason of your
accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work

To apply the Apache License to your work, attach the following boilerplate
notice, with the fields enclosed by brackets `[]` replaced with your own
identifying information. (Don't include the brackets!) The text should be
enclosed in the appropriate comment syntax for the file format. We also
recommend that a file or class name and description of purpose be included on
the same printed page as the copyright notice for easier identification within
third-party archives.

   Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

https://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

Eclipse Public License - v 2.0
==========

THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS ECLIPSE PUBLIC LICENSE (AGREEMENT). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.

1. Definitions

Contribution means:
* **a)** in the case of the initial Contributor, the initial content Distributed under this Agreement, and
* **b)** in the case of each subsequent Contributor:
 * **i)** changes to the Program, and
 * **ii)** additions to the Program;
where such changes and/or additions to the Program originate from and are Distributed by that particular Contributor. A Contribution originates from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor's behalf. Contributions do not include changes or additions to the Program that are not Modified Works.

Contributor means any person or entity that Distributes the Program.

Licensed Patents mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program.

Program means the Contributions Distributed in accordance with this Agreement.

Recipient means anyone who receives the Program under this Agreement or any Secondary License (as applicable), including Contributors.

Derivative Works shall mean any work, whether in Source Code or other form, that is based on (or derived from) the Program and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship.

Modified Works shall mean any work in Source Code or other form that results from an addition to, deletion from, or modification of the contents of the Program, including, for purposes of clarity any new file in Source Code form that contains any contents of the Program. Modified Works shall not include works that contain only declarations,

interfaces, types, classes, structures, or files of the Program solely in each case in order to link to, bind by name, or subclass the Program or Modified Works thereof.

Distribute means the acts of **a)** distributing or **b)** making available in any manner that enables the transfer of a copy.

Source Code means the form of a Program preferred for making modifications, including but not limited to software source code, documentation source, and configuration files.

Secondary License means either the GNU General Public License, Version 2.0, or any later versions of that license, including any exceptions or additional permissions as identified by the initial Contributor.

2. Grant of Rights

a) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, Distribute and sublicense the Contribution of such Contributor, if any, and such Derivative Works.

b) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in Source Code or other form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder.

c) Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to Distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.

d) Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement.

e) Notwithstanding the terms of any Secondary License, no Contributor makes additional grants to any Recipient (other than those set forth in this Agreement) as a result of such Recipient's receipt of the Program under the terms of a Secondary License (if permitted under the terms of Section 3).

3. Requirements

3.1 If a Contributor Distributes the Program in any form, then:

* **a)** the Program must also be made available as Source Code, in accordance with section 3.2, and the Contributor must accompany the Program with a statement that the Source Code for the Program is available under this Agreement, and informs Recipients how to obtain it in a reasonable manner on or through a medium customarily used for software exchange; and

* **b)** the Contributor may Distribute the Program under a license different than this Agreement, provided that such license:
 * **i)** effectively disclaims on behalf of all other Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose;
 * **ii)** effectively excludes on behalf of all other Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits;
 * **iii)** does not attempt to limit or alter the recipients' rights in the Source Code under section 3.2; and
 * **iv)** requires any subsequent distribution of the Program by any party to be under a license that satisfies the requirements of this section 3.

3.2 When the Program is Distributed as Source Code:

* **a)** it must be made available under this Agreement, or if the Program **(i)** is combined with other material in a separate file or files made available under a Secondary License, and **(ii)** the initial Contributor attached to the Source Code the notice described in Exhibit A of this Agreement, then the Program may be made available under the terms of such Secondary Licenses, and
* **b)** a copy of this Agreement must be included with each copy of the Program.

3.3 Contributors may not remove or alter any copyright, patent, trademark, attribution notices, disclaimers of warranty, or limitations of liability (notices) contained within the Program from any copy of the Program which they Distribute, provided that Contributors may add their own appropriate notices.

4. Commercial Distribution

Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor (Commercial Contributor) hereby agrees to defend and indemnify every other Contributor (Indemnified Contributor) against any losses, damages and costs (collectively Losses) arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must: **a)** promptly notify the Commercial Contributor in writing of such claim, and **b)** allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those damages.

5. No Warranty

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, AND TO THE EXTENT PERMITTED BY APPLICABLE LAW, THE PROGRAM IS PROVIDED ON AN AS IS BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

6. Disclaimer of Liability

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, AND TO THE EXTENT PERMITTED BY APPLICABLE LAW, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. General

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. The Eclipse Foundation is the initial Agreement Steward. The Eclipse Foundation may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be Distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to Distribute the Program (including its Contributions) under the new version.

Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved. Nothing in this Agreement is intended to be enforceable by any entity that is not a Contributor or Recipient. No third-party beneficiary rights are created under this Agreement.

Exhibit A - Form of Secondary Licenses Notice

> This Source Code may also be made available under the following Secondary Licenses when the conditions for such availability set forth in the Eclipse Public License, v. 2.0 are satisfied: {name license(s), version(s), and exceptions or additional permissions here}.

Simply including a copy of this Agreement, including this Exhibit A is not sufficient to license the Source Code under Secondary Licenses.

If it is not possible or desirable to put the notice in a particular file, then You may include the notice in a location (such as a LICENSE file in a relevant directory) where a recipient would be likely to look for such a notice.

You may add additional accurate notices of copyright ownership.
Open Source Licenses
==========

This product may include a number of subcomponents with separate
copyright notices and license terms. Your use of the source code for
these subcomponents is subject to the terms and conditions of the
subcomponent's license, as noted in the LICENSE-<subcomponent>.md
files.
[[contributors]]
== Contributors

Browse the {junit5-repo}/graphs/contributors[current list of contributors] directly on GitHub.

1.26 urllib3 1.26.7

1.26.1 Available under license :

MIT License

Copyright (c) 2008-2020 Andrey Petrov and contributors (see CONTRIBUTORS.txt)

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all

copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.27 tdigestsketch 0.20.0

1.27.1 Available under license :

Apache Druid
Copyright 2011-2020 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made,

use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions

for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability

incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following
boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.28 t-digest 3.2

1.28.1 Available under license :

```
/*
* Licensed to Ted Dunning under one or more
* contributor license agreements.  See the NOTICE file distributed with
* this work for additional information regarding copyright ownership.
* The ASF licenses this file to You under the Apache License, Version 2.0
* (the "License"); you may not use this file except in compliance with
* the License.  You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
```

* limitations under the License.
*/

1.29 jackson-datatype-guava 2.10.3

1.29.1 Available under license :

Apache-2.0

1.30 apache-commons-math 3.6.1

1.30.1 Available under license :

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or

Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work

or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work

by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following
boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

Apache Commons Math includes the following code provided to the ASF under the
Apache License 2.0:

- The inverse error function implementation in the Erf class is based on CUDA
  code developed by Mike Giles, Oxford-Man Institute of Quantitative Finance,
  and published in GPU Computing Gems, volume 2, 2010 (grant received on
  March 23th 2013)
- The LinearConstraint, LinearObjectiveFunction, LinearOptimizer,
  RelationShip, SimplexSolver and SimplexTableau classes in package
  org.apache.commons.math3.optimization.linear include software developed by
  Benjamin McCann (http://www.benmccann.com) and distributed with
  the following copyright: Copyright 2009 Google Inc. (grant received on
  March 16th 2009)
- The class "org.apache.commons.math3.exception.util.LocalizedFormatsTest" which
  is an adapted version of "OrekitMessagesTest" test class for the Orekit library
- The "org.apache.commons.math3.analysis.interpolation.HermiteInterpolator"
  has been imported from the Orekit space flight dynamics library.

==========

APACHE COMMONS MATH DERIVATIVE WORKS:

The Apache commons-math library includes a number of subcomponents
whose implementation is derived from original sources written
in C or Fortran. License terms of the original sources
are reproduced below.

==========

For the lmder, lmpar and qrsolv Fortran routine from minpack and translated in
the LevenbergMarquardtOptimizer class in package
org.apache.commons.math3.optimization.general
Original source copyright and license statement:

Minpack Copyright Notice (1999) University of Chicago. All rights reserved

Redistribution and use in source and binary forms, with or
without modification, are permitted provided that the
following conditions are met:

1. Redistributions of source code must retain the above
copyright notice, this list of conditions and the following
disclaimer.

2. Redistributions in binary form must reproduce the above
copyright notice, this list of conditions and the following
disclaimer in the documentation and/or other materials
provided with the distribution.

3. The end-user documentation included with the
redistribution, if any, must include the following
acknowledgment:

  "This product includes software developed by the
  University of Chicago, as Operator of Argonne National
  Laboratory.

Alternately, this acknowledgment may appear in the software
itself, if and wherever such third-party acknowledgments
normally appear.

4. WARRANTY DISCLAIMER. THE SOFTWARE IS SUPPLIED "AS IS"
WITHOUT WARRANTY OF ANY KIND. THE COPYRIGHT HOLDER, THE
UNITED STATES, THE UNITED STATES DEPARTMENT OF ENERGY, AND
THEIR EMPLOYEES: (1) DISCLAIM ANY WARRANTIES, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES
OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE
OR NON-INFRINGEMENT, (2) DO NOT ASSUME ANY LEGAL LIABILITY
OR RESPONSIBILITY FOR THE ACCURACY, COMPLETENESS, OR
USEFULNESS OF THE SOFTWARE, (3) DO NOT REPRESENT THAT USE OF
THE SOFTWARE WOULD NOT INFRINGE PRIVATELY OWNED RIGHTS, (4)

DO NOT WARRANT THAT THE SOFTWARE WILL FUNCTION
UNINTERRUPTED, THAT IT IS ERROR-FREE OR THAT ANY ERRORS WILL
BE CORRECTED.

5. LIMITATION OF LIABILITY. IN NO EVENT WILL THE COPYRIGHT
HOLDER, THE UNITED STATES, THE UNITED STATES DEPARTMENT OF
ENERGY, OR THEIR EMPLOYEES: BE LIABLE FOR ANY INDIRECT,
INCIDENTAL, CONSEQUENTIAL, SPECIAL OR PUNITIVE DAMAGES OF
ANY KIND OR NATURE, INCLUDING BUT NOT LIMITED TO LOSS OF
PROFITS OR LOSS OF DATA, FOR ANY REASON WHATSOEVER, WHETHER
SUCH LIABILITY IS ASSERTED ON THE BASIS OF CONTRACT, TORT
(INCLUDING NEGLIGENCE OR STRICT LIABILITY), OR OTHERWISE,
EVEN IF ANY OF SAID PARTIES HAS BEEN WARNED OF THE
POSSIBILITY OF SUCH LOSS OR DAMAGES.
==========

Copyright and license statement for the odex Fortran routine developed by
E. Hairer and G. Wanner and translated in GraggBulirschStoerIntegrator class
in package org.apache.commons.math3.ode.nonstiff:

Copyright (c) 2004, Ernst Hairer

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

- Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
==========

Copyright and license statement for the original Mersenne twister C

routines translated in MersenneTwister class in package
org.apache.commons.math3.random:

Copyright (C) 1997 - 2002, Makoto Matsumoto and Takuji Nishimura,
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.

3. The names of its contributors may not be used to endorse or promote
products derived from this software without specific prior written
permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

==========

The initial code for shuffling an array (originally in class
"org.apache.commons.math3.random.RandomDataGenerator", now replaced by
a method in class "org.apache.commons.math3.util.MathArrays") was
inspired from the algorithm description provided in
"Algorithms", by Ian Craw and John Pulham (University of Aberdeen 1999).
The textbook (containing a proof that the shuffle is uniformly random) is
available here:
http://citeseerx.ist.psu.edu/viewdoc/download;?doi=10.1.1.173.1898&rep=rep1&type=pdf

==========
License statement for the direction numbers in the resource files for Sobol sequences.

Licence pertaining to sobol.cc and the accompanying sets of direction numbers

---
Copyright (c) 2008, Frances Y. Kuo and Stephen Joe
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

  * Redistributions of source code must retain the above copyright
    notice, this list of conditions and the following disclaimer.

  * Redistributions in binary form must reproduce the above copyright
    notice, this list of conditions and the following disclaimer in the
    documentation and/or other materials provided with the distribution.

  * Neither the names of the copyright holders nor the names of the
    University of New South Wales and the University of Waikato
    and its contributors may be used to endorse or promote products derived
    from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS ``AS IS" AND ANY
EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDERS BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
==========

The initial commit of package "org.apache.commons.math3.ml.neuralnet" is
an adapted version of code developed in the context of the Data Processing
and Analysis Consortium (DPAC) of the "Gaia" project of the European Space
Agency (ESA).
==========

The initial commit of the class "org.apache.commons.math3.special.BesselJ" is
an adapted version of code translated from the netlib Fortran program, rjbesl
http://www.netlib.org/specfun/rjbesl by R.J. Cody at Argonne National
Laboratory (USA). There is no license or copyright statement included with the
original Fortran sources.
==========

The BracketFinder (package org.apache.commons.math3.optimization.univariate)

and PowellOptimizer (package org.apache.commons.math3.optimization.general)
classes are based on the Python code in module "optimize.py" (version 0.5)
developed by Travis E. Oliphant for the SciPy library (http://www.scipy.org/)
Copyright 2003-2009 SciPy Developers.

SciPy license
Copyright 2001, 2002 Enthought, Inc.
All rights reserved.

Copyright 2003-2013 SciPy Developers.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright
  notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright
  notice, this list of conditions and the following disclaimer in the
  documentation and/or other materials provided with the distribution.

* Neither the name of Enthought nor the names of the SciPy Developers may
  be used to endorse or promote products derived from this software without
  specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS AND ANY
EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
==========
Apache Commons Math
Copyright 2001-2016 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

This product includes software developed for Orekit by
CS Systmes d'Information (http://www.c-s.fr/)
Copyright 2010-2012 CS Systmes d'Information

1.31 asm-commons 7.0

1.31.1 Available under license :

<OWNER> = Regents of the University of California
<ORGANIZATION> = University of California, Berkeley
<YEAR> = 1998

In the original BSD license, both occurrences of the phrase "COPYRIGHT HOLDERS AND CONTRIBUTORS" in the disclaimer read "REGENTS AND CONTRIBUTORS".

Here is the license template:

Copyright (c) <YEAR>, <OWNER>

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
Neither the name of the <ORGANIZATION> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

1.32 metrics-integration-for-apache-httpasyncclient 4.0.5

1.32.1 Available under license :

No license file was found, but licenses were detected in source scan.

Manifest-Version: 1.0

Bnd-LastModified: 1545937994519
Build-Jdk: 1.8.0_191
Built-By: artem
Bundle-Description: An Apache HttpAsyncClient wrapper providing Metric
s instrumentation of connection pools, request durations and r
ates, and other useful information.
Bundle-License: http://www.apache.org/licenses/LICENSE-2.0.html
Bundle-ManifestVersion: 2
Bundle-Name: Metrics Integration for Apache HttpAsyncClient
Bundle-SymbolicName: io.dropwizard.metrics.httpasyncclient
Bundle-Version: 4.0.5
Created-By: Apache Maven Bundle Plugin
Export-Package: com.codahale.metrics.httpasyncclient;uses:="com.codaha
le.metrics,com.codahale.metrics.httpclient,org.apache.http.config,org
.apache.http.conn,org.apache.http.impl.nio.client,org.apache.http.imp
l.nio.conn,org.apache.http.nio.conn,org.apache.http.nio.reactor";vers
ion="4.0.5"
Implementation-Title: Metrics Integration for Apache HttpAsyncClient
Implementation-URL: http://metrics.dropwizard.io/metrics-httpasyncclie
nt
Implementation-Vendor-Id: io.dropwizard.metrics
Implementation-Version: 4.0.5
Import-Package: com.codahale.metrics;version="[4.0,5)",com.codahale.me
trics.httpclient;version="[4.0,5)",org.apache.http,org.apache.http.co
ncurrent,org.apache.http.config,org.apache.http.conn,org.apache.http.
impl.nio.client,org.apache.http.impl.nio.conn,org.apache.http.nio.con
n,org.apache.http.nio.protocol,org.apache.http.nio.reactor,org.apache
.http.pool,org.apache.http.protocol
Require-Capability: osgi.ee;filter:="(&(osgi.ee=JavaSE)(version=1.8))"
Tool: Bnd-3.3.0.201609221906

Found in path(s):
* /opt/cola/permits/1340031649_1654689479.4970224/0/metrics-httpasyncclient-4-0-5-jar/META-INF/MANIFEST.MF

1.33 shims 0.8.11

1.33.1 Available under license :

Apache-2.0

1.34 apache-http-client 4.5.13

1.34.1 Available under license :

Apache HttpComponents Client
Copyright 1999-2020 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity
on behalf of whom a Contribution has been received by Licensor and
subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
copyright license to reproduce, prepare Derivative Works of,
publicly display, publicly perform, sublicense, and distribute the
Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
(except as stated in this section) patent license to make, have made,
use, offer to sell, sell, import, and otherwise transfer the Work,
where such license applies only to those patent claims licensable
by such Contributor that are necessarily infringed by their
Contribution(s) alone or by combination of their Contribution(s)
with the Work to which such Contribution(s) was submitted. If You
institute patent litigation against any entity (including a
cross-claim or counterclaim in a lawsuit) alleging that the Work
or a Contribution incorporated within the Work constitutes direct
or contributory patent infringement, then any patent licenses
granted to You under this License for that Work shall terminate
as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
Work or Derivative Works thereof in any medium, with or without
modifications, and in Source or Object form, provided that You
meet the following conditions:

(a) You must give any other recipients of the Work or

Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
agreed to in writing, Licensor provides the Work (and each
Contributor provides its Contributions) on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied, including, without limitation, any warranties or conditions
of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
PARTICULAR PURPOSE. You are solely responsible for determining the
appropriateness of using or redistributing the Work and assume any
risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
whether in tort (including negligence), contract, or otherwise,
unless required by applicable law (such as deliberate and grossly
negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special,
incidental, or consequential damages of any character arising as a
result of this License or out of the use or inability to use the
Work (including but not limited to damages for loss of goodwill,
work stoppage, computer failure or malfunction, or any and all
other commercial damages or losses), even if such Contributor
has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
the Work or Derivative Works thereof, You may choose to offer,
and charge a fee for, acceptance of support, warranty, indemnity,
or other liability obligations and/or rights consistent with this
License. However, in accepting such obligations, You may act only
on Your own behalf and on Your sole responsibility, not on behalf
of any other Contributor, and only if You agree to indemnify,
defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

==========

This project includes Public Suffix List copied from
<https://publicsuffix.org/list/effective_tld_names.dat>
licensed under the terms of the Mozilla Public License, v. 2.0

Full license text: <http://mozilla.org/MPL/2.0/>

Mozilla Public License Version 2.0
==========

1. Definitions

1.1. "Contributor"
means each individual or legal entity that creates, contributes to the creation of, or owns Covered Software.

1.2. "Contributor Version"
means the combination of the Contributions of others (if any) used by a Contributor and that particular Contributor's Contribution.

1.3. "Contribution"
means Covered Software of a particular Contributor.

1.4. "Covered Software"
means Source Code Form to which the initial Contributor has attached the notice in Exhibit A, the Executable Form of such Source Code Form, and Modifications of such Source Code Form, in each case including portions thereof.

1.5. "Incompatible With Secondary Licenses"
means

(a) that the initial Contributor has attached the notice described in Exhibit B to the Covered Software; or

(b) that the Covered Software was made available under the terms of version 1.1 or earlier of the License, but not also under the terms of a Secondary License.

1.6. "Executable Form"
means any form of the work other than Source Code Form.

1.7. "Larger Work"
means a work that combines Covered Software with other material, in a separate file or files, that is not Covered Software.

1.8. "License"
means this document.

1.9. "Licensable"
means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently, any and all of the rights conveyed by this License.

1.10. "Modifications"
means any of the following:

(a) any file in Source Code Form that results from an addition to, deletion from, or modification of the contents of Covered

Software; or

(b) any new file in Source Code Form that contains any Covered Software.

1.11. "Patent Claims" of a Contributor
means any patent claim(s), including without limitation, method, process, and apparatus claims, in any patent Licensable by such Contributor that would be infringed, but for the grant of the License, by the making, using, selling, offering for sale, having made, import, or transfer of either its Contributions or its Contributor Version.

1.12. "Secondary License"
means either the GNU General Public License, Version 2.0, the GNU Lesser General Public License, Version 2.1, the GNU Affero General Public License, Version 3.0, or any later versions of those licenses.

1.13. "Source Code Form"
means the form of the work preferred for making modifications.

1.14. "You" (or "Your")
means an individual or a legal entity exercising rights under this License. For legal entities, "You" includes any entity that controls, is controlled by, or is under common control with You. For purposes of this definition, "control" means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. License Grants and Conditions

2.1. Grants

Each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:

(a) under intellectual property rights (other than patent or trademark) Licensable by such Contributor to use, reproduce, make available, modify, display, perform, distribute, and otherwise exploit its Contributions, either on an unmodified basis, with Modifications, or as part of a Larger Work; and

(b) under Patent Claims of such Contributor to make, use, sell, offer for sale, have made, import, and otherwise transfer either its

Contributions or its Contributor Version.

2.2. Effective Date

The licenses granted in Section 2.1 with respect to any Contribution become effective for each Contribution on the date the Contributor first distributes such Contribution.

2.3. Limitations on Grant Scope

The licenses granted in this Section 2 are the only rights granted under this License. No additional rights or licenses will be implied from the distribution or licensing of Covered Software under this License. Notwithstanding Section 2.1(b) above, no patent license is granted by a Contributor:

(a) for any code that a Contributor has removed from Covered Software; or

(b) for infringements caused by: (i) Your and any other third party's modifications of Covered Software, or (ii) the combination of its Contributions with other software (except as part of its Contributor Version); or

(c) under Patent Claims infringed by Covered Software in the absence of its Contributions.

This License does not grant any rights in the trademarks, service marks, or logos of any Contributor (except as may be necessary to comply with the notice requirements in Section 3.4).

2.4. Subsequent Licenses

No Contributor makes additional grants as a result of Your choice to distribute the Covered Software under a subsequent version of this License (see Section 10.2) or under the terms of a Secondary License (if permitted under the terms of Section 3.3).

2.5. Representation

Each Contributor represents that the Contributor believes its Contributions are its original creation(s) or it has sufficient rights to grant the rights to its Contributions conveyed by this License.

2.6. Fair Use

This License is not intended to limit any rights You have under applicable copyright doctrines of fair use, fair dealing, or other

equivalents.

2.7. Conditions

Sections 3.1, 3.2, 3.3, and 3.4 are conditions of the licenses granted in Section 2.1.

3. Responsibilities

3.1. Distribution of Source Form

All distribution of Covered Software in Source Code Form, including any Modifications that You create or to which You contribute, must be under the terms of this License. You must inform recipients that the Source Code Form of the Covered Software is governed by the terms of this License, and how they can obtain a copy of this License. You may not attempt to alter or restrict the recipients' rights in the Source Code Form.

3.2. Distribution of Executable Form

If You distribute Covered Software in Executable Form then:

(a) such Covered Software must also be made available in Source Code Form, as described in Section 3.1, and You must inform recipients of the Executable Form how they can obtain a copy of such Source Code Form by reasonable means in a timely manner, at a charge no more than the cost of distribution to the recipient; and

(b) You may distribute such Executable Form under the terms of this License, or sublicense it under different terms, provided that the license for the Executable Form does not attempt to limit or alter the recipients' rights in the Source Code Form under this License.

3.3. Distribution of a Larger Work

You may create and distribute a Larger Work under terms of Your choice, provided that You also comply with the requirements of this License for the Covered Software. If the Larger Work is a combination of Covered Software with a work governed by one or more Secondary Licenses, and the Covered Software is not Incompatible With Secondary Licenses, this License permits You to additionally distribute such Covered Software under the terms of such Secondary License(s), so that the recipient of the Larger Work may, at their option, further distribute the Covered Software under the terms of either this License or such Secondary License(s).

3.4. Notices

You may not remove or alter the substance of any license notices (including copyright notices, patent notices, disclaimers of warranty, or limitations of liability) contained within the Source Code Form of the Covered Software, except that You may alter any license notices to the extent required to remedy known factual inaccuracies.

3.5. Application of Additional Terms

You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However, You may do so only on Your own behalf, and not on behalf of any Contributor. You must make it absolutely clear that any such warranty, support, indemnity, or liability obligation is offered by You alone, and You hereby agree to indemnify every Contributor for any liability incurred by such Contributor as a result of warranty, support, indemnity or liability terms You offer. You may include additional disclaimers of warranty and limitations of liability specific to any jurisdiction.

4. Inability to Comply Due to Statute or Regulation

If it is impossible for You to comply with any of the terms of this License with respect to some or all of the Covered Software due to statute, judicial order, or regulation then You must: (a) comply with the terms of this License to the maximum extent possible; and (b) describe the limitations and the code they affect. Such description must be placed in a text file included with all distributions of the Covered Software under this License. Except to the extent prohibited by statute or regulation, such description must be sufficiently detailed for a recipient of ordinary skill to be able to understand it.

5. Termination

5.1. The rights granted under this License will terminate automatically if You fail to comply with any of its terms. However, if You become compliant, then the rights granted under this License from a particular Contributor are reinstated (a) provisionally, unless and until such Contributor explicitly and finally terminates Your grants, and (b) on an ongoing basis, if such Contributor fails to notify You of the non-compliance by some reasonable means prior to 60 days after You have come back into compliance. Moreover, Your grants from a particular Contributor are reinstated on an ongoing basis if such Contributor notifies You of the non-compliance by some reasonable means, this is the first time You have received notice of non-compliance with this License

from such Contributor, and You become compliant prior to 30 days after Your receipt of the notice.

5.2. If You initiate litigation against any entity by asserting a patent infringement claim (excluding declaratory judgment actions, counter-claims, and cross-claims) alleging that a Contributor Version directly or indirectly infringes any patent, then the rights granted to You by any and all Contributors for the Covered Software under Section 2.1 of this License shall terminate.

5.3. In the event of termination under Sections 5.1 or 5.2 above, all end user license agreements (excluding distributors and resellers) which have been validly granted by You or Your distributors under this License prior to termination shall survive termination.

```
************************************************************************
*                                                                      *
* 6. Disclaimer of Warranty                                            *
* -------------------------                                            *
*                                                                      *
* Covered Software is provided under this License on an "as is"        *
* basis, without warranty of any kind, either expressed, implied, or   *
* statutory, including, without limitation, warranties that the        *
* Covered Software is free of defects, merchantable, fit for a         *
* particular purpose or non-infringing. The entire risk as to the      *
* quality and performance of the Covered Software is with You.         *
* Should any Covered Software prove defective in any respect, You      *
* (not any Contributor) assume the cost of any necessary servicing,    *
* repair, or correction. This disclaimer of warranty constitutes an    *
* essential part of this License. No use of any Covered Software is    *
* authorized under this License except under this disclaimer.          *
*                                                                      *
************************************************************************

************************************************************************
*                                                                      *
* 7. Limitation of Liability                                           *
* --------------------------                                           *
*                                                                      *
* Under no circumstances and under no legal theory, whether tort       *
* (including negligence), contract, or otherwise, shall any            *
* Contributor, or anyone who distributes Covered Software as           *
* permitted above, be liable to You for any direct, indirect,          *
* special, incidental, or consequential damages of any character       *
* including, without limitation, damages for lost profits, loss of     *
* goodwill, work stoppage, computer failure or malfunction, or any     *
* and all other commercial damages or losses, even if such party       *
* shall have been informed of the possibility of such damages. This    *
```

* limitation of liability shall not apply to liability for death or *
* personal injury resulting from such party's negligence to the *
* extent applicable law prohibits such limitation. Some *
* jurisdictions do not allow the exclusion or limitation of *
* incidental or consequential damages, so this exclusion and *
* limitation may not apply to You. *
* *
****

8. Litigation

Any litigation relating to this License may be brought only in the courts of a jurisdiction where the defendant maintains its principal place of business and such litigation shall be governed by laws of that jurisdiction, without reference to its conflict-of-law provisions. Nothing in this Section shall prevent a party's ability to bring cross-claims or counter-claims.

9. Miscellaneous

This License represents the complete agreement concerning the subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not be used to construe this License against a Contributor.

10. Versions of the License

10.1. New Versions

Mozilla Foundation is the license steward. Except as provided in Section 10.3, no one other than the license steward has the right to modify or publish new versions of this License. Each version will be given a distinguishing version number.

10.2. Effect of New Versions

You may distribute the Covered Software under the terms of the version of the License under which You originally received the Covered Software, or under the terms of any subsequent version published by the license steward.

10.3. Modified Versions

If you create software not governed by this License, and you want to create a new license for such software, you may create and use a modified version of this License if you rename the license and remove any references to the name of the license steward (except to note that such modified license differs from this License).

10.4. Distributing Source Code Form that is Incompatible With Secondary Licenses

If You choose to distribute Source Code Form that is Incompatible With Secondary Licenses under the terms of this version of the License, the notice described in Exhibit B of this License must be attached.

Exhibit A - Source Code Form License Notice

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at http://mozilla.org/MPL/2.0/.

If it is not possible or desirable to put the notice in a particular file, then You may include the notice in a location (such as a LICENSE file in a relevant directory) where a recipient would be likely to look for such a notice.

You may add additional accurate notices of copyright ownership.

Exhibit B - "Incompatible With Secondary Licenses" Notice

This Source Code Form is "Incompatible With Secondary Licenses", as defined by the Mozilla Public License, v. 2.0.

1.35 utils 5.3.1

1.35.1 Available under license :

MIT License

Copyright (c) 2019 TypeScript ESLint and other contributors

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.36 expression-language 3.0.0

1.36.1 Available under license :

Copyright (c) 2004-2015 Fabien Potencier

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is furnished
to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.37 zkclient 0.10

1.37.1 Available under license :

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1

through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their

Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

You must give any other recipients of the Work or Derivative Works a copy of this License; and

You must cause any modified files to carry prominent notices stating that You changed the files; and

You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.
You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.
5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or

otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic

mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

You must give any other recipients of the Work or Derivative Works a copy of this License; and

You must cause any modified files to carry prominent notices stating that You changed the files; and

You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.
You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.
5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for

inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.38 system-stubs-jupiter 1.1.0

1.38.1 Available under license :

MIT

1.39 project-lombok 1.18.8

1.39.1 Available under license :

Copyright (C) 2009-2015 The Project Lombok Authors.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.

1.40 netty/transport/classes/epoll 4.1.84.Final

1.40.1 Available under license :

No license file was found, but licenses were detected in source scan.

Manifest-Version: 1.0
Implementation-Title: Netty/Codec/HTTP
Bundle-Description: Netty is an asynchronous event-driven network appl
ication framework for rapid development of maintainable high perfo
rmance protocol servers and clients.

Automatic-Module-Name: io.netty.codec.http
Bundle-License: https://www.apache.org/licenses/LICENSE-2.0
Bundle-SymbolicName: io.netty.codec-http
Implementation-Version: 4.1.84.Final
Built-By: chris
Bnd-LastModified: 1665536154725
Bundle-ManifestVersion: 2
Implementation-Vendor-Id: io.netty
Bundle-DocURL: https://netty.io/
Bundle-Vendor: The Netty Project
Import-Package: com.aayushatharva.brotli4j.encoder;resolution:=optiona
l,com.jcraft.jzlib;resolution:=optional,io.netty.buffer;version="[4.1
,5)",io.netty.channel;version="[4.1,5)",io.netty.channel.embedded;ver
sion="[4.1,5)",io.netty.handler.codec,io.netty.handler.codec.base64;v
ersion="[4.1,5)",io.netty.handler.codec.compression;version="[4.1,5)"
,io.netty.handler.ssl;version="[4.1,5)",io.netty.handler.stream;versi
on="[4.1,5)",io.netty.util;version="[4.1,5)",io.netty.util.concurrent
;version="[4.1,5)",io.netty.util.internal;version="[4.1,5)",io.netty.
util.internal.logging;version="[4.1,5)",sun.nio.ch;resolution:=option
al,org.eclipse.jetty.npn;version="[1,2)";resolution:=optional,org.ecl
ipse.jetty.alpn;version="[1,2)";resolution:=optional
Require-Capability: osgi.ee;filter:="(&(osgi.ee=JavaSE)(version=1.6))"
Tool: Bnd-2.4.1.201501161923
Implementation-Vendor: The Netty Project
Export-Package: io.netty.handler.codec.http;uses:="io.netty.buffer,io.
netty.channel,io.netty.channel.embedded,io.netty.handler.codec,io.net
ty.handler.codec.compression,io.netty.handler.codec.http.cookie,io.ne
tty.handler.stream,io.netty.util";version="4.1.84",io.netty.handler.c
odec.http.cookie;version="4.1.84",io.netty.handler.codec.http.cors;us
es:="io.netty.channel,io.netty.handler.codec.http";version="4.1.84",i
o.netty.handler.codec.http.multipart;uses:="io.netty.buffer,io.netty.
channel,io.netty.handler.codec,io.netty.handler.codec.http,io.netty.h
andler.stream,io.netty.util";version="4.1.84",io.netty.handler.codec.
http.websocketx;uses:="io.netty.buffer,io.netty.channel,io.netty.hand
ler.codec,io.netty.handler.codec.http,io.netty.handler.stream,io.nett
y.util,io.netty.util.internal.logging";version="4.1.84",io.netty.hand
ler.codec.http.websocketx.extensions;uses:="io.netty.channel,io.netty
.handler.codec,io.netty.handler.codec.http.websocketx";version="4.1.8
4",io.netty.handler.codec.http.websocketx.extensions.compression;uses
:="io.netty.channel,io.netty.handler.codec.http.websocketx.extensions
";version="4.1.84",io.netty.handler.codec.rtsp;uses:="io.netty.buffer
,io.netty.channel,io.netty.handler.codec.http,io.netty.util";version=
"4.1.84",io.netty.handler.codec.spdy;uses:="io.netty.buffer,io.netty.
channel,io.netty.handler.codec,io.netty.handler.codec.http,io.netty.u
til";version="4.1.84"
Bundle-Name: Netty/Codec/HTTP
Bundle-Version: 4.1.84.Final
Created-By: Apache Maven Bundle Plugin

Build-Jdk: 1.8.0_312
Implementation-URL: https://netty.io/netty-codec-http/

Found in path(s):
* /opt/cola/permits/1470280885_1668115952.2421944/0/netty-codec-http-4-1-84-final-jar/META-INF/MANIFEST.MF
No license file was found, but licenses were detected in source scan.

<!--
~ Copyright 2012 The Netty Project
~
~ The Netty Project licenses this file to you under the Apache License,
~ version 2.0 (the "License"); you may not use this file except in compliance
~ with the License. You may obtain a copy of the License at:
~
~   https://www.apache.org/licenses/LICENSE-2.0
~
~ Unless required by applicable law or agreed to in writing, software
~ distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
~ WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
~ License for the specific language governing permissions and limitations
~ under the License.
-->

Found in path(s):
* /opt/cola/permits/1470280885_1668115952.2421944/0/netty-codec-http-4-1-84-final-jar/META-INF/maven/io.netty/netty-codec-http/pom.xml
No license file was found, but licenses were detected in source scan.

The Netty Project licenses this file to you under the Apache License,
version 2.0 (the "License"); you may not use this file except in compliance
with the License. You may obtain a copy of the License at:
distributed under the License is distributed on an "AS IS" BASIS, WITHOUT

Found in path(s):
* /opt/cola/permits/1470280885_1668115952.2421944/0/netty-codec-http-4-1-84-final-jar/META-INF/native-image/io.netty/codec-http/native-image.properties

1.41 junit-jupiter-junit-jupiter-api 5.7.2

1.41.1 Available under license :

import java.io.File
import java.net.URI

data class License(val name: String, val url: URI, val headerFile: File)
Apache License
==========

Version 2.0, January 2004
<<https://www.apache.org/licenses/>>

Terms and Conditions for use, reproduction, and distribution

1. Definitions

License shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

Licensor shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

Legal Entity shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, control means **(i)** the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or **(ii)** ownership of fifty percent (50%) or more of the outstanding shares, or **(iii)** beneficial ownership of such entity.

You (or Your) shall mean an individual or Legal Entity exercising permissions granted by this License.

Source form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

Object form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

Work shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

Derivative Works shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

Contribution shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition,

submitted means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems, and
issue tracking systems that are managed by, or on behalf of, the Licensor for
the purpose of discussing and improving the Work, but excluding communication
that is conspicuously marked or otherwise designated in writing by the copyright
owner as Not a Contribution.

Contributor shall mean Licensor and any individual or Legal Entity on behalf
of whom a Contribution has been received by Licensor and subsequently
incorporated within the Work.

2. Grant of Copyright License

Subject to the terms and conditions of this License, each Contributor hereby
grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free,
irrevocable copyright license to reproduce, prepare Derivative Works of,
publicly display, publicly perform, sublicense, and distribute the Work and such
Derivative Works in Source or Object form.

3. Grant of Patent License

Subject to the terms and conditions of this License, each Contributor hereby
grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free,
irrevocable (except as stated in this section) patent license to make, have
made, use, offer to sell, sell, import, and otherwise transfer the Work, where
such license applies only to those patent claims licensable by such Contributor
that are necessarily infringed by their Contribution(s) alone or by combination
of their Contribution(s) with the Work to which such Contribution(s) was
submitted. If You institute patent litigation against any entity (including a
cross-claim or counterclaim in a lawsuit) alleging that the Work or a
Contribution incorporated within the Work constitutes direct or contributory
patent infringement, then any patent licenses granted to You under this License
for that Work shall terminate as of the date such litigation is filed.

4. Redistribution

You may reproduce and distribute copies of the Work or Derivative Works thereof
in any medium, with or without modifications, and in Source or Object form,
provided that You meet the following conditions:

* **(a)** You must give any other recipients of the Work or Derivative Works a copy of
this License; and
* **(b)** You must cause any modified files to carry prominent notices stating that You
changed the files; and
* **(c)** You must retain, in the Source form of any Derivative Works that You distribute,
all copyright, patent, trademark, and attribution notices from the Source form
of the Work, excluding those notices that do not pertain to any part of the

Derivative Works; and
* **(d)** If the Work includes a NOTICE text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an AS IS BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.
Apache License
==========

Version 2.0, January 2004
<<https://www.apache.org/licenses/>>

Terms and Conditions for use, reproduction, and distribution

1. Definitions

License shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

Licensor shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

Legal Entity shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, control means **(i)** the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or **(ii)** ownership of fifty percent (50%) or more of the outstanding shares, or **(iii)** beneficial ownership of such entity.

You (or Your) shall mean an individual or Legal Entity exercising permissions granted by this License.

Source form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

Object form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

Work shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

Derivative Works shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

Contribution shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, submitted means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as Not a Contribution.

Contributor shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free,

irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

* **(a)** You must give any other recipients of the Work or Derivative Works a copy of this License; and
* **(b)** You must cause any modified files to carry prominent notices stating that You changed the files; and
* **(c)** You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
* **(d)** If the Work includes a NOTICE text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions

Unless You explicitly state otherwise, any Contribution intentionally submitted

for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an AS IS BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work

To apply the Apache License to your work, attach the following boilerplate
notice, with the fields enclosed by brackets `[]` replaced with your own
identifying information. (Don't include the brackets!) The text should be
enclosed in the appropriate comment syntax for the file format. We also
recommend that a file or class name and description of purpose be included on
the same printed page as the copyright notice for easier identification within
third-party archives.

  Copyright [yyyy] [name of copyright owner]

  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    https://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
Eclipse Public License - v 2.0
==========

THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS ECLIPSE PUBLIC LICENSE (AGREEMENT). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.

1. Definitions

Contribution means:
* **a)** in the case of the initial Contributor, the initial content Distributed under this Agreement, and
* **b)** in the case of each subsequent Contributor:
 * **i)** changes to the Program, and
 * **ii)** additions to the Program;
where such changes and/or additions to the Program originate from and are Distributed by that particular Contributor. A Contribution originates from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor's behalf. Contributions do not include changes or additions to the Program that are not Modified Works.

Contributor means any person or entity that Distributes the Program.

Licensed Patents mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program.

Program means the Contributions Distributed in accordance with this Agreement.

Recipient means anyone who receives the Program under this Agreement or any Secondary License (as applicable), including Contributors.

Derivative Works shall mean any work, whether in Source Code or other form, that is based on (or derived from) the Program and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship.

Modified Works shall mean any work in Source Code or other form that results from an addition to, deletion from, or modification of the contents of the Program, including, for purposes of clarity any new file in Source Code form that contains any contents of the Program. Modified Works shall not include works that contain only declarations, interfaces, types, classes, structures, or files of the Program solely in each case in order to link to, bind by name, or subclass the Program or Modified Works thereof.

Distribute means the acts of **a)** distributing or **b)** making available in any manner that enables the transfer of a copy.

Source Code means the form of a Program preferred for making modifications, including but not limited to software source code, documentation source, and configuration files.

Secondary License means either the GNU General Public License, Version 2.0, or any later versions of that license, including any exceptions or additional permissions as identified by the initial Contributor.

2. Grant of Rights

a) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, Distribute and sublicense the Contribution of such Contributor, if any, and such Derivative Works.

b) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in Source Code or other form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder.

c) Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to Distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.

d) Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement.

e) Notwithstanding the terms of any Secondary License, no Contributor makes additional grants to any Recipient (other than those set forth in this Agreement) as a result of such Recipient's receipt of the Program under the terms of a Secondary License (if permitted under the terms of Section 3).

3. Requirements

3.1 If a Contributor Distributes the Program in any form, then:

* **a)** the Program must also be made available as Source Code, in accordance with section 3.2, and the Contributor must accompany the Program with a statement that the Source Code for the Program is available under this Agreement, and informs Recipients how to obtain it in a reasonable manner on or through a medium customarily used for software exchange; and

* **b)** the Contributor may Distribute the Program under a license different than this Agreement, provided that such license:
 * **i)** effectively disclaims on behalf of all other Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose;
 * **ii)** effectively excludes on behalf of all other Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits;
 * **iii)** does not attempt to limit or alter the recipients' rights in the Source Code under section 3.2; and
 * **iv)** requires any subsequent distribution of the Program by any party to be under a license that satisfies the requirements of this section 3.

3.2 When the Program is Distributed as Source Code:

* **a)** it must be made available under this Agreement, or if the Program **(i)** is combined with other material in a separate file or files made available under a Secondary License, and **(ii)** the initial Contributor attached to the Source Code the notice described in Exhibit A of this Agreement, then the Program may be made available under the terms of such Secondary Licenses, and
* **b)** a copy of this Agreement must be included with each copy of the Program.

3.3 Contributors may not remove or alter any copyright, patent, trademark, attribution notices, disclaimers of warranty, or limitations of liability (notices) contained within the Program from any copy of the Program which they Distribute, provided that Contributors may add their own appropriate notices.

4. Commercial Distribution

Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor (Commercial Contributor) hereby agrees to defend and indemnify every other Contributor (Indemnified Contributor) against any losses, damages and costs (collectively Losses) arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must: **a)** promptly

notify the Commercial Contributor in writing of such claim, and **b)** allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those damages.

5. No Warranty

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, AND TO THE EXTENT PERMITTED BY APPLICABLE LAW, THE PROGRAM IS PROVIDED ON AN AS IS BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

6. Disclaimer of Liability

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, AND TO THE EXTENT PERMITTED BY APPLICABLE LAW, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. General

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this

Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. The Eclipse Foundation is the initial Agreement Steward. The Eclipse Foundation may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be Distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to Distribute the Program (including its Contributions) under the new version.

Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved. Nothing in this Agreement is intended to be enforceable by any entity that is not a Contributor or Recipient. No third-party beneficiary rights are created under this Agreement.

Exhibit A - Form of Secondary Licenses Notice

> This Source Code may also be made available under the following Secondary Licenses when the conditions for such availability set forth in the Eclipse Public License, v. 2.0 are satisfied: {name license(s), version(s), and exceptions or additional permissions here}.

Simply including a copy of this Agreement, including this Exhibit A is not sufficient to license the Source Code under Secondary Licenses.

If it is not possible or desirable to put the notice in a particular file, then You may include the notice in a location (such as a LICENSE file in a relevant directory) where a recipient would be likely to look for such a notice.

You may add additional accurate notices of copyright ownership.
Open Source Licenses
==========

This product may include a number of subcomponents with separate
copyright notices and license terms. Your use of the source code for
these subcomponents is subject to the terms and conditions of the
subcomponent's license, as noted in the LICENSE-<subcomponent>.md
files.
[[contributors]]
== Contributors

Browse the {junit5-repo}/graphs/contributors[current list of contributors] directly on GitHub.

1.42 jackson-datatype-joda 2.14.0

1.42.1 Available under license :

This copy of Jackson JSON processor streaming parser/generator is licensed under the
Apache (Software) License, version 2.0 ("the License").
See the License for details about distribution rights, and the
specific rights regarding derivate works.

You may obtain a copy of the License at:

http://www.apache.org/licenses/LICENSE-2.0

1.43 jackson-core 2.14.0

1.43.1 Available under license :

Jackson JSON processor

Jackson is a high-performance, Free/Open Source JSON processing library.
It was originally written by Tatu Saloranta (tatu.saloranta@iki.fi), and has
been in development since 2007.
It is currently developed by a community of developers.

Licensing

Jackson 2.x core and extension components are licensed under Apache License 2.0
To find the details that apply to this artifact see the accompanying LICENSE file.

Credits

A list of contributors may be found from CREDITS(-2.x) file, which is included
in some artifacts (usually source distributions); but is always available
from the source code management (SCM) system project uses.

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common

control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object
form, that is based on (or derived from) the Work and for which the
editorial revisions, annotations, elaborations, or other modifications
represent, as a whole, an original work of authorship. For the purposes
of this License, Derivative Works shall not include works that remain
separable from, or merely link (or bind by name) to the interfaces of,
the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity
on behalf of whom a Contribution has been received by Licensor and
subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or

documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill,

work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.44 jackson-datatype-guava 2.14.0

1.44.1 Available under license :

This copy of Jackson JSON processor `jackson-datatype-guava` module is licensed under the
Apache (Software) License, version 2.0 ("the License").
See the License for details about distribution rights, and the
specific rights regarding derivate works.

You may obtain a copy of the License at:

http://www.apache.org/licenses/LICENSE-2.0

1.45 kafka-avro-serializer 5.3.1

1.45.1 Available under license :

No license file was found, but licenses were detected in source scan.

```
<project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns="http://maven.apache.org/POM/4.0.0"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <parent>
    <groupId>io.confluent</groupId>
    <artifactId>kafka-schema-registry-parent</artifactId>
    <version>5.3.1</version>
  </parent>

  <licenses>
    <license>
      <name>Apache License 2.0</name>
      <url>http://www.apache.org/licenses/LICENSE-2.0.html</url>
      <distribution>repo</distribution>
    </license>
  </licenses>

  <artifactId>kafka-avro-serializer</artifactId>
  <packaging>jar</packaging>
  <name>kafka-avro-serializer</name>

  <dependencies>
    <dependency>
      <groupId>org.apache.kafka</groupId>
      <artifactId>kafka_${kafka.scala.version}</artifactId>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>org.apache.avro</groupId>
```

```
      <artifactId>avro</artifactId>
    </dependency>
    <dependency>
      <groupId>io.confluent</groupId>
      <artifactId>kafka-schema-registry-client</artifactId>
    </dependency>
    <dependency>
      <groupId>io.confluent</groupId>
      <artifactId>common-config</artifactId>
    </dependency>

    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.avro</groupId>
        <artifactId>avro-maven-plugin</artifactId>
        <executions>
          <execution>
            <phase>generate-sources</phase>
            <goals>
              <goal>schema</goal>
            </goals>
            <configuration>
              <testSourceDirectory>${project.basedir}/src/test/avro</testSourceDirectory>
            </configuration>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
</project>
```

Found in path(s):
* /opt/cola/permits/1473561905_1668675253.166193/0/kafka-avro-serializer-5-3-1-jar/META-INF/maven/io.confluent/kafka-avro-serializer/pom.xml

1.46 mockito 2.27.0

1.46.1 Available under license :

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity
on behalf of whom a Contribution has been received by Licensor and
subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
copyright license to reproduce, prepare Derivative Works of,
publicly display, publicly perform, sublicense, and distribute the
Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
(except as stated in this section) patent license to make, have made,
use, offer to sell, sell, import, and otherwise transfer the Work,
where such license applies only to those patent claims licensable
by such Contributor that are necessarily infringed by their
Contribution(s) alone or by combination of their Contribution(s)
with the Work to which such Contribution(s) was submitted. If You
institute patent litigation against any entity (including a
cross-claim or counterclaim in a lawsuit) alleging that the Work
or a Contribution incorporated within the Work constitutes direct
or contributory patent infringement, then any patent licenses
granted to You under this License for that Work shall terminate
as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
Work or Derivative Works thereof in any medium, with or without
modifications, and in Source or Object form, provided that You
meet the following conditions:

(a) You must give any other recipients of the Work or
Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or

agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "{}" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright {yyyy} {name of copyright owner}

Licensed under the Apache License, Version 2.0 (the "License");

you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
====
Copyright (c) 2016 Mockito contributors
This program is made available under the terms of the MIT License.
====

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You

institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
   any Contribution intentionally submitted for inclusion in the Work
   by You to the Licensor shall be under the terms and conditions of
   this License, without any additional terms or conditions.
   Notwithstanding the above, nothing herein shall supersede or modify
   the terms of any separate license agreement you may have executed
   with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

The MIT License

Copyright (c) 2007 Mockito contributors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.47 lz4-and-xxhash 1.6.0

1.47.1 Available under license :

No license file was found, but licenses were detected in source scan.

/*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*   http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/lz4/LZ4Decompressor.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/xxhash/AbstractStreamingXXHash64Java.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/util/UnsafeUtils.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/xxhash/XXHash64JNI.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/xxhash/XXHashJNI.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/lz4/LZ4DecompressorWithLength.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/xxhash/XXHashFactory.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/xxhash/StreamingXXHash32JNI.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/lz4/LZ4BlockInputStream.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/util/Native.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/xxhash/XXHash64.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/lz4/LZ4Constants.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/lz4/LZ4FrameInputStream.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/lz4/LZ4SafeUtils.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/xxhash/AbstractStreamingXXHash32Java.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/lz4/LZ4CompressorWithLength.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-

jar/net/jpountz/lz4/LZ4UnknownSizeDecompressor.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/lz4/LZ4Factory.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-
jar/net/jpountz/lz4/LZ4JNISafeDecompressor.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-
jar/net/jpountz/lz4/LZ4ByteBufferUtils.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-
jar/net/jpountz/xxhash/XXHashConstants.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/util/SafeUtils.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-
jar/net/jpountz/lz4/LZ4HCJNICompressor.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-
jar/net/jpountz/xxhash/StreamingXXHash64JNI.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-
jar/net/jpountz/lz4/LZ4JNICompressor.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-
jar/net/jpountz/lz4/LZ4SafeDecompressor.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/lz4/LZ4UnsafeUtils.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-
jar/net/jpountz/xxhash/StreamingXXHash32.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/xxhash/XXHash32.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-
jar/net/jpountz/lz4/LZ4FastDecompressor.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/lz4/LZ4Utils.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/lz4/LZ4Compressor.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-
jar/net/jpountz/lz4/LZ4FrameOutputStream.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/util/Utils.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-
jar/net/jpountz/xxhash/XXHash32JNI.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/lz4/LZ4JNI.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/lz4/LZ4Exception.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-
jar/net/jpountz/lz4/LZ4BlockOutputStream.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-
jar/net/jpountz/xxhash/StreamingXXHash64.java
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-
jar/net/jpountz/lz4/LZ4JNIFastDecompressor.java
No license file was found, but licenses were detected in source scan.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE
2.0

Unless required by applicable law or agreed to in writing, software

distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

Found in path(s):
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/lz4/package.html
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/xxhash/package.html
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/overview.html
* /opt/cola/permits/1130187341_1612482199.27/0/lz4-java-1-6-0-sources-jar/net/jpountz/util/package.html

1.48 caffeine-cache 2.8.0

1.48.1 Available under license :

The trace files are copyrighted by "headissue GmbH, Jens Wilke" and provided under the CC BY 4.0 license.

File: orm-busy.trace.xz
Description: Database object access of a e-commerce web application during a busy daytime.

File: orm-night.trace.xz
Description: Database object access of a e-ommerce web application during the night time.

File: web07.trace.xz
Description: Normalized access trace (HTTP requests) a product detail page in July 2013.

File: web12.trace.xz
Description: Normalized access trace (HTTP requests) on a product detail page in December 2013.

Format: The accessed objects comprise of a mixture of product inventory, availability per price and also customer data. Objects are keyed by type, id and a index (e.g. the 3rd price of a product). All data is normalized into numbers starting at 0 (or 1 for sub-ids) and then collapsed into a single integer consisting of,
- Bits 27-31: type
- Bits 9-26: id
- Bits 0-9: index

Attribution 4.0 International

===

Creative Commons Corporation ("Creative Commons") is not a law firm and
does not provide legal services or legal advice. Distribution of
Creative Commons public licenses does not create a lawyer-client or
other relationship. Creative Commons makes its licenses and related
information available on an "as-is" basis. Creative Commons gives no

warranties regarding its licenses, any material licensed under their terms and conditions, or any related information. Creative Commons disclaims all liability for damages resulting from their use to the fullest extent possible.

Using Creative Commons Public Licenses

Creative Commons public licenses provide a standard set of terms and conditions that creators and other rights holders may use to share original works of authorship and other material subject to copyright and certain other rights specified in the public license below. The following considerations are for informational purposes only, are not exhaustive, and do not form part of our licenses.

Considerations for licensors: Our public licenses are intended for use by those authorized to give the public permission to use material in ways otherwise restricted by copyright and certain other rights. Our licenses are irrevocable. Licensors should read and understand the terms and conditions of the license they choose before applying it. Licensors should also secure all rights necessary before applying our licenses so that the public can reuse the material as expected. Licensors should clearly mark any material not subject to the license. This includes other CC-licensed material, or material used under an exception or limitation to copyright. More considerations for licensors: wiki.creativecommons.org/Considerations_for_licensors

Considerations for the public: By using one of our public licenses, a licensor grants the public permission to use the licensed material under specified terms and conditions. If the licensor's permission is not necessary for any reason--for example, because of any applicable exception or limitation to copyright--then that use is not regulated by the license. Our licenses grant only permissions under copyright and certain other rights that a licensor has authority to grant. Use of the licensed material may still be restricted for other reasons, including because others have copyright or other rights in the material. A licensor may make special requests, such as asking that all changes be marked or described. Although not required by our licenses, you are encouraged to respect those requests where reasonable. More_considerations for the public: wiki.creativecommons.org/Considerations_for_licensees

==========

Creative Commons Attribution 4.0 International Public License

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

Section 1 -- Definitions.

a. Adapted Material means material subject to Copyright and Similar Rights that is derived from or based upon the Licensed Material and in which the Licensed Material is translated, altered, arranged, transformed, or otherwise modified in a manner requiring permission under the Copyright and Similar Rights held by the Licensor. For purposes of this Public License, where the Licensed Material is a musical work, performance, or sound recording, Adapted Material is always produced where the Licensed Material is synched in timed relation with a moving image.

b. Adapter's License means the license You apply to Your Copyright and Similar Rights in Your contributions to Adapted Material in accordance with the terms and conditions of this Public License.

c. Copyright and Similar Rights means copyright and/or similar rights closely related to copyright including, without limitation, performance, broadcast, sound recording, and Sui Generis Database Rights, without regard to how the rights are labeled or categorized. For purposes of this Public License, the rights specified in Section 2(b)(1)-(2) are not Copyright and Similar Rights.

d. Effective Technological Measures means those measures that, in the absence of proper authority, may not be circumvented under laws fulfilling obligations under Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996, and/or similar international agreements.

e. Exceptions and Limitations means fair use, fair dealing, and/or any other exception or limitation to Copyright and Similar Rights that applies to Your use of the Licensed Material.

f. Licensed Material means the artistic or literary work, database, or other material to which the Licensor applied this Public License.

g. Licensed Rights means the rights granted to You subject to the terms and conditions of this Public License, which are limited to all Copyright and Similar Rights that apply to Your use of the Licensed Material and that the Licensor has authority to license.

h. Licensor means the individual(s) or entity(ies) granting rights under this Public License.

i. Share means to provide material to the public by any means or process that requires permission under the Licensed Rights, such as reproduction, public display, public performance, distribution, dissemination, communication, or importation, and to make material available to the public including in ways that members of the public may access the material from a place and at a time individually chosen by them.

j. Sui Generis Database Rights means rights other than copyright resulting from Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, as amended and/or succeeded, as well as other essentially equivalent rights anywhere in the world.

k. You means the individual or entity exercising the Licensed Rights under this Public License. Your has a corresponding meaning.

Section 2 -- Scope.

a. License grant.

1. Subject to the terms and conditions of this Public License, the Licensor hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive, irrevocable license to exercise the Licensed Rights in the Licensed Material to:

a. reproduce and Share the Licensed Material, in whole or in part; and

b. produce, reproduce, and Share Adapted Material.

2. Exceptions and Limitations. For the avoidance of doubt, where Exceptions and Limitations apply to Your use, this Public License does not apply, and You do not need to comply with its terms and conditions.

3. Term. The term of this Public License is specified in Section 6(a).

4. Media and formats; technical modifications allowed. The Licensor authorizes You to exercise the Licensed Rights in all media and formats whether now known or hereafter created, and to make technical modifications necessary to do so. The Licensor waives and/or agrees not to assert any right or authority to forbid You from making technical modifications necessary to exercise the Licensed Rights, including technical modifications necessary to circumvent Effective Technological Measures. For purposes of this Public License, simply making modifications authorized by this Section 2(a)(4) never produces Adapted Material.

5. Downstream recipients.

   a. Offer from the Licensor -- Licensed Material. Every recipient of the Licensed Material automatically receives an offer from the Licensor to exercise the Licensed Rights under the terms and conditions of this Public License.

   b. No downstream restrictions. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, the Licensed Material if doing so restricts exercise of the Licensed Rights by any recipient of the Licensed Material.

6. No endorsement. Nothing in this Public License constitutes or may be construed as permission to assert or imply that You are, or that Your use of the Licensed Material is, connected with, or sponsored, endorsed, or granted official status by, the Licensor or others designated to receive attribution as provided in Section 3(a)(1)(A)(i).

b. Other rights.

1. Moral rights, such as the right of integrity, are not licensed under this Public License, nor are publicity, privacy, and/or other similar personality rights; however, to the extent possible, the Licensor waives and/or agrees not to assert any such rights held by the Licensor to the limited extent necessary to allow You to exercise the Licensed Rights, but not otherwise.

2. Patent and trademark rights are not licensed under this Public License.

3. To the extent possible, the Licensor waives any right to collect royalties from You for the exercise of the Licensed Rights, whether directly or through a collecting society under any voluntary or waivable statutory or compulsory licensing scheme. In all other cases the Licensor expressly reserves any right to collect such royalties.

Section 3 -- License Conditions.

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

a. Attribution.

1. If You Share the Licensed Material (including in modified form), You must:

a. retain the following if it is supplied by the Licensor with the Licensed Material:

i. identification of the creator(s) of the Licensed Material and any others designated to receive attribution, in any reasonable manner requested by the Licensor (including by pseudonym if designated);

ii. a copyright notice;

iii. a notice that refers to this Public License;

iv. a notice that refers to the disclaimer of warranties;

v. a URI or hyperlink to the Licensed Material to the extent reasonably practicable;

b. indicate if You modified the Licensed Material and retain an indication of any previous modifications; and

c. indicate the Licensed Material is licensed under this Public License, and include the text of, or the URI or hyperlink to, this Public License.

2. You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medium, means, and context in which You Share the Licensed Material. For example, it may be reasonable to satisfy the conditions by providing a URI or

hyperlink to a resource that includes the required information.

3. If requested by the Licensor, You must remove any of the information required by Section 3(a)(1)(A) to the extent reasonably practicable.

4. If You Share Adapted Material You produce, the Adapter's License You apply must not prevent recipients of the Adapted Material from complying with this Public License.

Section 4 -- Sui Generis Database Rights.

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed Material:

a. for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, and Share all or a substantial portion of the contents of the database;

b. if You include all or a substantial portion of the database contents in a database in which You have Sui Generis Database Rights, then the database in which You have Sui Generis Database Rights (but not its individual contents) is Adapted Material; and

c. You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of the contents of the database.

For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under this Public License where the Licensed Rights include other Copyright and Similar Rights.

Section 5 -- Disclaimer of Warranties and Limitation of Liability.

a. UNLESS OTHERWISE SEPARATELY UNDERTAKEN BY THE LICENSOR, TO THE EXTENT POSSIBLE, THE LICENSOR OFFERS THE LICENSED MATERIAL AS-IS AND AS-AVAILABLE, AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE LICENSED MATERIAL, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHER. THIS INCLUDES, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OR ABSENCE OF ERRORS, WHETHER OR NOT KNOWN OR DISCOVERABLE. WHERE DISCLAIMERS OF WARRANTIES ARE NOT ALLOWED IN FULL OR IN PART, THIS DISCLAIMER MAY NOT APPLY TO YOU.

b. TO THE EXTENT POSSIBLE, IN NO EVENT WILL THE LICENSOR BE LIABLE

TO YOU ON ANY LEGAL THEORY (INCLUDING, WITHOUT LIMITATION, NEGLIGENCE) OR OTHERWISE FOR ANY DIRECT, SPECIAL, INDIRECT, INCIDENTAL, CONSEQUENTIAL, PUNITIVE, EXEMPLARY, OR OTHER LOSSES, COSTS, EXPENSES, OR DAMAGES ARISING OUT OF THIS PUBLIC LICENSE OR USE OF THE LICENSED MATERIAL, EVEN IF THE LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH LOSSES, COSTS, EXPENSES, OR DAMAGES. WHERE A LIMITATION OF LIABILITY IS NOT ALLOWED IN FULL OR IN PART, THIS LIMITATION MAY NOT APPLY TO YOU.

c. The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

Section 6 -- Term and Termination.

a. This Public License applies for the term of the Copyright and Similar Rights licensed here. However, if You fail to comply with this Public License, then Your rights under this Public License terminate automatically.

b. Where Your right to use the Licensed Material has terminated under Section 6(a), it reinstates:

1. automatically as of the date the violation is cured, provided it is cured within 30 days of Your discovery of the violation; or

2. upon express reinstatement by the Licensor.

For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to seek remedies for Your violations of this Public License.

c. For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate terms or conditions or stop distributing the Licensed Material at any time; however, doing so will not terminate this Public License.

d. Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

Section 7 -- Other Terms and Conditions.

a. The Licensor shall not be bound by any additional or different terms or conditions communicated by You unless expressly agreed.

b. Any arrangements, understandings, or agreements regarding the Licensed Material not stated herein are separate from and independent of the terms and conditions of this Public License.

Section 8 -- Interpretation.

a. For the avoidance of doubt, this Public License does not, and shall not be interpreted to, reduce, limit, restrict, or impose conditions on any use of the Licensed Material that could lawfully be made without permission under this Public License.

b. To the extent possible, if any provision of this Public License is deemed unenforceable, it shall be automatically reformed to the minimum extent necessary to make it enforceable. If the provision cannot be reformed, it shall be severed from this Public License without affecting the enforceability of the remaining terms and conditions.

c. No term or condition of this Public License will be waived and no failure to comply consented to unless expressly agreed to by the Licensor.

d. Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver of, any privileges and immunities that apply to the Licensor or You, including from the legal processes of any jurisdiction or authority.

==========

Creative Commons is not a party to its public licenses. Notwithstanding, Creative Commons may elect to apply one of its public licenses to material it publishes and in those instances will be considered the Licensor. The text of the Creative Commons public licenses is dedicated to the public domain under the CC0 Public Domain Dedication. Except for the limited purpose of indicating that material is shared under a Creative Commons public license or as otherwise permitted by the Creative Commons policies published at creativecommons.org/policies, Creative Commons does not authorize the use of the trademark "Creative Commons" or any other trademark or logo of Creative Commons without its prior written consent including, without limitation, in connection with any unauthorized modifications to any of its public licenses or any other arrangements, understandings, or agreements concerning use of licensed material. For the avoidance of doubt, this paragraph does not form part of the public licenses.

Creative Commons may be contacted at creativecommons.org.
Copyright $L Ben Manes. All Rights Reserved.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,

and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s)

with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
   any Contribution intentionally submitted for inclusion in the Work
   by You to the Licensor shall be under the terms and conditions of
   this License, without any additional terms or conditions.
   Notwithstanding the above, nothing herein shall supersede or modify
   the terms of any separate license agreement you may have executed
   with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.49 objenesis 2.6

1.49.1 Available under license :

// --
// NOTICE file corresponding to the section 4d of The Apache License,
// Version 2.0, in this case for Objenesis
// --

Objenesis
Copyright 2006-2017 Joe Walnes, Henri Tremblay, Leonardo Mesquita

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object
form, that is based on (or derived from) the Work and for which the
editorial revisions, annotations, elaborations, or other modifications
represent, as a whole, an original work of authorship. For the purposes
of this License, Derivative Works shall not include works that remain
separable from, or merely link (or bind by name) to the interfaces of,
the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but

excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its

distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise,

unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.50 metrics-utility-servlets 4.0.5

1.50.1 Available under license :

No license file was found, but licenses were detected in source scan.

Manifest-Version: 1.0
Bnd-LastModified: 1545938238563
Build-Jdk: 1.8.0_191
Built-By: artem
Bundle-Description: A set of utility servlets for Metrics, allowing yo
u to expose valuable information about your production environ
ment.
Bundle-License: http://www.apache.org/licenses/LICENSE-2.0.html
Bundle-ManifestVersion: 2
Bundle-Name: Metrics Utility Servlets
Bundle-SymbolicName: io.dropwizard.metrics.servlets
Bundle-Version: 4.0.5
Created-By: Apache Maven Bundle Plugin
Export-Package: com.codahale.metrics.servlets;uses:="com.codahale.metr
ics,com.codahale.metrics.health,javax.servlet,javax.servlet.http";ver
sion="4.0.5"
Implementation-Title: Metrics Utility Servlets
Implementation-URL: http://metrics.dropwizard.io/metrics-servlets
Implementation-Vendor-Id: io.dropwizard.metrics
Implementation-Version: 4.0.5
Import-Package: javax.servlet;version="[2.5.0,4.0.0)",javax.servlet.ht
tp;version="[2.5.0,4.0.0)",com.codahale.metrics;version="[4.0,5)",com
.codahale.metrics.health;version="[4.0,5)",com.codahale.metrics.json;
version="[4.0,5)",com.codahale.metrics.jvm;version="[4.0,5)",com.fast
erxml.jackson.databind;version="[2.9,3)",com.fasterxml.jackson.databi
nd.util;version="[2.9,3)",com.papertrail.profiler,org.joda.time;versi
on="[2.9,3)"
Require-Capability: osgi.ee;filter:="(&(osgi.ee=JavaSE)(version=1.8))"
Tool: Bnd-3.3.0.201609221906

Found in path(s):
* /opt/cola/permits/1274701310_1648835822.95/0/metrics-servlets-4-0-5-jar/META-INF/MANIFEST.MF

1.51 okio 1.17.2

1.51.1 Available under license :

No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2014 Square, Inc.

*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*   http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/RealBufferedSink.java
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/SegmentPool.java
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/Okio.java
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/Source.java
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/InflaterSource.java
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/DeflaterSink.java
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/ForwardingSink.java
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/Sink.java
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/RealBufferedSource.java
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/BufferedSource.java
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/ForwardingSource.java
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/GzipSource.java
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/Buffer.java
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/Timeout.java
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/GzipSink.java
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/Util.java
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/BufferedSink.java
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/Segment.java
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/AsyncTimeout.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2019 Square, Inc.
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*   http://www.apache.org/licenses/LICENSE-2.0
*

* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/PushableTimeout.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2018 Square, Inc.
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*   http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/PeekSource.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2017 Square, Inc.
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*   http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/Utf8.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2016 Square, Inc.
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*   http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/HashingSource.java
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/Pipe.java
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/HashingSink.java
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/Options.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2015 Square, Inc.
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*   http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/SegmentedByteString.java
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/ForwardingTimeout.java

No license file was found, but licenses were detected in source scan.

```
/*
* Copyright 2014 Square Inc.
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
```

Found in path(s):
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/ByteString.java
No license file was found, but licenses were detected in source scan.

```
/*
*  Licensed to the Apache Software Foundation (ASF) under one or more
*  contributor license agreements.  See the NOTICE file distributed with
*  this work for additional information regarding copyright ownership.
*  The ASF licenses this file to You under the Apache License, Version 2.0
*  (the "License"); you may not use this file except in compliance with
*  the License.  You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
*  Unless required by applicable law or agreed to in writing, software
*  distributed under the License is distributed on an "AS IS" BASIS,
*  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
*  See the License for the specific language governing permissions and
*  limitations under the License.
*/
```

Found in path(s):
* /opt/ws_local/PERMITS_SQL/1014221201_1591897829.68/0/okio-1-17-2-sources-jar/okio/Base64.java

1.52 python-requests 2.27.1

1.52.1 Available under license :

Requests
Copyright 2019 Kenneth Reitz

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain

separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or
Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices
stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works
that You distribute, all copyright, patent, trademark, and
attribution notices from the Source form of the Work,
excluding those notices that do not pertain to any part of
the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its
distribution, then any Derivative Works that You distribute must
include a readable copy of the attribution notices contained
within such NOTICE file, excluding those notices that do not
pertain to any part of the Derivative Works, in at least one
of the following places: within a NOTICE text file distributed
as part of the Derivative Works; within the Source form or
documentation, if provided along with the Derivative Works; or,
within a display generated by the Derivative Works, if and
wherever such third-party notices normally appear. The contents
of the NOTICE file are for informational purposes only and
do not modify the License. You may add Your own attribution
notices within Derivative Works that You distribute, alongside
or as an addendum to the NOTICE text from the Work, provided
that such additional attribution notices cannot be construed
as modifying the License.

You may add Your own copyright statement to Your modifications and
may provide additional or different license terms and conditions
for use, reproduction, or distribution of Your modifications, or
for any such Derivative Works as a whole, provided Your use,
reproduction, and distribution of the Work otherwise complies with
the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
any Contribution intentionally submitted for inclusion in the Work
by You to the Licensor shall be under the terms and conditions of
this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify
the terms of any separate license agreement you may have executed
with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
names, trademarks, service marks, or product names of the Licensor,
except as required for reasonable and customary use in describing the

origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
agreed to in writing, Licensor provides the Work (and each
Contributor provides its Contributions) on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied, including, without limitation, any warranties or conditions
of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
PARTICULAR PURPOSE. You are solely responsible for determining the
appropriateness of using or redistributing the Work and assume any
risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
whether in tort (including negligence), contract, or otherwise,
unless required by applicable law (such as deliberate and grossly
negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special,
incidental, or consequential damages of any character arising as a
result of this License or out of the use or inability to use the
Work (including but not limited to damages for loss of goodwill,
work stoppage, computer failure or malfunction, or any and all
other commercial damages or losses), even if such Contributor
has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
the Work or Derivative Works thereof, You may choose to offer,
and charge a fee for, acceptance of support, warranty, indemnity,
or other liability obligations and/or rights consistent with this
License. However, in accepting such obligations, You may act only
on Your own behalf and on Your sole responsibility, not on behalf
of any other Contributor, and only if You agree to indemnify,
defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

1.53 apache-yetus-audience-annotations 0.12.0

1.53.1 Available under license :

Apache Yetus
Copyright 2008-2020 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (https://www.apache.org/).

Additional licenses for the Apache Yetus Source/Website:

See LICENSE for terms.
Licensed to the Apache Software Foundation (ASF) under one or more
contributor license agreements. See the NOTICE file distributed with
this work for additional information regarding copyright ownership.
The ASF licenses this file to You under the Apache License, Version 2.0
(the "License"); you may not use this file except in compliance with
the License. You may obtain a copy of the License at
#
http://www.apache.org/licenses/LICENSE-2.0
#
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made,

use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions

for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability

incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following
boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

Additional licenses for the Apache Yetus Source/Website:

This project incorporates portions of the Bootstrap project available
under the MIT license:

The MIT License (MIT)

Copyright (c) 2011-2015 Twitter, Inc

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

This project incorporates NORMALIZE.css as bundled with the Twitter Bootstrap
project which is released under the same license as Bootstrap.

Copyright Nicolas Gallagher and Jonathan Neal

This project incorporates GLYPHICONS FREE as bundled with the Twitter Bootstrap
project which are released under the same license as Bootstrap.

Copyright (c) 2010 - 2015 Jan Kovarik

This project incorporates portions of the Font Awesome project available
under the MIT license and SIL OFL 1.1 .

Copyright (c) 2015 Dave Gandy

The MIT License (MIT)

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER

LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

SIL OPEN FONT LICENSE Version 1.1 - 26 February 2007

PREAMBLE

The goals of the Open Font License (OFL) are to stimulate worldwide development of collaborative font projects, to support the font creation efforts of academic and linguistic communities, and to provide a free and open framework in which fonts may be shared and improved in partnership with others.

The OFL allows the licensed fonts to be used, studied, modified and redistributed freely as long as they are not sold by themselves. The fonts, including any derivative works, can be bundled, embedded, redistributed and/or sold with any software provided that any reserved names are not used by derivative works. The fonts and derivatives, however, cannot be released under any other type of license. The requirement for fonts to remain under this license does not apply to any document created using the fonts or their derivatives.

DEFINITIONS

"Font Software" refers to the set of files released by the Copyright Holder(s) under this license and clearly marked as such. This may include source files, build scripts and documentation.

"Reserved Font Name" refers to any names specified as such after the copyright statement(s).

"Original Version" refers to the collection of Font Software components as distributed by the Copyright Holder(s).

"Modified Version" refers to any derivative made by adding to, deleting, or substituting -- in part or in whole -- any of the components of the Original Version, by changing formats or by porting the Font Software to a new environment.

"Author" refers to any designer, engineer, programmer, technical writer or other person who contributed to the Font Software.

PERMISSION & CONDITIONS

Permission is hereby granted, free of charge, to any person obtaining a copy of the Font Software, to use, study, copy, merge, embed, modify, redistribute, and sell modified and unmodified copies of the Font Software, subject to the following conditions:

1) Neither the Font Software nor any of its individual components,
in Original or Modified Versions, may be sold by itself.

2) Original or Modified Versions of the Font Software may be bundled,
redistributed and/or sold with any software, provided that each copy
contains the above copyright notice and this license. These can be
included either as stand-alone text files, human-readable headers or
in the appropriate machine-readable metadata fields within text or
binary files as long as those fields can be easily viewed by the user.

3) No Modified Version of the Font Software may use the Reserved Font
Name(s) unless explicit written permission is granted by the corresponding
Copyright Holder. This restriction only applies to the primary font name as
presented to the users.

4) The name(s) of the Copyright Holder(s) or the Author(s) of the Font
Software shall not be used to promote, endorse or advertise any
Modified Version, except to acknowledge the contribution(s) of the
Copyright Holder(s) and the Author(s) or with their explicit written
permission.

5) The Font Software, modified or unmodified, in part or in whole,
must be distributed entirely under this license, and must not be
distributed under any other license. The requirement for fonts to
remain under this license does not apply to any document created
using the Font Software.

TERMINATION
This license becomes null and void if any of the above conditions are
not met.

DISCLAIMER
THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT
OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL THE
COPYRIGHT HOLDER BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL
DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM
OTHER DEALINGS IN THE FONT SOFTWARE.

This project incorporates portions of the JQuery project available under the
MIT license:

Copyright jQuery Foundation and other contributors, https://jquery.org/

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

This project incorporates via jQuery portions of the Sizzle project
available under the MIT license:

Copyright JS Foundation and other contributors, https://js.foundation/

This software consists of voluntary contributions made by many
individuals. For exact contribution history, see the revision history
available at https://github.com/jquery/sizzle

The following license applies to all parts of this software except as
documented below:

====

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

This project utilizes Jython 2.7 for running Python code on JVMs. It is available under the Python Software Foundation License v2:

PYTHON SOFTWARE FOUNDATION LICENSE VERSION 2
---

1. This LICENSE AGREEMENT is between the Python Software Foundation ("PSF"), and the Individual or Organization ("Licensee") accessing and otherwise using this software ("Jython") in source or binary form and its associated documentation.

2. Subject to the terms and conditions of this License Agreement, PSF hereby grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use Jython alone or in any derivative version, provided, however, that PSF's License Agreement and PSF's notice of copyright, i.e., "Copyright (c) 2007 Python Software Foundation; All Rights Reserved" are retained in Jython alone or in any derivative version prepared by Licensee.

3. In the event Licensee prepares a derivative work that is based on or incorporates Jython or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to Jython.

4. PSF is making Jython available to Licensee on an "AS IS" basis. PSF MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, PSF MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF JYTHON WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF JYTHON FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING JYTHON, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.

7. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between PSF and Licensee. This License Agreement does not grant permission to use PSF trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.

8. By copying, installing or otherwise using Jython, Licensee agrees to be bound by the terms and conditions of this License Agreement.

1.54 bean-validation-api 2.0.1

1.54.1 Available under license :

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but

not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object
form, that is based on (or derived from) the Work and for which the
editorial revisions, annotations, elaborations, or other modifications
represent, as a whole, an original work of authorship. For the purposes
of this License, Derivative Works shall not include works that remain
separable from, or merely link (or bind by name) to the interfaces of,
the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity
on behalf of whom a Contribution has been received by Licensor and
subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
copyright license to reproduce, prepare Derivative Works of,
publicly display, publicly perform, sublicense, and distribute the
Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
(except as stated in this section) patent license to make, have made,
use, offer to sell, sell, import, and otherwise transfer the Work,
where such license applies only to those patent claims licensable
by such Contributor that are necessarily infringed by their

Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with

the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
   any Contribution intentionally submitted for inclusion in the Work
   by You to the Licensor shall be under the terms and conditions of
   this License, without any additional terms or conditions.
   Notwithstanding the above, nothing herein shall supersede or modify
   the terms of any separate license agreement you may have executed
   with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright 2013 Cognifide Limited

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.55 zookeeper 3.8.0

1.55.1 Available under license :

No license file was found, but licenses were detected in source scan.

<!--

Licensed to the Apache Software Foundation (ASF) under one or more
contributor license agreements. See the NOTICE file distributed with
this work for additional information regarding copyright ownership.
The ASF licenses this file to You under the Apache License, Version 2.0
(the "License"); you may not use this file except in compliance with
the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and

limitations under the License.

-->

Found in path(s):
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/META-INF/maven/org.apache.camel/camel-zookeeper/pom.xml
No license file was found, but licenses were detected in source scan.

/*
* Licensed to the Apache Software Foundation (ASF) under one or more
* contributor license agreements. See the NOTICE file distributed with
* this work for additional information regarding copyright ownership.
* The ASF licenses this file to You under the Apache License, Version 2.0
* (the "License"); you may not use this file except in compliance with
* the License. You may obtain a copy of the License at
*
*   http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/ZooKeeperMessage.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/NaturalSortComparator.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/ZooKeeperUtils.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/ZooKeeperEndpoint.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/cloud/ZooKeeperServiceDiscoveryFactory.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/ZooKeeperCuratorHelper.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/ZooKeeperConnectionManager.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/ZooKeeperCuratorConfiguration.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/cluster/ZooKeeperClusterView.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/operations/AnyOfOperations.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-

jar/org/apache/camel/component/zookeeper/operations/CreateOperation.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/ZooKeeperConsumer.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/cloud/ZooKeeperServiceRegistryConfiguration.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/operations/GetDataOperation.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/cloud/ZooKeeperServiceDiscovery.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/operations/DataChangedOperation.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/operations/SetDataOperation.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/operations/DeleteOperation.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/operations/ChildrenChangedOperation.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/cloud/ZooKeeperServiceRegistry.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/operations/OperationResult.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/operations/ZooKeeperOperation.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/operations/ExistenceChangedOperation.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/ZooKeeperProducer.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/operations/ZooKeeperHelper.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/SequenceComparator.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/ZooKeeperConfiguration.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/cluster/ZooKeeperClusterService.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/operations/FutureEventDrivenOperation.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/ZooKeeperComponent.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/operations/ExistsOperation.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/operations/GetChildrenOperation.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/ConnectionHolder.java
* /opt/cola/permits/1425206646_1663952692.6053333/0/camel-zookeeper-3-8-0-sources-2-jar/org/apache/camel/component/zookeeper/operations/WatchedEventProvider.java

1.56 jackson-integration-for-metrics 4.0.5

1.56.1 Available under license :

No license file was found, but licenses were detected in source scan.

Manifest-Version: 1.0
Bnd-LastModified: 1545938158164
Build-Jdk: 1.8.0_191
Built-By: artem
Bundle-Description: A set of Jackson modules which provide serializers
 for most Metrics classes.
Bundle-License: http://www.apache.org/licenses/LICENSE-2.0.html
Bundle-ManifestVersion: 2
Bundle-Name: Jackson Integration for Metrics
Bundle-SymbolicName: io.dropwizard.metrics.json
Bundle-Version: 4.0.5
Created-By: Apache Maven Bundle Plugin
Export-Package: com.codahale.metrics.json;uses:="com.codahale.metrics,
com.fasterxml.jackson.core,com.fasterxml.jackson.databind";version="4
.0.5"
Implementation-Title: Jackson Integration for Metrics
Implementation-URL: http://metrics.dropwizard.io/metrics-json
Implementation-Vendor-Id: io.dropwizard.metrics
Implementation-Version: 4.0.5
Import-Package: com.codahale.metrics;version="[4.0,5)",com.codahale.me
trics.health;version="[4.0,5)";resolution:=optional,com.fasterxml.jac
kson.core;version="[2.9,3)",com.fasterxml.jackson.databind;version="[
2.9,3)",com.fasterxml.jackson.databind.module;version="[2.9,3)",com.f
asterxml.jackson.databind.ser;version="[2.9,3)",com.fasterxml.jackson
.databind.ser.std;version="[2.9,3)"
Require-Capability: osgi.ee;filter:="(&(osgi.ee=JavaSE)(version=1.8))"
Tool: Bnd-3.3.0.201609221906

Found in path(s):
* /opt/cola/permits/1274705522_1648836004.08/0/metrics-json-4-0-5-jar/META-INF/MANIFEST.MF

1.57 logback-core 1.2.10

1.57.1 Available under license :

Found license 'Eclipse Public License 1.0' in '* Copyright (C) 1999-2015, QOS.ch. All rights reserved. * This program and the accompanying materials are dual-licensed under * either the terms of the Eclipse Public License v1.0 as published by * under the terms of the GNU Lesser General Public License version 2.1 * as published by the Free Software Foundation.'
Found license 'GNU Lesser General Public License' in '* Copyright (C) 1999-2015, QOS.ch. All rights reserved. * This program and the accompanying materials are dual-licensed under * either the terms of the Eclipse Public

License v1.0 as published by * under the terms of the GNU Lesser General Public License version 2.1 * as published by the Free Software Foundation.'

1.58 python-certifi 2021.10.8

1.58.1 Available under license :

This package contains a modified version of ca-bundle.crt:

ca-bundle.crt -- Bundle of CA Root Certificates

Certificate data from Mozilla as of: Thu Nov 3 19:04:19 2011#
This is a bundle of X.509 certificates of public Certificate Authorities
(CA). These were automatically extracted from Mozilla's root certificates
file (certdata.txt). This file can be found in the mozilla source tree:
http://mxr.mozilla.org/mozilla/source/security/nss/lib/ckfw/builtins/certdata.txt?raw=1#
It contains the certificates in PEM format and therefore
can be directly used with curl / libcurl / php_curl, or with
an Apache+mod_ssl webserver for SSL client authentication.
Just configure this file as the SSLCACertificateFile.#

***** BEGIN LICENSE BLOCK *****
This Source Code Form is subject to the terms of the Mozilla Public License,
v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain
one at http://mozilla.org/MPL/2.0/.

***** END LICENSE BLOCK *****
@(#) $RCSfile: certdata.txt,v $ $Revision: 1.80 $ $Date: 2011/11/03 15:11:58 $

1.59 jacoco v0.8.3

1.59.1 Available under license :

License
=======

Copyright (c) 2009, 2019 Mountainminds GmbH & Co. KG and Contributors

The JaCoCo Java Code Coverage Library and all included documentation is made
available by Mountainminds GmbH & Co. KG, Munich. Except indicated below, the
Content is provided to you under the terms and conditions of the Eclipse Public
License Version 1.0 ("EPL"). A copy of the EPL is available at
http://www.eclipse.org/legal/epl-v10.html.

Please visit
http://www.jacoco.org/jacoco/trunk/doc/license.html
for the complete license information including third party licenses and trademarks.

1.60 metrics-core 4.0.5

1.60.1 Available under license :

Apache License

Version 2.0, January 2004

http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic

mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

You must give any other recipients of the Work or Derivative Works a copy of this License; and
You must cause any modified files to carry prominent notices stating that You changed the files; and
You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.
5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of

any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: HOW TO APPLY THE APACHE LICENSE TO YOUR WORK
To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and
limitations under the License.

1.61 jackson-module-afterburner 2.14.0

1.61.1 Available under license :

Jackson JSON processor

Jackson is a high-performance, Free/Open Source JSON processing library.
It was originally written by Tatu Saloranta (tatu.saloranta@iki.fi), and has
been in development since 2007.
It is currently developed by a community of developers, as well as supported
commercially by FasterXML.com.

Licensing

Jackson core and extension components (as well their dependencies) may be licensed under
different licenses.
To find the details that apply to this artifact see the accompanying LICENSE file.
For more information, including possible other licensing options, contact
FasterXML.com (http://fasterxml.com).

Credits

A list of contributors may be found from CREDITS file, which is included
in some artifacts (usually source distributions); but is always available
from the source code management (SCM) system project uses.
This copy of Jackson JSON processor `jackson-module-afterburner` module is licensed under the
Apache (Software) License, version 2.0 ("the License").
See the License for details about distribution rights, and the
specific rights regarding derivate works.

You may obtain a copy of the License at:

  http://www.apache.org/licenses/LICENSE-2.0

Additional licensing information exists for following 3rd party library dependencies

ASM

ASM: a very small and fast Java bytecode manipulation framework
Copyright (c) 2000-2011 INRIA, France Telecom
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holders nor the names of its
contributors may be used to endorse or promote products derived from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
THE POSSIBILITY OF SUCH DAMAGE.

1.62 junit-5-bill-of-materials 5.7.2

1.62.1 Available under license :

import java.io.File
import java.net.URI

data class License(val name: String, val url: URI, val headerFile: File)
Apache License
==========

Version 2.0, January 2004
<<https://www.apache.org/licenses/>>

Terms and Conditions for use, reproduction, and distribution

1. Definitions

License shall mean the terms and conditions for use, reproduction, and
distribution as defined by Sections 1 through 9 of this document.

Licensor shall mean the copyright owner or entity authorized by the copyright
owner that is granting the License.

Legal Entity shall mean the union of the acting entity and all other entities
that control, are controlled by, or are under common control with that entity.

For the purposes of this definition, control means **(i)** the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or **(ii)** ownership of fifty percent (50%) or more of the outstanding shares, or **(iii)** beneficial ownership of such entity.

You (or Your) shall mean an individual or Legal Entity exercising permissions granted by this License.

Source form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

Object form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

Work shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

Derivative Works shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

Contribution shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, submitted means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as Not a Contribution.

Contributor shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free,

irrevocable copyright license to reproduce, prepare Derivative Works of,
publicly display, publicly perform, sublicense, and distribute the Work and such
Derivative Works in Source or Object form.

3. Grant of Patent License

Subject to the terms and conditions of this License, each Contributor hereby
grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free,
irrevocable (except as stated in this section) patent license to make, have
made, use, offer to sell, sell, import, and otherwise transfer the Work, where
such license applies only to those patent claims licensable by such Contributor
that are necessarily infringed by their Contribution(s) alone or by combination
of their Contribution(s) with the Work to which such Contribution(s) was
submitted. If You institute patent litigation against any entity (including a
cross-claim or counterclaim in a lawsuit) alleging that the Work or a
Contribution incorporated within the Work constitutes direct or contributory
patent infringement, then any patent licenses granted to You under this License
for that Work shall terminate as of the date such litigation is filed.

4. Redistribution

You may reproduce and distribute copies of the Work or Derivative Works thereof
in any medium, with or without modifications, and in Source or Object form,
provided that You meet the following conditions:

* **(a)** You must give any other recipients of the Work or Derivative Works a copy of
this License; and
* **(b)** You must cause any modified files to carry prominent notices stating that You
changed the files; and
* **(c)** You must retain, in the Source form of any Derivative Works that You distribute,
all copyright, patent, trademark, and attribution notices from the Source form
of the Work, excluding those notices that do not pertain to any part of the
Derivative Works; and
* **(d)** If the Work includes a NOTICE text file as part of its distribution, then any
Derivative Works that You distribute must include a readable copy of the
attribution notices contained within such NOTICE file, excluding those notices
that do not pertain to any part of the Derivative Works, in at least one of the
following places: within a NOTICE text file distributed as part of the
Derivative Works; within the Source form or documentation, if provided along
with the Derivative Works; or, within a display generated by the Derivative
Works, if and wherever such third-party notices normally appear. The contents of
the NOTICE file are for informational purposes only and do not modify the
License. You may add Your own attribution notices within Derivative Works that
You distribute, alongside or as an addendum to the NOTICE text from the Work,
provided that such additional attribution notices cannot be construed as
modifying the License.

You may add Your own copyright statement to Your modifications and may provide

additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an AS IS BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or

other liability obligations and/or rights consistent with this License. However,
in accepting such obligations, You may act only on Your own behalf and on Your
sole responsibility, not on behalf of any other Contributor, and only if You
agree to indemnify, defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason of your
accepting any such warranty or additional liability.
Eclipse Public License - v 2.0
==========

THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS ECLIPSE PUBLIC LICENSE (AGREEMENT). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.

1. Definitions

Contribution means:
* **a)** in the case of the initial Contributor, the initial content Distributed under this Agreement, and
* **b)** in the case of each subsequent Contributor:
 * **i)** changes to the Program, and
 * **ii)** additions to the Program;
where such changes and/or additions to the Program originate from and are Distributed by that particular Contributor. A Contribution originates from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor's behalf. Contributions do not include changes or additions to the Program that are not Modified Works.

Contributor means any person or entity that Distributes the Program.

Licensed Patents mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program.

Program means the Contributions Distributed in accordance with this Agreement.

Recipient means anyone who receives the Program under this Agreement or any Secondary License (as applicable), including Contributors.

Derivative Works shall mean any work, whether in Source Code or other form, that is based on (or derived from) the Program and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship.

Modified Works shall mean any work in Source Code or other form that results from an addition to, deletion from, or modification of the contents of the Program, including, for purposes of clarity any new file in Source Code form that contains any contents of the Program. Modified Works shall not include works that contain only declarations, interfaces, types, classes, structures, or files of the Program solely in each case in order to link to, bind by name, or subclass the Program or Modified Works thereof.

Distribute means the acts of **a)** distributing or **b)** making available in any manner that enables the transfer of a copy.

Source Code means the form of a Program preferred for making modifications, including but not limited to software source code, documentation source, and configuration files.

Secondary License means either the GNU General Public License, Version 2.0, or any later versions of that license, including any exceptions or additional permissions as identified by the initial Contributor.

2. Grant of Rights

a) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, Distribute and sublicense the Contribution of such Contributor, if any, and such Derivative Works.

b) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in Source Code or other form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder.

c) Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to Distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.

d) Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement.

e) Notwithstanding the terms of any Secondary License, no Contributor makes additional grants to any Recipient (other than those set forth in this Agreement) as a result of such Recipient's receipt of the Program under the terms of a Secondary License (if permitted under the terms of Section 3).

3. Requirements

3.1 If a Contributor Distributes the Program in any form, then:

* **a)** the Program must also be made available as Source Code, in accordance with section 3.2, and the Contributor must accompany the Program with a statement that the Source Code for the Program is available under this Agreement, and informs Recipients how to obtain it in a reasonable manner on or through a medium customarily used for software exchange; and

* **b)** the Contributor may Distribute the Program under a license different than this Agreement, provided that such license:
 * **i)** effectively disclaims on behalf of all other Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose;

 * **ii)** effectively excludes on behalf of all other Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits;
 * **iii)** does not attempt to limit or alter the recipients' rights in the Source Code under section 3.2; and
 * **iv)** requires any subsequent distribution of the Program by any party to be under a license that satisfies the requirements of this section 3.

3.2 When the Program is Distributed as Source Code:

* **a)** it must be made available under this Agreement, or if the Program **(i)** is combined with other material in a separate file or files made available under a Secondary License, and **(ii)** the initial Contributor attached to the Source Code the notice described in Exhibit A of this Agreement, then the Program may be made available under the terms of such Secondary Licenses, and
* **b)** a copy of this Agreement must be included with each copy of the Program.

3.3 Contributors may not remove or alter any copyright, patent, trademark, attribution notices, disclaimers of warranty, or limitations of liability (notices) contained within the Program from any copy of the Program which they Distribute, provided that Contributors may add their own appropriate notices.

4. Commercial Distribution

Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor (Commercial Contributor) hereby agrees to defend and indemnify every other Contributor (Indemnified Contributor) against any losses, damages and costs (collectively Losses) arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must: **a)** promptly notify the Commercial Contributor in writing of such claim, and **b)** allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those damages.

5. No Warranty

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, AND TO THE EXTENT PERMITTED BY APPLICABLE LAW, THE PROGRAM IS PROVIDED ON AN AS IS BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the

appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

6. Disclaimer of Liability

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, AND TO THE EXTENT PERMITTED BY APPLICABLE LAW, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. General

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. The Eclipse Foundation is the initial Agreement Steward. The Eclipse Foundation may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be Distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to Distribute the Program (including its Contributions) under the new version.

Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved. Nothing in this Agreement is intended to be enforceable by any entity that is not a Contributor or Recipient. No third-party beneficiary rights are created under this Agreement.

Exhibit A - Form of Secondary Licenses Notice

> This Source Code may also be made available under the following Secondary Licenses when the conditions for such availability set forth in the Eclipse Public License, v. 2.0 are satisfied: {name license(s), version(s), and exceptions or additional permissions here}.

Simply including a copy of this Agreement, including this Exhibit A is not sufficient to license the Source Code under Secondary Licenses.

If it is not possible or desirable to put the notice in a particular file, then You may include the notice in a location (such as a LICENSE file in a relevant directory) where a recipient would be likely to look for such a notice.

You may add additional accurate notices of copyright ownership.
Apache License
==========

Version 2.0, January 2004
<<https://www.apache.org/licenses/>>

Terms and Conditions for use, reproduction, and distribution

1. Definitions

License shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

Licensor shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

Legal Entity shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, control means **(i)** the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or **(ii)** ownership of fifty percent (50%) or more of the outstanding shares, or **(iii)** beneficial ownership of such entity.

You (or Your) shall mean an individual or Legal Entity exercising permissions granted by this License.

Source form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

Object form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

Work shall mean the work of authorship, whether in Source or Object form, made

available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

Derivative Works shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

Contribution shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, submitted means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as Not a Contribution.

Contributor shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License

for that Work shall terminate as of the date such litigation is filed.

4. Redistribution

You may reproduce and distribute copies of the Work or Derivative Works thereof
in any medium, with or without modifications, and in Source or Object form,
provided that You meet the following conditions:

* **(a)** You must give any other recipients of the Work or Derivative Works a copy of
this License; and
* **(b)** You must cause any modified files to carry prominent notices stating that You
changed the files; and
* **(c)** You must retain, in the Source form of any Derivative Works that You distribute,
all copyright, patent, trademark, and attribution notices from the Source form
of the Work, excluding those notices that do not pertain to any part of the
Derivative Works; and
* **(d)** If the Work includes a NOTICE text file as part of its distribution, then any
Derivative Works that You distribute must include a readable copy of the
attribution notices contained within such NOTICE file, excluding those notices
that do not pertain to any part of the Derivative Works, in at least one of the
following places: within a NOTICE text file distributed as part of the
Derivative Works; within the Source form or documentation, if provided along
with the Derivative Works; or, within a display generated by the Derivative
Works, if and wherever such third-party notices normally appear. The contents of
the NOTICE file are for informational purposes only and do not modify the
License. You may add Your own attribution notices within Derivative Works that
You distribute, alongside or as an addendum to the NOTICE text from the Work,
provided that such additional attribution notices cannot be construed as
modifying the License.

You may add Your own copyright statement to Your modifications and may provide
additional or different license terms and conditions for use, reproduction, or
distribution of Your modifications, or for any such Derivative Works as a whole,
provided Your use, reproduction, and distribution of the Work otherwise complies
with the conditions stated in this License.

5. Submission of Contributions

Unless You explicitly state otherwise, any Contribution intentionally submitted
for inclusion in the Work by You to the Licensor shall be under the terms and
conditions of this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify the terms of
any separate license agreement you may have executed with Licensor regarding
such Contributions.

6. Trademarks

This License does not grant permission to use the trade names, trademarks,

service marks, or product names of the Licensor, except as required for
reasonable and customary use in describing the origin of the Work and
reproducing the content of the NOTICE file.

7. Disclaimer of Warranty

Unless required by applicable law or agreed to in writing, Licensor provides the
Work (and each Contributor provides its Contributions) on an AS IS BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied,
including, without limitation, any warranties or conditions of TITLE,
NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are
solely responsible for determining the appropriateness of using or
redistributing the Work and assume any risks associated with Your exercise of
permissions under this License.

8. Limitation of Liability

In no event and under no legal theory, whether in tort (including negligence),
contract, or otherwise, unless required by applicable law (such as deliberate
and grossly negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special, incidental,
or consequential damages of any character arising as a result of this License or
out of the use or inability to use the Work (including but not limited to
damages for loss of goodwill, work stoppage, computer failure or malfunction, or
any and all other commercial damages or losses), even if such Contributor has
been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability

While redistributing the Work or Derivative Works thereof, You may choose to
offer, and charge a fee for, acceptance of support, warranty, indemnity, or
other liability obligations and/or rights consistent with this License. However,
in accepting such obligations, You may act only on Your own behalf and on Your
sole responsibility, not on behalf of any other Contributor, and only if You
agree to indemnify, defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason of your
accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work

To apply the Apache License to your work, attach the following boilerplate
notice, with the fields enclosed by brackets `[]` replaced with your own
identifying information. (Don't include the brackets!) The text should be
enclosed in the appropriate comment syntax for the file format. We also
recommend that a file or class name and description of purpose be included on
the same printed page as the copyright notice for easier identification within

third-party archives.

  Copyright [yyyy] [name of copyright owner]

  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    https://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
Open Source Licenses
====================

This product may include a number of subcomponents with separate
copyright notices and license terms. Your use of the source code for
these subcomponents is subject to the terms and conditions of the
subcomponent's license, as noted in the LICENSE-<subcomponent>.md
files.
[[contributors]]
== Contributors

Browse the {junit5-repo}/graphs/contributors[current list of contributors] directly on GitHub.

1.63 netty/tomcatnative-[openssl---classes] 2.0.48.Final

1.63.1 Available under license :

No license file was found, but licenses were detected in source scan.

~ Copyright 2021 The Netty Project
 ~
 ~ The Netty Project licenses this file to you under the Apache License,
 ~ version 2.0 (the "License"); you may not use this file except in compliance
 ~ with the License. You may obtain a copy of the License at:
 ~
 ~   http://www.apache.org/licenses/LICENSE
2.0
 ~
 ~ Unless required by applicable law or agreed to in writing, software
 ~ distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
 ~ WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the

~ License for the specific language governing permissions and limitations
~ under the License.

Found in path(s):
* /opt/cola/permits/1288695178_1648173860.38/0/netty-tcnative-classes-2-0-48-final-jar/META-INF/maven/io.netty/netty-tcnative-classes/pom.xml
No license file was found, but licenses were detected in source scan.

Manifest-Version: 1.0
Implementation-Title: Netty/TomcatNative [OpenSSL - Classes]
Bundle-Description: A Mavenized fork of Tomcat Native which incorporat
es various patches. This artifact is dynamically linked to OpenSSL
and Apache APR.
Automatic-Module-Name: io.netty.internal.tcnative
Bundle-License: http://www.apache.org/licenses/LICENSE-2.0.txt
Bundle-SymbolicName: io.netty.tcnative-classes
Implementation-Version: 2.0.48.Final
Built-By: norman
Bnd-LastModified: 1643125583217
Bundle-ManifestVersion: 2
Implementation-Vendor-Id: io.netty
Require-Capability: osgi.ee;filter:="(&(osgi.ee=JavaSE)(version=1.6))"
Tool: Bnd-5.1.1.202006162103
Export-Package: io.netty.internal.tcnative;version="2.0.48"
Bundle-Name: Netty/TomcatNative [OpenSSL - Classes]
Bundle-Version: 2.0.48.Final
Build-Jdk-Spec: 1.8
Created-By: Apache Maven Bundle Plugin
Build-Jdk: 1.8.0_252
Implementation-URL: https://github.com/netty/netty-tcnative/netty-tcna
tive-classes/

Found in path(s):
* /opt/cola/permits/1288695178_1648173860.38/0/netty-tcnative-classes-2-0-48-final-jar/META-INF/MANIFEST.MF

1.64 kafka-schema-registry-client 5.3.1

1.64.1 Available under license :

No license file was found, but licenses were detected in source scan.

<project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://maven.apache.org/POM/4.0.0"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">

<modelVersion>4.0.0</modelVersion>

```
<parent>
   <groupId>io.confluent</groupId>
   <artifactId>kafka-schema-registry-parent</artifactId>
   <version>5.3.1</version>
</parent>

<licenses>
   <license>
      <name>Apache License 2.0</name>
      <url>http://www.apache.org/licenses/LICENSE-2.0.html</url>
      <distribution>repo</distribution>
   </license>
</licenses>

<artifactId>kafka-schema-registry-client</artifactId>
<packaging>jar</packaging>
<name>kafka-schema-registry-client</name>

<dependencies>
   <dependency>
      <groupId>org.apache.kafka</groupId>
      <artifactId>kafka-clients</artifactId>
   </dependency>
   <dependency>
      <groupId>io.confluent</groupId>
      <artifactId>common-config</artifactId>
   </dependency>

   <dependency>
      <groupId>org.apache.avro</groupId>
      <artifactId>avro</artifactId>
   </dependency>
   <dependency>
      <groupId>com.fasterxml.jackson.core</groupId>
      <artifactId>jackson-databind</artifactId>
   </dependency>

   <dependency>
      <groupId>org.easymock</groupId>
      <artifactId>easymock</artifactId>
      <scope>test</scope>
   </dependency>
   <dependency>
      <groupId>org.powermock</groupId>
      <artifactId>powermock-module-junit4</artifactId>
      <scope>test</scope>
   </dependency>
   <dependency>
```

```
      <groupId>org.powermock</groupId>
      <artifactId>powermock-api-easymock</artifactId>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Found in path(s):
* /opt/cola/permits/1473561838_1669090334.2069662/0/kafka-schema-registry-client-5-3-1-jar/META-INF/maven/io.confluent/kafka-schema-registry-client/pom.xml

1.65 jackson-dataformat-yaml 2.14.0

1.65.1 Available under license :

```
# Jackson JSON processor

Jackson is a high-performance, Free/Open Source JSON processing library.
It was originally written by Tatu Saloranta (tatu.saloranta@iki.fi), and has
been in development since 2007.
It is currently developed by a community of developers, as well as supported
commercially by FasterXML.com.

## Licensing

Jackson core and extension components may be licensed under different licenses.
To find the details that apply to this artifact see the accompanying LICENSE file.
For more information, including possible other licensing options, contact
FasterXML.com (http://fasterxml.com).

## Credits

A list of contributors may be found from CREDITS file, which is included
in some artifacts (usually source distributions); but is always available
from the source code management (SCM) system project uses.
This copy of Jackson JSON processor YAML module is licensed under the
Apache (Software) License, version 2.0 ("the License").
See the License for details about distribution rights, and the
specific rights regarding derivate works.

You may obtain a copy of the License at:

http://www.apache.org/licenses/LICENSE-2.0
```

1.66 apache-log4j-api 2.17.2

1.66.1 Available under license :

Apache Log4j Core
Copyright 1999-2012 Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

ResolverUtil.java
Copyright 2005-2006 Tim Fennell

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a

cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,

any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following
boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright 1999-2005 The Apache Software Foundation

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
Apache Log4j
Copyright 1999-2021 Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

ResolverUtil.java
Copyright 2005-2006 Tim Fennell

Dumbster SMTP test server
Copyright 2004 Jason Paul Kitchen

TypeUtil.java
Copyright 2002-2012 Ramnivas Laddad, Juergen Hoeller, Chris Beams

picocli (http://picocli.info)
Copyright 2017 Remko Popma
Apache Log4j
Copyright 1999-2012 Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

Dumbster SMTP test server
Copyright 2004 Jason Paul Kitchen

Maven Wrapper Jar

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object
form, that is based on (or derived from) the Work and for which the

editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the

Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

   To apply the Apache License to your work, attach the following
   boilerplate notice, with the fields enclosed by brackets "[]"
   replaced with your own identifying information. (Don't include
   the brackets!) The text should be enclosed in the appropriate
   comment syntax for the file format. We also recommend that a
   file or class name and description of purpose be included on the

same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.67 byte-buddy byte-buddy-1.9.10

1.67.1 Available under license :

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,

including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual,

worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf

of any other Contributor, and only if You agree to indemnify,
defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following
boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

Apache Maven includes a number of components and libraries with separate
copyright notices and license terms. Your use of those components are
subject to the terms and conditions of the following licenses.

AOP alliance (http://aopalliance.sourceforge.net) aopalliance:aopalliance:jar:1.0
License: Public Domain

JSR-250 Common Annotations for the JavaTM Platform
(http://jcp.org/aboutJava/communityprocess/final/jsr250/index.html) javax.annotation:jsr250-api:jar:1.0
License: COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL) Version 1.0
https://glassfish.java.net/public/CDDLv1.0.html (lib/jsr250-api.license)

CDI APIs (http://www.seamframework.org/Weld/cdi-api) javax.enterprise:cdi-api:jar:1.0
License: Apache License, Version 2.0 http://www.apache.org/licenses/LICENSE-2.0 (lib/cdi-api.license)

Maven Aether Provider (http://maven.apache.org/ref/3.2.5/maven-aether-provider) org.apache.maven:maven-aether-provider:jar:3.2.5
License: Apache License, Version 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt (lib/maven-aether-provider.license)

Maven Artifact (http://maven.apache.org/ref/3.2.5/maven-artifact) org.apache.maven:maven-artifact:jar:3.2.5
License: Apache License, Version 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt (lib/maven-artifact.license)

Maven Compat (http://maven.apache.org/ref/3.2.5/maven-compat) org.apache.maven:maven-compat:jar:3.2.5
License: Apache License, Version 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt (lib/maven-compat.license)

Maven Core (http://maven.apache.org/ref/3.2.5/maven-core) org.apache.maven:maven-core:jar:3.2.5
License: Apache License, Version 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt (lib/maven-core.license)

Maven Embedder (http://maven.apache.org/ref/3.2.5/maven-embedder) org.apache.maven:maven-embedder:jar:3.2.5
License: Apache License, Version 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt (lib/maven-embedder.license)

Maven Model (http://maven.apache.org/ref/3.2.5/maven-model) org.apache.maven:maven-model:jar:3.2.5
License: Apache License, Version 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt (lib/maven-model.license)

Maven Model Builder (http://maven.apache.org/ref/3.2.5/maven-model-builder) org.apache.maven:maven-model-builder:jar:3.2.5
License: Apache License, Version 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt (lib/maven-model-builder.license)

Maven Plugin API (http://maven.apache.org/ref/3.2.5/maven-plugin-api) org.apache.maven:maven-plugin-api:jar:3.2.5
License: Apache License, Version 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt (lib/maven-plugin-api.license)

Maven Repository Metadata Model (http://maven.apache.org/ref/3.2.5/maven-repository-metadata) org.apache.maven:maven-repository-metadata:jar:3.2.5
License: Apache License, Version 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt (lib/maven-repository-metadata.license)

Maven Settings (http://maven.apache.org/ref/3.2.5/maven-settings) org.apache.maven:maven-settings:jar:3.2.5
License: Apache License, Version 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt (lib/maven-settings.license)

Maven Settings Builder (http://maven.apache.org/ref/3.2.5/maven-settings-builder) org.apache.maven:maven-settings-builder:jar:3.2.5
License: Apache License, Version 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt (lib/maven-settings-

builder.license)

Apache Maven Wagon :: Providers :: File Provider (http://maven.apache.org/wagon/wagon-providers/wagon-file) org.apache.maven.wagon:wagon-file:jar:2.8
License: Apache License, Version 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt (lib/wagon-file.license)

Apache Maven Wagon :: Providers :: HTTP Provider (http://maven.apache.org/wagon/wagon-providers/wagon-http) org.apache.maven.wagon:wagon-http:jar:2.8
License: Apache License, Version 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt (lib/wagon-http.license)

Apache Maven Wagon :: Providers :: HTTP Shared Library (http://maven.apache.org/wagon/wagon-providers/wagon-http-shared) org.apache.maven.wagon:wagon-http-shared:jar:2.8
License: Apache License, Version 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt (lib/wagon-http-shared.license)

Apache Maven Wagon :: API (http://maven.apache.org/wagon/wagon-provider-api) org.apache.maven.wagon:wagon-provider-api:jar:2.8
License: Apache License, Version 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt (lib/wagon-provider-api.license)

Aether API (http://www.eclipse.org/aether/aether-api/) org.eclipse.aether:aether-api:jar:1.0.0.v20140518
License: Eclipse Public License, Version 1.0 http://www.eclipse.org/legal/epl-v10.html (lib/aether-api.license)

Aether Connector Basic (http://www.eclipse.org/aether/aether-connector-basic/) org.eclipse.aether:aether-connector-basic:jar:1.0.0.v20140518
License: Eclipse Public License, Version 1.0 http://www.eclipse.org/legal/epl-v10.html (lib/aether-connector-basic.license)

Aether Implementation (http://www.eclipse.org/aether/aether-impl/) org.eclipse.aether:aether-impl:jar:1.0.0.v20140518
License: Eclipse Public License, Version 1.0 http://www.eclipse.org/legal/epl-v10.html (lib/aether-impl.license)

Aether SPI (http://www.eclipse.org/aether/aether-spi/) org.eclipse.aether:aether-spi:jar:1.0.0.v20140518
License: Eclipse Public License, Version 1.0 http://www.eclipse.org/legal/epl-v10.html (lib/aether-spi.license)

Aether Transport Wagon (http://www.eclipse.org/aether/aether-transport-wagon/) org.eclipse.aether:aether-transport-wagon:jar:1.0.0.v20140518
License: Eclipse Public License, Version 1.0 http://www.eclipse.org/legal/epl-v10.html (lib/aether-transport-wagon.license)

Aether Utilities (http://www.eclipse.org/aether/aether-util/) org.eclipse.aether:aether-util:jar:1.0.0.v20140518
License: Eclipse Public License, Version 1.0 http://www.eclipse.org/legal/epl-v10.html (lib/aether-util.license)

org.eclipse.sisu.inject (http://www.eclipse.org/sisu/org.eclipse.sisu.inject/) org.eclipse.sisu:org.eclipse.sisu.inject:eclipse-plugin:0.3.0.M1
License: Eclipse Public License, Version 1.0 http://www.eclipse.org/legal/epl-v10.html (lib/org.eclipse.sisu.inject.license)

org.eclipse.sisu.plexus (http://www.eclipse.org/sisu/org.eclipse.sisu.plexus/)
org.eclipse.sisu:org.eclipse.sisu.plexus:eclipse-plugin:0.3.0.M1
  License: Eclipse Public License, Version 1.0 http://www.eclipse.org/legal/epl-v10.html
(lib/org.eclipse.sisu.plexus.license)

jsoup (http://jsoup.org/) org.jsoup:jsoup:jar:1.7.2
  License: The MIT License http://jsoup.com/license (lib/jsoup.license)

SLF4J API Module (http://www.slf4j.org) org.slf4j:slf4j-api:jar:1.7.5
  License: MIT License http://www.opensource.org/licenses/mit-license.php (lib/slf4j-api.license)

SLF4J Simple Binding (http://www.slf4j.org) org.slf4j:slf4j-simple:jar:1.7.5
  License: MIT License http://www.opensource.org/licenses/mit-license.php (lib/slf4j-simple.license)

Plexus Cipher: encryption/decryption Component (http://spice.sonatype.org/plexus-cipher)
org.sonatype.plexus:plexus-cipher:jar:1.7
  License: Apache Public License 2.0 http://www.apache.org/licenses/LICENSE-2.0 (lib/plexus-cipher.license)

Plexus Security Dispatcher Component (http://spice.sonatype.org/plexus-sec-dispatcher)
org.sonatype.plexus:plexus-sec-dispatcher:jar:1.3
  License: Apache Public License 2.0 http://www.apache.org/licenses/LICENSE-2.0 (lib/plexus-sec-
dispatcher.license)
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
 "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" version="XHTML+RDFa 1.0" dir="ltr">

<head profile="http://www.w3.org/1999/xhtml/vocab">
 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="shortcut icon" href="http://opensource.org/files/garland_favicon.png" type="image/png" />
<link rel="shortlink" href="/node/66" />
<link rel="canonical" href="/licenses/MIT" />
<meta name="Generator" content="Drupal 7 (http://drupal.org)" />
 <title>The MIT License (MIT) | Open Source Initiative</title>
 <link type="text/css" rel="stylesheet" href="http://opensource.org/files/css/css_xE-rWrJf-
fncB6ztZfd2huxqgxu4WO-qwma6Xer30m4.css" media="all" />
<link type="text/css" rel="stylesheet" href="http://opensource.org/files/css/css_2ATB4XKGEvmoUk_p62PwI-
o2aW47EqqS0nD2dmPZoV4.css" media="all" />
<link type="text/css" rel="stylesheet" href="http://opensource.org/files/css/css_2wI77kyP-
rJKVpFW5M3KFcj7Cb99lZalmubKIwWwsmU.css" media="all" />
<link type="text/css" rel="stylesheet"
href="http://opensource.org/files/css/css_k3snrbsthqot7V7ccRZHS9OkCZkwBv4adtNieIVlbEU.css" media="print"
/>

<!--[if lt IE 7]>
<link type="text/css" rel="stylesheet" href="http://opensource.org/themes/garland/fix-ie.css?nfb7pm" media="all"
/>
<![endif]-->
 <script type="text/javascript"

src="http://opensource.org/files/js/js_xAPl0qIk9eowy_iS9tNkCWXLUVoat94SQT48UBCFkyQ.js"></script>
<script type="text/javascript">
<!--//--><![CDATA[//><!--
jQuery.extend(Drupal.settings,
{"basePath":"\/","pathPrefix":"","ajaxPageState":{"theme":"garland","theme_token":"meUjjBBfr6QFJv5kp0oKi152l673C3xJMLGIQzbH9g0","js":{"misc\/jquery.js":1,"misc\/jquery.once.js":1,"misc\/drupal.js":1},"css":{"modules\/system\/system.base.css":1,"modules\/system\/system.menus.css":1,"modules\/system\/system.messages.css":1,"modules\/system\/system.theme.css":1,"modules\/aggregator\/aggregator.css":1,"modules\/comment\/comment.css":1,"modules\/field\/theme\/field.css":1,"sites\/all\/modules\/mollom\/mollom.css":1,"modules\/node\/node.css":1,"modules\/search\/search.css":1,"modules\/user\/user.css":1,"themes\/garland\/style.css":1,"themes\/garland\/print.css":1,"themes\/garland\/fix-ie.css":1}}});
//--><!]]>
</script>
</head>
<body class="html not-front not-logged-in one-sidebar sidebar-first page-node page-node- page-node-66 node-type-page fluid-width" >

Skip to main content

Open Source Initiative

 <!-- /#header -->

<h2 class="title">Search this site:</h2>

<form action="/licenses/mit-license.php" method="post" id="search-block-form" accept-charset="UTF-8">

<label class="element-invisible" for="edit-search-block-form--2">Search </label>
<input title="Enter the terms you wish to search for." type="text" id="edit-search-block-form--2" name="search_block_form" value="" size="15" maxlength="128" class="form-text" />

<input type="submit" id="edit-submit" name="op"

value="Search" class="form-submit" /> <input type="hidden" name="form_build_id" value="form-MyBqFtDVdzOmaSnCHKIG9yhm0ofMr7fNMG5Vy76N_uk" />
<input type="hidden" name="form_id" value="search_block_form" />

</form>

  <h2 class="title">Navigation</h2>

  <ul class="menu"><li class="first collapsed">About the OSI
<li class="collapsed">The Open Source Definition
<li class="collapsed">Open Source Licenses
<li class="leaf">Working Groups
<li class="leaf">FAQ
<li class="collapsed">Trademark and Logo Usage
<li class="collapsed">Open Standards
<li class="leaf">Open Source Education
<li class="collapsed">Mailing lists
<li class="collapsed">Getting Help
<li class="collapsed">Donate to the OSI
<li class="leaf">OSI Individual Membership
<li class="leaf">OSI Store
<li class="collapsed">OSI Affiliate Membership
<li class="leaf">Contact OSI
<li class="leaf">Terms of Service
<li class="last leaf">OSI Corporate Support

      <h2 class="element-invisible">You are here</h2> Home

                          <h1 class="with-tabs">The MIT License (MIT)</h1>

The MIT License (MIT)
Copyright (c) <year> <copyright holders>
Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.

```
 <div class="content">
   <p style="text-align:center">Help shape the future of the Open Source Initiative...<br /><a
href="http://osi.xwiki.com">visit and participate in the OSI wiki</a>.
</p>

<div>
<a href="https://twitter.com/OpenSourceOrg" class="twitter-follow-button" data-show-count="false" data-
lang="en">Follow @OpenSourceOrg</a>
<script>
<!--//--><![CDATA[// ><!--
!function(d,s,id){var
js,fjs=d.getElementsByTagName(s)[0];if(!d.getElementById(id)){js=d.createElement(s);js.id=id;js.src="//platform.t
witter.com/widgets.js";fjs.parentNode.insertBefore(js,fjs);}}(document,"script","twitter-wjs");
//--><!]]>
</script></div>

<p>
<!-- Creative Commons License --><a rel="license" href="http://creativecommons.org/licenses/by/4.0/"><img
alt="Creative Commons License" style="border-width: 0" src="http://i.creativecommons.org/l/by/4.0/88x31.png"
/></a><br />Opensource.org site content is licensed under a <a rel="license"
href="http://creativecommons.org/licenses/by/4.0/">Creative Commons Attribution 4.0 International
License</a>.<!-- /Creative Commons License -->

<!-- <rdf:RDF xmlns="http://web.resource.org/cc/" xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
 <Work rdf:about="">
  <license rdf:resource="http://creativecommons.org/licenses/by/3.0/" />
 </Work>
 <License rdf:about="http://creativecommons.org/licenses/by/3.0/"><permits
rdf:resource="http://web.resource.org/cc/Reproduction"/><permits
rdf:resource="http://web.resource.org/cc/Distribution"/><requires
rdf:resource="http://web.resource.org/cc/Notice"/><requires
rdf:resource="http://web.resource.org/cc/Attribution"/><permits
rdf:resource="http://web.resource.org/cc/DerivativeWorks"/></License></rdf:RDF>
-->

| <a href="../ToS">Terms of Service</a>

</p>
 </div>
</div>
<div id="block-block-7" class="block block-block clearfix">


 <div class="content">
   <script src="http://www.google-analytics.com/urchin.js" type="text/javascript">
<!--//--><![CDATA[// ><!--
```

```
//--><!]]>
</script><script type="text/javascript">
<!--//--><![CDATA[// ><!--

_uacct = "UA-3916956-1";
urchinTracker();

//--><!]]>
</script>  </div>
</div>
 </div>
    </div></div></div></div> <!-- /.left-corner, /.right-corner, /#squeeze, /#center -->


  </div> <!-- /#container -->
 </div> <!-- /#wrapper -->
 </body>
</html>
```

```
==============================================================================
==  NOTICE file corresponding to the section 4 d of                 ==
==  the Apache License, Version 2.0,                                ==
==  in this case for the Gradle distribution.                       ==
==============================================================================
```

This product includes software developed by
The Apache Software Foundation (http://www.apache.org/).

It includes the following other software:

Groovy (http://groovy-lang.org)
SLF4J (http://www.slf4j.org)
Junit (http://www.junit.org)
JCIFS (http://jcifs.samba.org)

For licenses see the LICENSE file.

If any software distributed with Gradle does not have an Apache 2 License, its license is explicitly listed in the LICENSE file.

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object
form, that is based on (or derived from) the Work and for which the
editorial revisions, annotations, elaborations, or other modifications
represent, as a whole, an original work of authorship. For the purposes
of this License, Derivative Works shall not include works that remain
separable from, or merely link (or bind by name) to the interfaces of,
the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,

and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
whether in tort (including negligence), contract, or otherwise,
unless required by applicable law (such as deliberate and grossly
negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special,
incidental, or consequential damages of any character arising as a
result of this License or out of the use or inability to use the
Work (including but not limited to damages for loss of goodwill,
work stoppage, computer failure or malfunction, or any and all
other commercial damages or losses), even if such Contributor
has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
the Work or Derivative Works thereof, You may choose to offer,
and charge a fee for, acceptance of support, warranty, indemnity,
or other liability obligations and/or rights consistent with this
License. However, in accepting such obligations, You may act only
on Your own behalf and on Your sole responsibility, not on behalf
of any other Contributor, and only if You agree to indemnify,
defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following
boilerplate notice, with the fields enclosed by brackets "{}"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright {yyyy} {name of copyright owner}

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and

limitations under the License.

==========

Gradle Subcomponents:

License for the slf4j package

SLF4J License

Copyright (c) 2004-2007 QOS.ch
All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

These terms are identical to those of the MIT License, also called the X License or the X11 License, which is a simple, permissive non-copyleft free software license. It is deemed compatible with virtually all types of licenses, commercial or otherwise. In particular, the Free Software Foundation has declared it compatible with GNU GPL. It is also known to be approved by the Apache Software Foundation as compatible with Apache Software License.

License for the JUnit package

THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS COMMON PUBLIC LICENSE ("AGREEMENT"). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.

1. DEFINITIONS

"Contribution" means:

a) in the case of the initial Contributor, the initial code and documentation distributed under this Agreement, and

b) in the case of each subsequent Contributor:

i) changes to the Program, and

ii) additions to the Program;

where such changes and/or additions to the Program originate from and are distributed by that particular Contributor. A Contribution 'originates' from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor's behalf. Contributions do not include additions to the Program which: (i) are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program.

"Contributor" means any person or entity that distributes the Program.

"Licensed Patents " mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program.

"Program" means the Contributions distributed in accordance with this Agreement.

"Recipient" means anyone who receives the Program under this Agreement, including all Contributors.

2. GRANT OF RIGHTS

a) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, distribute and sublicense the Contribution of such Contributor, if any, and such derivative works, in source code and object code form.

b) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in source code and object code form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations

which include the Contribution. No hardware per se is licensed hereunder.

c) Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.

d) Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement.

3. REQUIREMENTS

A Contributor may choose to distribute the Program in object code form under its own license agreement, provided that:

a) it complies with the terms and conditions of this Agreement; and

b) its license agreement:

i) effectively disclaims on behalf of all Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose;

ii) effectively excludes on behalf of all Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits;

iii) states that any provisions which differ from this Agreement are offered by that Contributor alone and not by any other party; and

iv) states that source code for the Program is available from such Contributor, and informs licensees how to obtain it in a reasonable manner on or through a medium customarily used for software exchange.

When the Program is made available in source code form:

a) it must be made available under this Agreement; and

b) a copy of this Agreement must be included with each copy of the Program.

Contributors may not remove or alter any copyright notices contained within the Program.

Each Contributor must identify itself as the originator of its Contribution, if any, in a manner that reasonably allows subsequent Recipients to identify the originator of the Contribution.

4. COMMERCIAL DISTRIBUTION

Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor ("Commercial Contributor") hereby agrees to defend and indemnify every other Contributor ("Indemnified Contributor") against any losses, damages and costs (collectively "Losses") arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must: a) promptly notify the Commercial Contributor in writing of such claim, and b) allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those damages.

5. NO WARRANTY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of

rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

6. DISCLAIMER OF LIABILITY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. GENERAL

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against a Contributor with respect to a patent applicable to software (including a cross-claim or counterclaim in a lawsuit), then any patent licenses granted by that Contributor to such Recipient under this Agreement shall terminate as of the date such litigation is filed. In addition, if Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. IBM is the initial Agreement Steward. IBM may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the

Agreement will be given a distinguishing version number. The Program (including Contributions) may always be distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to distribute the Program (including its Contributions) under the new version. Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved.

This Agreement is governed by the laws of the State of New York and the intellectual property laws of the United States of America. No party to this Agreement will bring a legal action under this Agreement more than one year after the cause of action arose. Each party waives its rights to a jury trial in any resulting litigation.

License for the JCIFS package

JCIFS License

GNU LESSER GENERAL PUBLIC LICENSE
Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use,

not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary

General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6.

Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these

materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying

the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the library's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library `Frob' (a library for tweaking knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990
Ty Coon, President of Vice

That's all there is to it!

License for the JGit package

Copyright (c) 2007, Eclipse Foundation, Inc. and its licensors.

All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice,
  this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.
* Neither the name of the Eclipse Foundation, Inc. nor the names of its
  contributors may be used to endorse or promote products derived from this
  software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
Copyright ${project.inceptionYear} - ${current.year} ${copyright.holder}

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

  http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
 <meta http-equiv="content-type"
content="text/html; charset=ISO-8859-1">
 <title>CDDL ver. 1.0</title>
 <meta name="author" content="Cliff Allen">
</head>
<body>
COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL) Version 1.0

1.

Definitions.

1.1. Contributor means each
individual or entity that creates or
contributes to the creation of Modifications.

1.2. Contributor Version
means the combination of the Original
Software, prior Modifications used by a Contributor (if any), and the
Modifications made by that particular Contributor.

1.3. Covered Software means
(a) the Original Software, or (b)
Modifications, or (c) the combination of files containing Original
Software with files containing Modifications, in each case including
portions thereof.

1.4. Executable means the
Covered Software in any form other than
Source Code.

1.5. Initial Developer means
the individual or entity that first makes
Original Software available under this License.

1.6. Larger Work means a work
which combines Covered Software or
portions thereof with code not governed by the terms of this License.

1.7. License means this
document.

1.8. Licensable means having
the right to grant, to the maximum extent
possible, whether at the time of the initial grant or subsequently
acquired, any and all of the rights conveyed herein.

1.9. Modifications means the Source Code and Executable form of any of
the following:
A. Any file that results from an addition to, deletion from or modification of the contents of a file containing Original Software or previous Modifications;
B. Any new file that contains any part of the Original Software or previous Modification; or
C. Any new file that is contributed or otherwise made available under the terms of this License.

1.10. Original Software means the Source Code and Executable form of
computer software code that is originally released under this License.

1.11. Patent Claims means any patent claim(s), now owned or hereafter
acquired, including without limitation, method, process, and apparatus claims, in any patent Licensable by grantor.

1.12. Source Code means (a) the common form of computer software code
in which modifications are made and (b) associated documentation included in or with such code.

1.13. You (or Your) means an individual or a legal entity exercising
rights under, and complying with all of the terms of, this License. For legal entities, You includes any entity which controls, is controlled by, or is under common control with You. For purposes of this definition, control means (a) the power, direct or indirect, to cause
the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the
outstanding shares or beneficial ownership of such entity.

2. License Grants.

 2.1. The Initial Developer Grant.
Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, the Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license:

(a) under intellectual property rights (other than patent or trademark) Licensable by Initial Developer, to use, reproduce, modify, display, perform, sublicense and distribute the Original Software (or portions thereof), with or without Modifications, and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using or selling of Original Software, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Software (or portions thereof);

 (c) The licenses granted in Sections 2.1(a) and (b) are effective on
the date Initial Developer first distributes or otherwise makes the Original Software available to a third party under the terms of this License;

 (d) Notwithstanding Section 2.1(b) above, no patent license is granted:
(1) for code that You delete from the Original Software, or
(2) for
infringements caused by: (i) the modification of the Original Software,
or (ii) the combination of the Original Software with other software or
devices.

2.2. Contributor Grant.
Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims,
each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:

(a) under intellectual property rights (other than patent or trademark) Licensable by Contributor to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof), either on an unmodified basis, with other Modifications, as Covered Software and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such

combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: (1) Modifications made by that Contributor (or
portions thereof); and (2) the combination of Modifications made by
that Contributor with its Contributor Version (or portions of such combination).

(c) The licenses granted in Sections 2.2(a) and 2.2(b) are effective on
the date Contributor first distributes or otherwise makes the Modifications available to a third party.

(d) Notwithstanding Section 2.2(b) above, no patent license is granted:
(1) for any code that Contributor has deleted from the Contributor Version; (2) for infringements caused by: (i) third party modifications
of Contributor Version, or (ii) the combination of Modifications made
by that Contributor with other software (except as part of the Contributor Version) or other devices; or (3) under Patent Claims infringed by Covered Software in the absence of Modifications made by that Contributor.

3. Distribution Obligations.

3.1. Availability of Source Code.
Any Covered Software that You distribute or otherwise make available in Executable form must also be made available in Source Code form and that Source Code form must be distributed only under the terms of this License. You must include a copy of this License with every copy of the Source Code form of the Covered Software You distribute or otherwise make available. You must inform recipients of any such Covered Software in Executable form as to how they can obtain such Covered Software in Source Code form in a reasonable manner on or through a medium customarily used for software exchange.

3.2. Modifications.
The Modifications that You create or to which You contribute are governed by the terms of this License. You represent that You believe Your Modifications are Your original creation(s) and/or You have sufficient rights to grant the rights conveyed by this License.

3.3. Required Notices.
You must include a notice in each of Your Modifications that identifies You as the Contributor of the Modification. You may not remove or alter any copyright, patent or trademark notices contained within the Covered Software, or any notices of licensing or any descriptive text giving attribution to any Contributor or the Initial Developer.

3.4. Application of Additional Terms.
You may not offer or impose any terms on any Covered Software in Source Code form that alters or restricts the applicable version of this License or the recipients rights hereunder. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However, you may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear that any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

3.5. Distribution of Executable Versions.
You may distribute the Executable form of the Covered Software under the terms of this License or under the terms of a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable form
does not attempt to limit or alter the recipients rights in the Source Code form from the rights set forth in this License. If You distribute the Covered Software in Executable form under a different license, You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.

3.6. Larger Works.
You may create a Larger Work by combining Covered Software with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Software.

4. Versions of the License.

4.1. New Versions.
Sun Microsystems, Inc. is the initial license steward and may publish revised and/or new versions of this License from time to time. Each version will be given a distinguishing version number. Except as provided in Section 4.3, no one other than the license steward has the right to modify this License.

4.2. Effect of New Versions.
You may always continue to use, distribute or otherwise make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. If the Initial Developer includes a notice in the Original Software prohibiting it from being distributed or otherwise made available under any subsequent version of the License, You must distribute and make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. Otherwise, You may also choose to use, distribute or otherwise make the Covered Software available under the terms of any subsequent version of the License published by the license steward.

4.3. Modified Versions.
When You are an Initial Developer and You want to create a new license for Your Original Software, You may create and use a modified version of this License if You: (a) rename the license and remove any references to the name of the license steward (except to note that the license differs from this License); and (b) otherwise make it clear
that the license contains terms which differ from this License.

5. DISCLAIMER OF WARRANTY.
COVERED SOFTWARE IS PROVIDED UNDER THIS LICENSE ON AN AS IS BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED SOFTWARE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED SOFTWARE IS WITH YOU. SHOULD ANY COVERED SOFTWARE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED SOFTWARE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

6. TERMINATION.

6.1. This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.

6.2. If You assert a patent infringement claim (excluding declaratory judgment actions) against Initial Developer or a Contributor (the Initial Developer or Contributor against whom You assert such claim is referred to as Participant) alleging that the Participant Software (meaning the Contributor Version where the Participant is a Contributor or the Original Software where the Participant is the Initial Developer) directly or indirectly infringes any patent, then any and all rights granted directly or indirectly to You by such Participant, the Initial Developer (if the Initial Developer is not the Participant) and all Contributors under
Sections 2.1 and/or 2.2 of this License shall, upon 60 days notice from
Participant terminate prospectively and automatically at the expiration of such 60 day notice period, unless if within such 60 day period You withdraw Your claim with respect to the Participant Software against such Participant either unilaterally or pursuant to a written agreement with Participant.

6.3. In the event of termination under Sections 6.1 or 6.2 above, all
end user licenses that have been validly granted by You or any distributor hereunder prior to termination (excluding licenses granted to You by any distributor) shall survive termination.

7. LIMITATION OF LIABILITY.
UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL YOU, THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED SOFTWARE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO ANY PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOST PROFITS, LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTYS NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT

ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

8. U.S. GOVERNMENT END USERS.
The Covered Software is a commercial item, as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of commercial computer software
(as that term is defined at 48 C.F.R. 252.227-7014(a)(1)) and commercial computer software documentation as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48
C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Software with only those rights set forth herein. This U.S. Government Rights clause is in lieu of, and supersedes, any other FAR, DFAR, or other clause or provision that addresses Government rights in computer software under this License.

9. MISCELLANEOUS.
This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall be governed by the law of the jurisdiction specified in a notice contained within the Original Software (except to the extent applicable law, if any, provides otherwise), excluding such jurisdictions conflict-of-law provisions. Any litigation relating to this License shall be subject to the jurisdiction of the courts located in the jurisdiction and venue specified in a notice contained within the Original Software, with the losing party responsible for costs, including, without limitation, court costs and reasonable attorneys fees and expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License. You agree that You alone are responsible for compliance with the United States export administration regulations (and the export control laws and regulation of any other countries) when You use, distribute or otherwise make available any Covered Software.

10. RESPONSIBILITY FOR CLAIMS.
As between Initial Developer and the Contributors, each party is responsible for claims and damages arising, directly or indirectly, out of its utilization of rights under this License and You agree to work with Initial Developer and Contributors to distribute such responsibility on an equitable basis. Nothing herein is intended or shall be deemed to constitute any admission of liability.

NOTICE PURSUANT TO SECTION 9 OF THE COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL)
The code released under the CDDL shall be governed by the
laws of the State of California (excluding conflict-of-law provisions).
Any litigation relating to this License shall be subject to the
jurisdiction of the Federal Courts of the Northern District of
California and the state courts of the State of California, with venue
lying in Santa Clara County, California.
</body>
</html>
<!DOCTYPE html>
<html>
<head>
 <title>jsoup License</title>
 <meta name="keywords" content="license, open source, mit">
 <meta name="description" content="jsoup is licensed under the MIT open source license">
 <link type="text/css" rel="stylesheet" href="/rez/style.css">
 <script type="text/javascript">
    var _gaq = _gaq || [];
    _gaq.push(['_setAccount', 'UA-89734-10']);
     _gaq.push(['_setDomainName', 'jsoup.org']);
    _gaq.push(['_trackPageview']);

    (function() {
      var ga = document.createElement('script'); ga.type = 'text/javascript'; ga.async = true;
      ga.src = ('https:' == document.location.protocol ? 'https://ssl' : 'http://www') + '.google-analytics.com/ga.js';
      var s = document.getElementsByTagName('script')[0]; s.parentNode.insertBefore(ga, s);
    })();
  </script>
</head>
<body class="n1-">

     <li class="n1-home"><h4>jsoup</h4>
     <li class="n1-news">News
     <li class="n1-bugs">Bugs
     <li class="n1-discussion">Discussion
     <li class="n1-download">Download
     <li class="n1-api">API Reference
     <li class="n1-cookbook">Cookbook
     <li class="n1-try">Try jsoup

jsoup
» jsoup License

<h1>jsoup License</h1>
The jsoup code-base (include source and compiled packages) are distributed under the open source MIT license as described below.
<h3>The MIT License</h3>
Copyright © 2009 - 2014 Jonathan Hedley (jonathan@hedley.net)
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

<!-- /col1 -->

<!-- /col2 -->

<!-- /content-->

jsoup HTML parser © 2009 - 2014
Jonathan Hedley

<!-- /wrap -->
<script src="/rez/prettify.js"></script>
<script>prettyPrint();</script>
</body>
</html>
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />

<title>Eclipse Public License - Version 1.0</title>
<style type="text/css">
 body {
   size: 8.5in 11.0in;
   margin: 0.25in 0.5in 0.25in 0.5in;
   tab-interval: 0.5in;
   }
 p {
   margin-left: auto;
   margin-top: 0.5em;
   margin-bottom: 0.5em;
   }
 p.list {
  margin-left: 0.5in;
   margin-top: 0.05em;
   margin-bottom: 0.05em;
   }
 </style>

</head>

<body lang="EN-US">

<h2>Eclipse Public License - v 1.0</h2>

THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS ECLIPSE PUBLIC LICENSE ("AGREEMENT"). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.

1. DEFINITIONS

"Contribution" means:

a) in the case of the initial Contributor, the initial
code and documentation distributed under this Agreement, and
b) in the case of each subsequent Contributor:
i) changes to the Program, and
ii) additions to the Program;
where such changes and/or additions to the Program
originate from and are distributed by that particular Contributor. A
Contribution 'originates' from a Contributor if it was added to the
Program by such Contributor itself or anyone acting on such
Contributor's behalf. Contributions do not include additions to the
Program which: (i) are separate modules of software distributed in
conjunction with the Program under their own license agreement, and (ii)
are not derivative works of the Program.

"Contributor" means any person or entity that distributes the Program.

"Licensed Patents" mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program.

"Program" means the Contributions distributed in accordance with this Agreement.

"Recipient" means anyone who receives the Program under this Agreement, including all Contributors.

2. GRANT OF RIGHTS

a) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, distribute and sublicense the Contribution of such Contributor, if any, and such derivative works, in source code and object code form.

b) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in source code and object code form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder.

c) Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.

d) Each Contributor represents that to its knowledge it

has sufficient copyright rights in its Contribution, if any, to grant
the copyright license set forth in this Agreement.

3. REQUIREMENTS

A Contributor may choose to distribute the Program in object code
form under its own license agreement, provided that:

a) it complies with the terms and conditions of this
Agreement; and

b) its license agreement:

i) effectively disclaims on behalf of all Contributors
all warranties and conditions, express and implied, including warranties
or conditions of title and non-infringement, and implied warranties or
conditions of merchantability and fitness for a particular purpose;

ii) effectively excludes on behalf of all Contributors
all liability for damages, including direct, indirect, special,
incidental and consequential damages, such as lost profits;

iii) states that any provisions which differ from this
Agreement are offered by that Contributor alone and not by any other
party; and

iv) states that source code for the Program is available
from such Contributor, and informs licensees how to obtain it in a
reasonable manner on or through a medium customarily used for software
exchange.

When the Program is made available in source code form:

a) it must be made available under this Agreement; and

b) a copy of this Agreement must be included with each
copy of the Program.

Contributors may not remove or alter any copyright notices contained
within the Program.

Each Contributor must identify itself as the originator of its
Contribution, if any, in a manner that reasonably allows subsequent
Recipients to identify the originator of the Contribution.

4. COMMERCIAL DISTRIBUTION

Commercial distributors of software may accept certain

responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor ("Commercial Contributor") hereby agrees to defend and indemnify every other Contributor ("Indemnified Contributor") against any losses, damages and costs (collectively "Losses") arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must: a) promptly notify the Commercial Contributor in writing of such claim, and b) allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those damages.

5. NO WARRANTY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement , including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

6. DISCLAIMER OF LIABILITY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. GENERAL

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. The Eclipse Foundation is the initial Agreement Steward. The Eclipse Foundation may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to distribute the Program (including its Contributions) under the new version. Except as

expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved.

This Agreement is governed by the laws of the State of New York and the intellectual property laws of the United States of America. No party to this Agreement will bring a legal action under this Agreement more than one year after the cause of action arose. Each party waives its rights to a jury trial in any resulting litigation.

</body>

</html>

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation,

and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s)

with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
   any Contribution intentionally submitted for inclusion in the Work
   by You to the Licensor shall be under the terms and conditions of
   this License, without any additional terms or conditions.
   Notwithstanding the above, nothing herein shall supersede or modify
   the terms of any separate license agreement you may have executed
   with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

License for the slf4j package

SLF4J License

Copyright (c) 2004-2007 QOS.ch
All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF

MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

These terms are identical to those of the MIT License, also called the X License or the X11 License, which is a simple, permissive non-copyleft free software license. It is deemed compatible with virtually all types of licenses, commercial or otherwise. In particular, the Free Software Foundation has declared it compatible with GNU GPL. It is also known to be approved by the Apache Software Foundation as compatible with Apache Software License.

License for the JUnit package

THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS COMMON PUBLIC LICENSE ("AGREEMENT"). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.

1. DEFINITIONS

"Contribution" means:

a) in the case of the initial Contributor, the initial code and documentation distributed under this Agreement, and

b) in the case of each subsequent Contributor:

i) changes to the Program, and

ii) additions to the Program;

where such changes and/or additions to the Program originate from and are distributed by that particular Contributor. A Contribution 'originates' from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor's behalf. Contributions do not include additions to the Program which: (i) are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program.

"Contributor" means any person or entity that distributes the Program.

"Licensed Patents " mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program.

"Program" means the Contributions distributed in accordance with this Agreement.

"Recipient" means anyone who receives the Program under this Agreement, including all Contributors.

2. GRANT OF RIGHTS

a) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, distribute and sublicense the Contribution of such Contributor, if any, and such derivative works, in source code and object code form.

b) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in source code and object code form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder.

c) Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.

d) Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement.

3. REQUIREMENTS

A Contributor may choose to distribute the Program in object code form under its own license agreement, provided that:

a) it complies with the terms and conditions of this Agreement; and

b) its license agreement:

i) effectively disclaims on behalf of all Contributors all warranties and

conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose;

ii) effectively excludes on behalf of all Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits;

iii) states that any provisions which differ from this Agreement are offered by that Contributor alone and not by any other party; and

iv) states that source code for the Program is available from such Contributor, and informs licensees how to obtain it in a reasonable manner on or through a medium customarily used for software exchange.

When the Program is made available in source code form:

a) it must be made available under this Agreement; and

b) a copy of this Agreement must be included with each copy of the Program.

Contributors may not remove or alter any copyright notices contained within the Program.

Each Contributor must identify itself as the originator of its Contribution, if any, in a manner that reasonably allows subsequent Recipients to identify the originator of the Contribution.

4. COMMERCIAL DISTRIBUTION

Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor ("Commercial Contributor") hereby agrees to defend and indemnify every other Contributor ("Indemnified Contributor") against any losses, damages and costs (collectively "Losses") arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must: a) promptly notify the Commercial Contributor in writing of such claim, and b) allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in

any such claim at its own expense.

For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those damages.

5. NO WARRANTY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

6. DISCLAIMER OF LIABILITY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. GENERAL

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against a Contributor with respect to a patent applicable to software (including a cross-claim or counterclaim in a lawsuit), then any patent licenses granted by that Contributor to such Recipient under this Agreement shall terminate as of the date such litigation is filed. In addition, if Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program

itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. IBM is the initial Agreement Steward. IBM may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to distribute the Program (including its Contributions) under the new version. Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved.

This Agreement is governed by the laws of the State of New York and the intellectual property laws of the United States of America. No party to this Agreement will bring a legal action under this Agreement more than one year after the cause of action arose. Each party waives its rights to a jury trial in any resulting litigation.

License for the JCIFS package

JCIFS License

GNU LESSER GENERAL PUBLIC LICENSE
Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original

author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that

you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it

contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application

to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact
that part of it is a work based on the Library, and explaining
where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any

patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR

OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the library's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public
License along with this library; if not, write to the Free Software

Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the library, if
necessary. Here is a sample; alter the names:

 Yoyodyne, Inc., hereby disclaims all copyright interest in the
 library `Frob' (a library for tweaking knobs) written by James Random Hacker.

 <signature of Ty Coon>, 1 April 1990
 Ty Coon, President of Vice

That's all there is to it!
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html lang="en">
 <head>
  <title>Apache License, Version 2.0</title>

  <meta http-equiv="Content-Type" content="text/html;charset=UTF-8">
  <meta property="og:image" content="http://www.apache.org/images/asf_logo.gif" />

  <link rel="stylesheet" type="text/css" media="screen" href="/css/style.css">
  <link rel="stylesheet" type="text/css" media="screen" href="/css/code.css">

  <script type="text/javascript" src="/js/jquery.js"></script>
  <script type="text/javascript" src="/js/apache_boot.js"></script>

  <!-- Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to you under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at . http://www.apache.org/licenses/LICENSE-2.0 . Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License. -->
 </head>

 <body>

    <img src="/images/feather-small.gif" alt="The Apache Software Foundation">
    <h1>The Apache Software Foundation</h1>
    <h2>Apache License, Version 2.0</h2>

<!-- Home -->
Foundation
Projects
People
Get Involved
Download
Support Apache

Home &raquo Licenses
<form name="search" id="search" action="http://www.google.com/search" method="get">
<input value="apache.org" name="sitesearch" type="hidden"/>
<input type="text" name="q" id="query">
<input type="submit" id="submit" value="Search">
</form>

Apache License
</br>Version 2.0, January 2004
</br>
http://www.apache.org/licenses/
TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION
1. Definitions.
"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.
"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.
"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.
"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.
"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.
"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.
"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).
"Derivative Works" shall mean any work, whether in Source or Object form,

that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.
"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."
"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.
2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.
4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:
<ol style="list-style: lower-latin;">
You must give any other recipients of the Work or Derivative Works a

copy of this License; and

You must cause any modified files to carry prominent notices stating
that You changed the files; and

You must retain, in the Source form of any Derivative Works that You
distribute, all copyright, patent, trademark, and attribution notices from
the Source form of the Work, excluding those notices that do not pertain to
any part of the Derivative Works; and

If the Work includes a "NOTICE" text file as part of its distribution,
then any Derivative Works that You distribute must include a readable copy
of the attribution notices contained within such NOTICE file, excluding
those notices that do not pertain to any part of the Derivative Works, in
at least one of the following places: within a NOTICE text file distributed
as part of the Derivative Works; within the Source form or documentation,
if provided along with the Derivative Works; or, within a display generated
by the Derivative Works, if and wherever such third-party notices normally
appear. The contents of the NOTICE file are for informational purposes only
and do not modify the License. You may add Your own attribution notices
within Derivative Works that You distribute, alongside or as an addendum to
the NOTICE text from the Work, provided that such additional attribution
notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may
provide additional or different license terms and conditions for use,
reproduction, or distribution of Your modifications, or for any such
Derivative Works as a whole, provided Your use, reproduction, and
distribution of the Work otherwise complies with the conditions stated in
this License.

5. Submission of Contributions. Unless You
explicitly state otherwise, any Contribution intentionally submitted for
inclusion in the Work by You to the Licensor shall be under the terms and
conditions of this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify the
terms of any separate license agreement you may have executed with Licensor
regarding such Contributions.
6. Trademarks. This License does not grant
permission to use the trade names, trademarks, service marks, or product
names of the Licensor, except as required for reasonable and customary use
in describing the origin of the Work and reproducing the content of the
NOTICE file.
7. Disclaimer of Warranty. Unless required by

applicable law or agreed to in writing, Licensor provides the Work (and
each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT
WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including,
without limitation, any warranties or conditions of TITLE,
NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You
are solely responsible for determining the appropriateness of using or
redistributing the Work and assume any risks associated with Your exercise
of permissions under this License.
8. Limitation of Liability. In no event and
under no legal theory, whether in tort (including negligence), contract, or
otherwise, unless required by applicable law (such as deliberate and
grossly negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special,
incidental, or consequential damages of any character arising as a result
of this License or out of the use or inability to use the Work (including
but not limited to damages for loss of goodwill, work stoppage, computer
failure or malfunction, or any and all other commercial damages or losses),
even if such Contributor has been advised of the possibility of such
damages.
9. Accepting Warranty or Additional Liability.
While redistributing the Work or Derivative Works thereof, You may choose
to offer, and charge a fee for, acceptance of support, warranty, indemnity,
or other liability obligations and/or rights consistent with this License.
However, in accepting such obligations, You may act only on Your own behalf
and on Your sole responsibility, not on behalf of any other Contributor,
and only if You agree to indemnify, defend, and hold each Contributor
harmless for any liability incurred by, or claims asserted against, such
Contributor by reason of your accepting any such warranty or additional
liability.
END OF TERMS AND CONDITIONS
<h1 id="apply">APPENDIX: How to apply the Apache License to your work</h1>
To apply the Apache License to your work, attach the following boilerplate
notice, with the fields enclosed by brackets "[]" replaced with your own
identifying information. (Don't include the brackets!) The text should be
enclosed in the appropriate comment syntax for the file format. We also
recommend that a file or class name and description of purpose be included
on the same "printed page" as the copyright notice for easier
identification within third-party archives.
<pre>Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

  http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
</pre>

<h4>Projects</h4>

HTTP Server
Abdera
Accumulo
ACE
ActiveMQ
Airavata
Allura
Ambari
Ant
Any23
APR
Archiva
Aries
Avro
Axis
Bigtop
Bloodhound
Buildr
BVal
Camel
Cassandra
Cayenne
Chemistry
Chukwa
Clerezza
CloudStack
<a href="http://cocoon.apache.org/" title="Web development framework: separation of concerns, component-

based">Cocoon
Commons
Continuum
Cordova
CouchDB
Creadur
Crunch
cTAKES
Curator
CXF
DB
Deltacloud
DeltaSpike
DirectMemory
Directory
Empire-db
Etch
Felix
Flex
Flume
Forrest
Geronimo
Giraph
Gora
Gump
Hadoop
Hama
HBase
Helix
Hive
HttpComponents
Isis

Jackrabbit
James
jclouds
Jena
JMeter
JSPWiki
jUDDI
Kafka
Karaf
Knox
Lenya
Libcloud
Logging
Lucene
Lucene.Net
Lucy
Mahout
ManifoldCF
Marmotta
Maven
Mesos
MINA
MRUnit
MyFaces
Nutch
ODE
OFBiz
Olingo
Oltu
Onami
OODT
Oozie
Open Climate Workbench
OpenJPA
<a href="http://openmeetings.apache.org/" title="OpenMeetings: Web-Conferencing and real-time

collaboration">OpenMeetings
OpenNLP
OpenOffice
OpenWebBeans
PDFBox
Perl
Pig
Pivot
POI
Portals
Qpid
Rave
River
Roller
Santuario
ServiceMix
Shindig
Shiro
SIS
Sling
SpamAssassin
Spark
Sqoop
Stanbol
STeVe
Storm
Struts
Subversion
Synapse
Syncope
Tajo
Tapestry
Tcl
Tez

Thrift
Tika
Tiles
Tomcat
TomEE
Traffic Server
Turbine
Tuscany
UIMA
VCL
Velocity
VXQuery
Web Services
Whirr
Wicket
Wink
Wookie
Xalan
Xerces
XMLBeans
XML Graphics
ZooKeeper

<h4>Foundation</h4>

FAQ
Glossary
Licenses
Trademarks
News

Press Inquiries
Public Records
Mailing Lists
Sponsorship
Donations
Buy Stuff

```
      <li><a href="/foundation/thanks.html" title="Thank you to our Sponsors">Thanks</a></li>
      <li><a href="/foundation/contact.html" title="Contact Us">Contact</a></li>
    </ul>
  </div>
  <div class="grid_3 suffix_1">
    <h4>Foundation Projects</h4>
    <ul>
      <li><a href="http://attic.apache.org/" title="Inactive projects repository">Attic</a></li>
      <li><a href="/foundation/conferences.html" title="Meetings of developers and users">Conferences</a></li>
      <li><a href="http://community.apache.org/" title="Helping newcomers to the ASF">Community
Development</a></li>
      <li><a href="http://incubator.apache.org/" title="Shepherd for new projects">Incubator</a></li>
      <li><a href="/dev/" title="ASF Infrastructure: Operations and howto documents for PMCs and
contributors">Infrastructure</a></li>
      <li><a href="http://labs.apache.org/" title="The Innovation Laboratories of the Apache Software
Foundation">Labs</a></li>
      <li><a href="/legal/" title="Legal Affairs">Legal Affairs</a></li>
      <li><a href="/press/" title="Public Relations">Public Relations</a></li>
      <li><a href="/security/" title="Security">Security</a></li>
      <li><a href="/travel/" title="Travel Assistance">Travel Assistance</a></li>
    </ul>
  </div>

  <div class="grid_3">
    <h4>Community</h4>
    <ul>
      <li><a href="http://people.apache.org/" title="Apache committer homepages">People</a></li>
      <li><a href="/memorials/" title="In memoriam of past committers">Memorials</a></li>
      <li><a href="http://feathercast.apache.org/" title="Apache Podcasts">Feathercast</a></li>
      <li><a href="http://blogs.apache.org/" title="Apache Project Blogs">Project Blogs</a></li>
      <li><a href="http://planet.apache.org/committers/" title="Apache Committers' Blogs">PlanetApache</a></li>

    </ul>
  </div>
  <div class="grid_3">
    <h4>How It Works</h4>
    <ul>
      <li><a href="/foundation/how-it-works.html">Introduction</a></li>
      <li><a href="/foundation/how-it-works.html#meritocracy">Meritocracy</a></li>
      <li><a href="/foundation/how-it-works.html#structure">Structure</a></li>
      <li><a href="/foundation/how-it-works.html#roles">Roles</a></li>
      <li><a href="/foundation/how-it-works.html#management">Collaboration</a></li>
      <li><a href="/foundation/how-it-works.html#incubator">Incubator</a></li>
      <li><a href="/foundation/how-it-works.html#other">Other entities</a></li>
      <li><a href="/foundation/glossary.html">Glossary</a></li>
      <li><a href="/foundation/voting.html">Voting</a></li>
    </ul>
  </div>
```

```
  </div>
  <div class="clear"></div>

  </div>
  <div id="copyright" class="container_16">
    <p>Copyright &amp;#169; 2012 The Apache Software Foundation, Licensed under the <a
href="http://www.apache.org/licenses/LICENSE-2.0">Apache License, Version 2.0</a>.<br/>Apache and the
Apache feather logo are trademarks of The Apache Software Foundation.</p>
  </div>
 </body>
</html>
```

Apache Maven Distribution
Copyright 2001-2014 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).
Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but

not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object
form, that is based on (or derived from) the Work and for which the
editorial revisions, annotations, elaborations, or other modifications
represent, as a whole, an original work of authorship. For the purposes
of this License, Derivative Works shall not include works that remain
separable from, or merely link (or bind by name) to the interfaces of,
the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity
on behalf of whom a Contribution has been received by Licensor and
subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
copyright license to reproduce, prepare Derivative Works of,
publicly display, publicly perform, sublicense, and distribute the
Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
(except as stated in this section) patent license to make, have made,
use, offer to sell, sell, import, and otherwise transfer the Work,
where such license applies only to those patent claims licensable
by such Contributor that are necessarily infringed by their

Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with

the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
   any Contribution intentionally submitted for inclusion in the Work
   by You to the Licensor shall be under the terms and conditions of
   this License, without any additional terms or conditions.
   Notwithstanding the above, nothing herein shall supersede or modify
   the terms of any separate license agreement you may have executed
   with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "{}" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright {yyyy} {name of copyright owner}

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

```
=========================================================================
==  NOTICE file corresponding to the section 4 d of                    ==
==  the Apache License, Version 2.0,                                   ==
==  in this case for the Gradle distribution.                          ==
=========================================================================
```

This product includes software developed by
The Apache Software Foundation (http://www.apache.org/).

It includes the following other software:

Groovy (http://groovy.codehaus.org)
SLF4J (http://www.slf4j.org)
Junit (http://www.junit.org)
JCIFS (http://jcifs.samba.org)

For licenses see the LICENSE file.

If any software distributed with Gradle does not have an Apache 2 License, its license is explicitly listed in the LICENSE file.

Apache License

Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions

to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices

stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.68 objenesis 3.0.1

1.68.1 Available under license :

// --
// NOTICE file corresponding to the section 4d of The Apache License,
// Version 2.0, in this case for Objenesis
// --

Objenesis
Copyright 2006-2018 Joe Walnes, Henri Tremblay, Leonardo Mesquita

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation

source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable

(except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and

may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify,

defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following
boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.69 jul-to-slf4j-bridge 1.7.26

1.69.1 Available under license :

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION

WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.70 javax.inject:javax.inject 1

1.70.1 Available under license :

No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2009 The JSR-330 Expert Group
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*   http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/cola/permits/1299411403_1650627395.55/0/javax-inject-1-sources-jar/javax/inject/Provider.java
* /opt/cola/permits/1299411403_1650627395.55/0/javax-inject-1-sources-jar/javax/inject/Named.java
* /opt/cola/permits/1299411403_1650627395.55/0/javax-inject-1-sources-jar/javax/inject/Qualifier.java
* /opt/cola/permits/1299411403_1650627395.55/0/javax-inject-1-sources-jar/javax/inject/Inject.java
* /opt/cola/permits/1299411403_1650627395.55/0/javax-inject-1-sources-jar/javax/inject/package-info.java
* /opt/cola/permits/1299411403_1650627395.55/0/javax-inject-1-sources-jar/javax/inject/Scope.java
* /opt/cola/permits/1299411403_1650627395.55/0/javax-inject-1-sources-jar/javax/inject/Singleton.java

1.71 jackson-xc 2.14.0

1.71.1 Available under license :

Jackson JSON processor

Jackson is a high-performance, Free/Open Source JSON processing library.
It was originally written by Tatu Saloranta (tatu.saloranta@iki.fi), and has
been in development since 2007.
It is currently developed by a community of developers, as well as supported
commercially by FasterXML.com.

Licensing

Jackson core and extension components may licensed under different licenses.

To find the details that apply to this artifact see the accompanying LICENSE file.
For more information, including possible other licensing options, contact
FasterXML.com (http://fasterxml.com).

Credits

A list of contributors may be found from CREDITS file, which is included
in some artifacts (usually source distributions); but is always available
from the source code management (SCM) system project uses.
This copy of Jackson JSON processor `jackson-module-jaxb-annotations` module is licensed under the
Apache (Software) License, version 2.0 ("the License").
See the License for details about distribution rights, and the
specific rights regarding derivate works.

You may obtain a copy of the License at:

http://www.apache.org/licenses/LICENSE-2.0

1.72 easymock 4.0.2

1.72.1 Available under license :

// --
// NOTICE file corresponding to the section 4d of The Apache License,
// Version 2.0, in this case for EasyMock
// --

EasyMock
Copyright 2001-2018 EasyMock contributors

This product includes/uses software(s) developed by 'an unknown organization'
- cglib (https://github.com/cglib/cglib/cglib)
- Dexmaker (https://github.com/droidparts/dexmaker)

This product includes/uses software(s) developed by 'Joe Walnes, Henri Tremblay, Leonardo Mesquita'
- Objenesis (http://objenesis.org)

This product includes/uses software(s) developed by 'OW2' (http://www.ow2.org/)
- asm (http://asm.ow2.org/)

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object
form, that is based on (or derived from) the Work and for which the
editorial revisions, annotations, elaborations, or other modifications
represent, as a whole, an original work of authorship. For the purposes
of this License, Derivative Works shall not include works that remain
separable from, or merely link (or bind by name) to the interfaces of,
the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,

and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
whether in tort (including negligence), contract, or otherwise,
unless required by applicable law (such as deliberate and grossly
negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special,
incidental, or consequential damages of any character arising as a
result of this License or out of the use or inability to use the
Work (including but not limited to damages for loss of goodwill,
work stoppage, computer failure or malfunction, or any and all
other commercial damages or losses), even if such Contributor
has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
the Work or Derivative Works thereof, You may choose to offer,
and charge a fee for, acceptance of support, warranty, indemnity,
or other liability obligations and/or rights consistent with this
License. However, in accepting such obligations, You may act only
on Your own behalf and on Your sole responsibility, not on behalf
of any other Contributor, and only if You agree to indemnify,
defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following
boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and

limitations under the License.

1.73 jackson-jaxrs 2.14.0

1.73.1 Available under license :

This copy of Jackson JSON processor databind module is licensed under the
Apache (Software) License, version 2.0 ("the License").
See the License for details about distribution rights, and the
specific rights regarding derivate works.

You may obtain a copy of the License at:

http://www.apache.org/licenses/LICENSE-2.0
Jackson JSON processor

Jackson is a high-performance, Free/Open Source JSON processing library.
It was originally written by Tatu Saloranta (tatu.saloranta@iki.fi), and has
been in development since 2007.
It is currently developed by a community of developers, as well as supported
commercially by FasterXML.com.

Licensing

Jackson core and extension components may be licensed under different licenses.
To find the details that apply to this artifact see the accompanying LICENSE file.
For more information, including possible other licensing options, contact
FasterXML.com (http://fasterxml.com).

Credits

A list of contributors may be found from CREDITS file, which is included
in some artifacts (usually source distributions); but is always available
from the source code management (SCM) system project uses.

1.74 commons-io 2.11.0

1.74.1 Available under license :

Apache Commons IO
Copyright 2002-2021 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (https://www.apache.org/).

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner

or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works
that You distribute, all copyright, patent, trademark, and
attribution notices from the Source form of the Work,
excluding those notices that do not pertain to any part of
the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its
distribution, then any Derivative Works that You distribute must
include a readable copy of the attribution notices contained
within such NOTICE file, excluding those notices that do not
pertain to any part of the Derivative Works, in at least one
of the following places: within a NOTICE text file distributed
as part of the Derivative Works; within the Source form or
documentation, if provided along with the Derivative Works; or,
within a display generated by the Derivative Works, if and
wherever such third-party notices normally appear. The contents
of the NOTICE file are for informational purposes only and
do not modify the License. You may add Your own attribution
notices within Derivative Works that You distribute, alongside
or as an addendum to the NOTICE text from the Work, provided
that such additional attribution notices cannot be construed
as modifying the License.

You may add Your own copyright statement to Your modifications and
may provide additional or different license terms and conditions
for use, reproduction, or distribution of Your modifications, or
for any such Derivative Works as a whole, provided Your use,
reproduction, and distribution of the Work otherwise complies with
the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
any Contribution intentionally submitted for inclusion in the Work
by You to the Licensor shall be under the terms and conditions of
this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify
the terms of any separate license agreement you may have executed
with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
names, trademarks, service marks, or product names of the Licensor,
except as required for reasonable and customary use in describing the
origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
agreed to in writing, Licensor provides the Work (and each
Contributor provides its Contributions) on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied, including, without limitation, any warranties or conditions

of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.75 testng 7.0.0

1.75.1 Available under license :

Apache-2.0

1.76 opentelemetry-java 1.9.1

1.76.1 Available under license :

No license file was found, but licenses were detected in source scan.

<name>Apache License, Version 2.0</name>
<url>http://www.apache.org/licenses/LICENSE-2.0.txt</url>

Found in path(s):
* /opt/cola/permits/1319213317_1651767645.43/0/opentelemetry-sdk-extension-autoconfigure-shaded-1-9-1-alpha-jar/META-INF/maven/org.jctools/jctools-core/pom.xml

1.77 hamcrest 1.3

1.77.1 Available under license :

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
<TITLE>Common Public License - v 1.0</TITLE>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</HEAD>

<BODY BGCOLOR="#FFFFFF" VLINK="#800000">

<P ALIGN="CENTER"><B>Common Public License - v 1.0</B>
<P><B></B><FONT SIZE="3"></FONT>
<P><FONT SIZE="3"></FONT><FONT SIZE="2">THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS COMMON PUBLIC LICENSE ("AGREEMENT"). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.</FONT>
<P><FONT SIZE="2"></FONT>
<P><FONT SIZE="2"><B>1. DEFINITIONS</B></FONT>

<P><FONT SIZE="2">"Contribution" means:</FONT>

<UL><FONT SIZE="2">a) in the case of the initial Contributor, the initial code and documentation distributed under this Agreement, and<BR CLEAR="LEFT">
b) in the case of each subsequent Contributor:</FONT></UL>

<UL><FONT SIZE="2">i) changes to the Program, and</FONT></UL>

<UL><FONT SIZE="2">ii) additions to the Program;</FONT></UL>

<UL><FONT SIZE="2">where such changes and/or additions to the Program originate from and are distributed by that particular Contributor. </FONT><FONT SIZE="2">A Contribution 'originates' from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor's behalf. </FONT><FONT SIZE="2">Contributions do not include additions to the Program which: (i) are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program. </FONT></UL>

<P><FONT SIZE="2"></FONT>
<P><FONT SIZE="2">"Contributor" means any person or entity that distributes the Program.</FONT>
<P><FONT SIZE="2"></FONT><FONT SIZE="2"></FONT>
<P><FONT SIZE="2">"Licensed Patents " mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program. </FONT>
<P><FONT SIZE="2"></FONT><FONT SIZE="2"></FONT>
<P><FONT SIZE="2"></FONT><FONT SIZE="2">"Program" means the Contributions distributed in accordance with this Agreement.</FONT>
<P><FONT SIZE="2"></FONT>
<P><FONT SIZE="2">"Recipient" means anyone who receives the Program under this Agreement, including all Contributors.</FONT>
<P><FONT SIZE="2"><B></B></FONT>
<P><FONT SIZE="2"><B>2. GRANT OF RIGHTS</B></FONT>

<UL><FONT SIZE="2"></FONT><FONT SIZE="2">a) </FONT><FONT SIZE="2">Subject to the terms of this Agreement, each Contributor hereby grants</FONT><FONT SIZE="2"> Recipient a non-exclusive, worldwide, royalty-free copyright license to</FONT><FONT SIZE="2" COLOR="#FF0000"> </FONT><FONT SIZE="2">reproduce, prepare derivative works of, publicly display, publicly perform, distribute and sublicense the Contribution of such Contributor, if any, and such derivative works, in source code and object code form.</FONT></UL>

<UL><FONT SIZE="2"></FONT></UL>

<UL><FONT SIZE="2"></FONT><FONT SIZE="2">b) Subject to the terms of this Agreement, each Contributor hereby grants </FONT><FONT SIZE="2">Recipient a non-exclusive, worldwide,</FONT><FONT SIZE="2" COLOR="#008000"> </FONT><FONT SIZE="2">royalty-free patent license under Licensed Patents to make, use,

sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in source code and object code form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder. </FONT></UL>

<UL><FONT SIZE="2"></FONT></UL>

<UL><FONT SIZE="2">c) Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.</FONT></UL>

<UL><FONT SIZE="2"></FONT></UL>

<UL><FONT SIZE="2">d) Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement. </FONT></UL>

<UL><FONT SIZE="2"></FONT></UL>

<P><FONT SIZE="2"><B>3. REQUIREMENTS</B></FONT>
<P><FONT SIZE="2"><B></B>A Contributor may choose to distribute the Program in object code form under its own license agreement, provided that:</FONT>

<UL><FONT SIZE="2">a) it complies with the terms and conditions of this Agreement; and</FONT></UL>

<UL><FONT SIZE="2">b) its license agreement:</FONT></UL>

<UL><FONT SIZE="2">i) effectively disclaims</FONT><FONT SIZE="2"> on behalf of all Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose; </FONT></UL>

<UL><FONT SIZE="2">ii) effectively excludes on behalf of all Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits; </FONT></UL>

<UL><FONT SIZE="2">iii)</FONT><FONT SIZE="2"> states that any provisions which differ from this Agreement are offered by that Contributor alone and not by any other party; and</FONT></UL>

<UL><FONT SIZE="2">iv) states that source code for the Program is available from such Contributor, and informs licensees how to obtain it in a reasonable manner on or through a medium customarily used for software exchange.</FONT><FONT SIZE="2" COLOR="#0000FF"> </FONT><FONT SIZE="2" COLOR="#FF0000"></FONT></UL>

<UL><FONT SIZE="2" COLOR="#FF0000"></FONT><FONT SIZE="2"></FONT></UL>

<P><FONT SIZE="2">When the Program is made available in source code form:</FONT>

<UL><FONT SIZE="2">a) it must be made available under this Agreement; and </FONT></UL>

<UL><FONT SIZE="2">b) a copy of this Agreement must be included with each copy of the Program. </FONT></UL>

<P><FONT SIZE="2"></FONT><FONT SIZE="2" COLOR="#0000FF"><STRIKE></STRIKE></FONT>
<P><FONT SIZE="2" COLOR="#0000FF"><STRIKE></STRIKE></FONT><FONT SIZE="2">Contributors may not remove or alter any copyright notices contained within the Program. </FONT>
<P><FONT SIZE="2"></FONT>
<P><FONT SIZE="2">Each Contributor must identify itself as the originator of its Contribution, if any, in a manner that reasonably allows subsequent Recipients to identify the originator of the Contribution. </FONT>
<P><FONT SIZE="2"></FONT>
<P><FONT SIZE="2"><B>4. COMMERCIAL DISTRIBUTION</B></FONT>
<P><FONT SIZE="2">Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor ("Commercial Contributor") hereby agrees to defend and indemnify every other Contributor ("Indemnified Contributor") against any losses, damages and costs (collectively "Losses") arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must: a) promptly notify the Commercial Contributor in writing of such claim, and b) allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.</FONT>
<P><FONT SIZE="2"></FONT>
<P><FONT SIZE="2">For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those

damages.</FONT>
<P><FONT SIZE="2"></FONT><FONT SIZE="2" COLOR="#0000FF"></FONT>
<P><FONT SIZE="2" COLOR="#0000FF"></FONT><FONT SIZE="2"><B>5. NO WARRANTY</B></FONT>
<P><FONT SIZE="2">EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is</FONT><FONT SIZE="2"> solely responsible for determining the appropriateness of using and distributing </FONT><FONT SIZE="2">the Program</FONT><FONT SIZE="2"> and assumes all risks associated with its exercise of rights under this Agreement</FONT><FONT SIZE="2">, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, </FONT><FONT SIZE="2">programs or equipment, and unavailability or interruption of operations</FONT><FONT SIZE="2">. </FONT><FONT SIZE="2"></FONT>
<P><FONT SIZE="2"></FONT>
<P><FONT SIZE="2"></FONT><FONT SIZE="2"><B>6. DISCLAIMER OF LIABILITY</B></FONT>
<P><FONT SIZE="2"></FONT><FONT SIZE="2">EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES </FONT><FONT SIZE="2">(INCLUDING WITHOUT LIMITATION LOST PROFITS),</FONT><FONT SIZE="2"> HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.</FONT>
<P><FONT SIZE="2"></FONT><FONT SIZE="2"></FONT>
<P><FONT SIZE="2"><B>7. GENERAL</B></FONT>
<P><FONT SIZE="2"></FONT><FONT SIZE="2">If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.</FONT>
<P><FONT SIZE="2"></FONT>
<P><FONT SIZE="2">If Recipient institutes patent litigation against a Contributor with respect to a patent applicable to software (including a cross-claim or counterclaim in a lawsuit), then any patent licenses granted by that Contributor to such Recipient under this Agreement shall terminate as of the date such litigation is filed. In addition, if Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed. </FONT><FONT SIZE="2"></FONT>
<P><FONT SIZE="2"></FONT>
<P><FONT SIZE="2">All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive. </FONT><FONT SIZE="2"></FONT>
<P><FONT SIZE="2"></FONT>
<P><FONT SIZE="2"></FONT><FONT SIZE="2">Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the

following manner. The Agreement Steward reserves the right to </FONT><FONT SIZE="2">publish new versions (including revisions) of this Agreement from time to </FONT><FONT SIZE="2">time. No one other than the Agreement Steward has the right to modify this Agreement. IBM is the initial Agreement Steward. IBM may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. </FONT><FONT SIZE="2">Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to distribute the Program (including its Contributions) under the new </FONT><FONT SIZE="2">version. </FONT><FONT SIZE="2">Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, </FONT><FONT SIZE="2">by implication, estoppel or otherwise</FONT><FONT SIZE="2">.</FONT><FONT SIZE="2"> All rights in the Program not expressly granted under this Agreement are reserved.</FONT>
<P><FONT SIZE="2"></FONT>
<P><FONT SIZE="2">This Agreement is governed by the laws of the State of New York and the intellectual property laws of the United States of America. No party to this Agreement will bring a legal action under this Agreement more than one year after the cause of action arose. Each party waives its rights to a jury trial in any resulting litigation.</FONT>
<P><FONT SIZE="2"></FONT><FONT SIZE="2"></FONT>
<P><FONT SIZE="2"></FONT>

</BODY>

</HTML>
BSD License

Copyright (c) 2000-2006, www.hamcrest.org
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer. Redistributions in binary form must reproduce
the above copyright notice, this list of conditions and the following disclaimer in
the documentation and/or other materials provided with the distribution.

Neither the name of Hamcrest nor the names of its contributors may be used to endorse
or promote products derived from this software without specific prior written
permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY
EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT

LIMITED
TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY
WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
Copyright (c) 2000-2003, jMock.org
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer. Redistributions in binary form must reproduce
the above copyright notice, this list of conditions and the following disclaimer in
the documentation and/or other materials provided with the distribution.

Neither the name of jMock nor the names of its contributors may be used to endorse
or promote products derived from this software without specific prior written
permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY
EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED
TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY
WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">

<head>
 <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
 <title>EasyMock License</title>
 <link rel="stylesheet" href="easymock.css" />
</head>

<body>

<h2>
EasyMock 2 License (MIT License)
</h2>

Copyright (c) 2001-2006 OFFIS, <a
href="http://www.tammofreese.de">Tammo Freese.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

</body>
</html>

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition,

"control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or,

within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all

other commercial damages or losses), even if such Contributor
has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
the Work or Derivative Works thereof, You may choose to offer,
and charge a fee for, acceptance of support, warranty, indemnity,
or other liability obligations and/or rights consistent with this
License. However, in accepting such obligations, You may act only
on Your own behalf and on Your sole responsibility, not on behalf
of any other Contributor, and only if You agree to indemnify,
defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following
boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.78 apache-log4j 2.17.2

1.78.1 Available under license :

Apache Log4j Docker Library
Copyright 1999-2022 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain

separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or
Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices
stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works
that You distribute, all copyright, patent, trademark, and
attribution notices from the Source form of the Work,
excluding those notices that do not pertain to any part of
the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its
distribution, then any Derivative Works that You distribute must
include a readable copy of the attribution notices contained
within such NOTICE file, excluding those notices that do not
pertain to any part of the Derivative Works, in at least one
of the following places: within a NOTICE text file distributed
as part of the Derivative Works; within the Source form or
documentation, if provided along with the Derivative Works; or,
within a display generated by the Derivative Works, if and
wherever such third-party notices normally appear. The contents
of the NOTICE file are for informational purposes only and
do not modify the License. You may add Your own attribution
notices within Derivative Works that You distribute, alongside
or as an addendum to the NOTICE text from the Work, provided
that such additional attribution notices cannot be construed
as modifying the License.

You may add Your own copyright statement to Your modifications and
may provide additional or different license terms and conditions
for use, reproduction, or distribution of Your modifications, or
for any such Derivative Works as a whole, provided Your use,
reproduction, and distribution of the Work otherwise complies with
the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
any Contribution intentionally submitted for inclusion in the Work
by You to the Licensor shall be under the terms and conditions of
this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify
the terms of any separate license agreement you may have executed
with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
names, trademarks, service marks, or product names of the Licensor,
except as required for reasonable and customary use in describing the

origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
agreed to in writing, Licensor provides the Work (and each
Contributor provides its Contributions) on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied, including, without limitation, any warranties or conditions
of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
PARTICULAR PURPOSE. You are solely responsible for determining the
appropriateness of using or redistributing the Work and assume any
risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
whether in tort (including negligence), contract, or otherwise,
unless required by applicable law (such as deliberate and grossly
negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special,
incidental, or consequential damages of any character arising as a
result of this License or out of the use or inability to use the
Work (including but not limited to damages for loss of goodwill,
work stoppage, computer failure or malfunction, or any and all
other commercial damages or losses), even if such Contributor
has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
the Work or Derivative Works thereof, You may choose to offer,
and charge a fee for, acceptance of support, warranty, indemnity,
or other liability obligations and/or rights consistent with this
License. However, in accepting such obligations, You may act only
on Your own behalf and on Your sole responsibility, not on behalf
of any other Contributor, and only if You agree to indemnify,
defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following
boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.79 error_prone_annotations 2.3.3

1.79.1 Available under license :

No license file was found, but licenses were detected in source scan.

<!--
Copyright 2015 The Error Prone Authors.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->

Found in path(s):
* /opt/cola/permits/1370078224_1658473960.0217261/0/error-prone-annotations-2-3-3-jar/META-INF/maven/com.google.errorprone/error_prone_annotations/pom.xml

1.80 apache-log4j-slf4j-binding 2.17.2

1.80.1 Available under license :

Apache Log4j Core
Copyright 1999-2012 Apache Software Foundation

This product includes software developed at

The Apache Software Foundation (http://www.apache.org/).

ResolverUtil.java
Copyright 2005-2006 Tim Fennell

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes

of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You

meet the following conditions:

(a) You must give any other recipients of the Work or
Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices
stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works
that You distribute, all copyright, patent, trademark, and
attribution notices from the Source form of the Work,
excluding those notices that do not pertain to any part of
the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its
distribution, then any Derivative Works that You distribute must
include a readable copy of the attribution notices contained
within such NOTICE file, excluding those notices that do not
pertain to any part of the Derivative Works, in at least one
of the following places: within a NOTICE text file distributed
as part of the Derivative Works; within the Source form or
documentation, if provided along with the Derivative Works; or,
within a display generated by the Derivative Works, if and
wherever such third-party notices normally appear. The contents
of the NOTICE file are for informational purposes only and
do not modify the License. You may add Your own attribution
notices within Derivative Works that You distribute, alongside
or as an addendum to the NOTICE text from the Work, provided
that such additional attribution notices cannot be construed
as modifying the License.

You may add Your own copyright statement to Your modifications and
may provide additional or different license terms and conditions
for use, reproduction, or distribution of Your modifications, or
for any such Derivative Works as a whole, provided Your use,
reproduction, and distribution of the Work otherwise complies with
the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
any Contribution intentionally submitted for inclusion in the Work
by You to the Licensor shall be under the terms and conditions of
this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify
the terms of any separate license agreement you may have executed
with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
names, trademarks, service marks, or product names of the Licensor,

except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright 1999-2005 The Apache Software Foundation

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

Apache Log4j
Copyright 1999-2021 Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

ResolverUtil.java
Copyright 2005-2006 Tim Fennell

Dumbster SMTP test server
Copyright 2004 Jason Paul Kitchen

TypeUtil.java
Copyright 2002-2012 Ramnivas Laddad, Juergen Hoeller, Chris Beams

picocli (http://picocli.info)
Copyright 2017 Remko Popma
Apache Log4j
Copyright 1999-2012 Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

Dumbster SMTP test server
Copyright 2004 Jason Paul Kitchen

Maven Wrapper Jar
Copyright 2016-2021 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

Apache License
Version 2.0, January 2004

http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally

submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works
that You distribute, all copyright, patent, trademark, and
attribution notices from the Source form of the Work,
excluding those notices that do not pertain to any part of
the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its
distribution, then any Derivative Works that You distribute must
include a readable copy of the attribution notices contained
within such NOTICE file, excluding those notices that do not
pertain to any part of the Derivative Works, in at least one
of the following places: within a NOTICE text file distributed
as part of the Derivative Works; within the Source form or
documentation, if provided along with the Derivative Works; or,
within a display generated by the Derivative Works, if and
wherever such third-party notices normally appear. The contents
of the NOTICE file are for informational purposes only and
do not modify the License. You may add Your own attribution
notices within Derivative Works that You distribute, alongside
or as an addendum to the NOTICE text from the Work, provided
that such additional attribution notices cannot be construed
as modifying the License.

You may add Your own copyright statement to Your modifications and
may provide additional or different license terms and conditions
for use, reproduction, or distribution of Your modifications, or
for any such Derivative Works as a whole, provided Your use,
reproduction, and distribution of the Work otherwise complies with
the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
any Contribution intentionally submitted for inclusion in the Work
by You to the Licensor shall be under the terms and conditions of
this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify
the terms of any separate license agreement you may have executed
with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
names, trademarks, service marks, or product names of the Licensor,
except as required for reasonable and customary use in describing the
origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
agreed to in writing, Licensor provides the Work (and each
Contributor provides its Contributions) on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or

implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.81 roaringbitmap 0.8.11

1.81.1 Available under license :

No license file was found, but licenses were detected in source scan.

/*
* (c) the authors Licensed under the Apache License, Version 2.0.
*/

Found in path(s):
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/longlong/PeekableLongIterator.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/buffer/ImmutableRoaringArray.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/buffer/package-info.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/longlong/RoaringIntPacking.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/IntIterator.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/buffer/BufferUtil.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/buffer/MutableRoaringArray.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/RoaringBitmapSupplier.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/longlong/LongIterator.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/longlong/package-info.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/buffer/BufferReverseIntIteratorFlyweight.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/buffer/MappeableBitmapContainer.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/RunContainer.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/buffer/MappeableContainer.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-

jar/org/roaringbitmap/ArrayContainer.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/ImmutableBitmapDataProvider.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/buffer/MappeableContainerPointer.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/buffer/MutableRoaringBitmap.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/buffer/MutableRoaringBitmapSupplier.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/RoaringBitmap.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/package-info.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/BitmapContainer.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/longlong/LongBitmapDataProvider.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/IntIteratorFlyweight.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/Container.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/buffer/ImmutableRoaringBitmap.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/longlong/ImmutableLongBitmapDataProvider.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/buffer/BufferIntIteratorFlyweight.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/ShortIterator.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/buffer/MappeableRunContainer.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/Util.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/ContainerPointer.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/buffer/PointableRoaringArray.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/longlong/Roaring64NavigableMap.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/FastAggregation.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/longlong/LongConsumer.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/BitmapDataProvider.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/buffer/BufferFastAggregation.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-

jar/org/roaringbitmap/PeekableShortIterator.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/ReverseIntIteratorFlyweight.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/RoaringArray.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/buffer/MappeableArrayContainer.java
* /opt/cola/permits/1473561826_1668493084.8906078/0/roaringbitmap-0-8-11-sources-2-jar/org/roaringbitmap/BitmapDataProviderSupplier.java

1.82 open-telemetry/opentelemetry-java 1.9.1

1.82.1 Available under license :

No license file was found, but licenses were detected in source scan.

```
/*
* Copyright 2014 The gRPC Authors
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
```

Found in path(s):
* /opt/cola/permits/1340815879_1654812121.818436/0/opentelemetry-exporter-otlp-common-1-9-1-sources-jar/io/opentelemetry/exporter/otlp/internal/grpc/MarshalerInputStream.java
* /opt/cola/permits/1340815879_1654812121.818436/0/opentelemetry-exporter-otlp-common-1-9-1-sources-jar/io/opentelemetry/exporter/otlp/internal/grpc/OkHttpGrpcExporter.java
No license file was found, but licenses were detected in source scan.

```
// Copyright 2008 Google Inc.  All rights reserved.
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
//     * Neither the name of Google Inc. nor the names of its
```

// this software without specific prior written permission.

Found in path(s):
* /opt/cola/permits/1340815879_1654812121.818436/0/opentelemetry-exporter-otlp-common-1-9-1-sources-jar/io/opentelemetry/exporter/otlp/internal/CodedOutputStream.java
* /opt/cola/permits/1340815879_1654812121.818436/0/opentelemetry-exporter-otlp-common-1-9-1-sources-jar/io/opentelemetry/exporter/otlp/internal/CodedInputStream.java
* /opt/cola/permits/1340815879_1654812121.818436/0/opentelemetry-exporter-otlp-common-1-9-1-sources-jar/io/opentelemetry/exporter/otlp/internal/WireFormat.java

1.83 commons-compress 1.21

1.83.1 Available under license :

Apache Commons Compress
Copyright 2002-2021 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (https://www.apache.org/).

The files in the package org.apache.commons.compress.archivers.sevenz
were derived from the LZMA SDK, version 9.20 (C/ and CPP/7zip/),
which has been placed in the public domain:

"LZMA SDK is placed in the public domain." (http://www.7-zip.org/sdk.html)

The test file lbzip2_32767.bz2 has been copied from libbzip2's source
repository:

This program, "bzip2", the associated library "libbzip2", and all
documentation, are copyright (C) 1996-2019 Julian R Seward. All
rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
  notice, this list of conditions and the following disclaimer.

2. The origin of this software must not be misrepresented; you must
  not claim that you wrote the original software. If you use this
  software in a product, an acknowledgment in the product
  documentation would be appreciated but is not required.

3. Altered source versions must be plainly marked as such, and must
not be misrepresented as being the original software.

4. The name of the author may not be used to endorse or promote
products derived from this software without specific prior written
permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS
OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Julian Seward, jseward@acm.org

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation

source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable

(except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and

may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify,

defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following
boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.84 jboss-logging 3.3.2.Final

1.84.1 Available under license :

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by

the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object
form, that is based on (or derived from) the Work and for which the
editorial revisions, annotations, elaborations, or other modifications
represent, as a whole, an original work of authorship. For the purposes
of this License, Derivative Works shall not include works that remain
separable from, or merely link (or bind by name) to the interfaces of,
the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained

within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be

liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.85 jackson-databind 2.14.0

1.85.1 Available under license :

Jackson JSON processor

Jackson is a high-performance, Free/Open Source JSON processing library.
It was originally written by Tatu Saloranta (tatu.saloranta@iki.fi), and has
been in development since 2007.
It is currently developed by a community of developers.

Licensing

Jackson 2.x core and extension components are licensed under Apache License 2.0
To find the details that apply to this artifact see the accompanying LICENSE file.

Credits

A list of contributors may be found from CREDITS(-2.x) file, which is included
in some artifacts (usually source distributions); but is always available
from the source code management (SCM) system project uses.

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,

including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual,

worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf

of any other Contributor, and only if You agree to indemnify,
defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following
boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.86 apache-httpcomponents-core 4.4.13

1.86.1 Available under license :

Apache HttpComponents Core
Copyright 2005-2020 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent

to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work,

excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any

risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

1.87 apache-httpcomponents-asyncclient 4.1.3

1.87.1 Available under license :

Apache HttpAsyncClient
Copyright 2010-2017 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object
form, that is based on (or derived from) the Work and for which the
editorial revisions, annotations, elaborations, or other modifications
represent, as a whole, an original work of authorship. For the purposes
of this License, Derivative Works shall not include works that remain
separable from, or merely link (or bind by name) to the interfaces of,
the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,

and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
whether in tort (including negligence), contract, or otherwise,
unless required by applicable law (such as deliberate and grossly
negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special,
incidental, or consequential damages of any character arising as a
result of this License or out of the use or inability to use the
Work (including but not limited to damages for loss of goodwill,
work stoppage, computer failure or malfunction, or any and all
other commercial damages or losses), even if such Contributor
has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
the Work or Derivative Works thereof, You may choose to offer,
and charge a fee for, acceptance of support, warranty, indemnity,
or other liability obligations and/or rights consistent with this
License. However, in accepting such obligations, You may act only
on Your own behalf and on Your sole responsibility, not on behalf
of any other Contributor, and only if You agree to indemnify,
defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following
boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and

limitations under the License.

1.88 joda-time v2.10.2

1.88.1 Available under license :

```
============================================================================
= NOTICE file corresponding to section 4d of the Apache License Version 2.0 =
============================================================================
This product includes software developed by
Joda.org (https://www.joda.org/).
```

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a

copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object
form, that is based on (or derived from) the Work and for which the
editorial revisions, annotations, elaborations, or other modifications
represent, as a whole, an original work of authorship. For the purposes
of this License, Derivative Works shall not include works that remain
separable from, or merely link (or bind by name) to the interfaces of,
the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity
on behalf of whom a Contribution has been received by Licensor and
subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
copyright license to reproduce, prepare Derivative Works of,
publicly display, publicly perform, sublicense, and distribute the
Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
(except as stated in this section) patent license to make, have made,
use, offer to sell, sell, import, and otherwise transfer the Work,
where such license applies only to those patent claims licensable
by such Contributor that are necessarily infringed by their
Contribution(s) alone or by combination of their Contribution(s)
with the Work to which such Contribution(s) was submitted. If You
institute patent litigation against any entity (including a
cross-claim or counterclaim in a lawsuit) alleging that the Work
or a Contribution incorporated within the Work constitutes direct

or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of

this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify
the terms of any separate license agreement you may have executed
with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
names, trademarks, service marks, or product names of the Licensor,
except as required for reasonable and customary use in describing the
origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
agreed to in writing, Licensor provides the Work (and each
Contributor provides its Contributions) on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied, including, without limitation, any warranties or conditions
of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
PARTICULAR PURPOSE. You are solely responsible for determining the
appropriateness of using or redistributing the Work and assume any
risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
whether in tort (including negligence), contract, or otherwise,
unless required by applicable law (such as deliberate and grossly
negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special,
incidental, or consequential damages of any character arising as a
result of this License or out of the use or inability to use the
Work (including but not limited to damages for loss of goodwill,
work stoppage, computer failure or malfunction, or any and all
other commercial damages or losses), even if such Contributor
has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
the Work or Derivative Works thereof, You may choose to offer,
and charge a fee for, acceptance of support, warranty, indemnity,
or other liability obligations and/or rights consistent with this
License. However, in accepting such obligations, You may act only
on Your own behalf and on Your sole responsibility, not on behalf
of any other Contributor, and only if You agree to indemnify,
defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following

boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.89 byte-buddy-agent 1.9.10

1.89.1 Available under license :

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

You must give any other recipients of the Work or Derivative Works a copy of this License; and

You must cause any modified files to carry prominent notices stating that You changed the files; and

You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.
You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.
5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

1.90 jcommander-library 1.72

1.90.1 Available under license :

No license file was found, but licenses were detected in source scan.

```
/**
* Copyright (C) 2011 the original author or authors.
* See the notice.md file distributed with this work for additional
* information regarding copyright ownership.
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
```

Found in path(s):

* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/validators/NoValueValidator.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/IParameterValidator.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/validators/PositiveInteger.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/IParameterValidator2.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/validators/NoValidator.java

No license file was found, but licenses were detected in source scan.

```
/**
* Copyright (C) 2010 the original author or authors.
* See the notice.md file distributed with this work for additional
* information regarding copyright ownership.
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
```

* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/ResourceBundle.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/converters/BaseConverter.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/internal/Sets.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/ParameterDescription.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/converters/IntegerConverter.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/converters/ISO8601DateConverter.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/converters/StringConverter.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/ParametersDelegate.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/converters/LongConverter.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/IDefaultProvider.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/converters/FileConverter.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/converters/PathConverter.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/defaultprovider/PropertyFileDefaultProvider.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/Parameters.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/converters/InetAddressConverter.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/IStringConverterFactory.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/converters/BigDecimalConverter.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/ParameterException.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/converters/FloatConverter.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/converters/NoConverter.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/internal/Maps.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-

jar/com/beust/jcommander/MissingCommandException.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/converters/URIConverter.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/IStringConverter.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/internal/Lists.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/converters/URLConverter.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/converters/BooleanConverter.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/internal/DefaultConverterFactory.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/converters/DoubleConverter.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/converters/CharArrayConverter.java
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/Parameter.java
No license file was found, but licenses were detected in source scan.

/**
* Copyright (C) 2010 the original author or authors.
* See the notice.md file distributed with this work for additional
* information regarding copyright ownership.
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/cola/permits/1009978799_1649800662.98/0/jcommander-1-72-sources-jar/com/beust/jcommander/JCommander.java

1.91 j2objc-annotations 1.3

1.91.1 Available under license :

No license file was found, but licenses were detected in source scan.

```
/*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
```

Found in path(s):
* /opt/cola/permits/1131003150_1612875443.99/0/j2objc-annotations-1-3-sources-3-jar/com/google/j2objc/annotations/RetainedWith.java
* /opt/cola/permits/1131003150_1612875443.99/0/j2objc-annotations-1-3-sources-3-jar/com/google/j2objc/annotations/Property.java
* /opt/cola/permits/1131003150_1612875443.99/0/j2objc-annotations-1-3-sources-3-jar/com/google/j2objc/annotations/LoopTranslation.java
* /opt/cola/permits/1131003150_1612875443.99/0/j2objc-annotations-1-3-sources-3-jar/com/google/j2objc/annotations/ObjectiveCName.java
* /opt/cola/permits/1131003150_1612875443.99/0/j2objc-annotations-1-3-sources-3-jar/com/google/j2objc/annotations/ReflectionSupport.java
* /opt/cola/permits/1131003150_1612875443.99/0/j2objc-annotations-1-3-sources-3-jar/com/google/j2objc/annotations/RetainedLocalRef.java
* /opt/cola/permits/1131003150_1612875443.99/0/j2objc-annotations-1-3-sources-3-jar/com/google/j2objc/annotations/J2ObjCIncompatible.java

No license file was found, but licenses were detected in source scan.

```
/*
* Copyright 2012 Google Inc. All Rights Reserved.
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
```

*/

Found in path(s):
* /opt/cola/permits/1131003150_1612875443.99/0/j2objc-annotations-1-3-sources-3-jar/com/google/j2objc/annotations/Weak.java
* /opt/cola/permits/1131003150_1612875443.99/0/j2objc-annotations-1-3-sources-3-jar/com/google/j2objc/annotations/AutoreleasePool.java
* /opt/cola/permits/1131003150_1612875443.99/0/j2objc-annotations-1-3-sources-3-jar/com/google/j2objc/annotations/WeakOuter.java

1.92 asm-analysis 7.0

1.92.1 Available under license :

<OWNER> = Regents of the University of California
<ORGANIZATION> = University of California, Berkeley
<YEAR> = 1998

In the original BSD license, both occurrences of the phrase "COPYRIGHT HOLDERS AND CONTRIBUTORS" in the disclaimer read "REGENTS AND CONTRIBUTORS".

Here is the license template:

Copyright (c) <YEAR>, <OWNER>

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
Neither the name of the <ORGANIZATION> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

1.93 opentest4j-opentest4j 1.2.0

1.93.1 Available under license :

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes

of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You

meet the following conditions:

(a) You must give any other recipients of the Work or
Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices
stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works
that You distribute, all copyright, patent, trademark, and
attribution notices from the Source form of the Work,
excluding those notices that do not pertain to any part of
the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its
distribution, then any Derivative Works that You distribute must
include a readable copy of the attribution notices contained
within such NOTICE file, excluding those notices that do not
pertain to any part of the Derivative Works, in at least one
of the following places: within a NOTICE text file distributed
as part of the Derivative Works; within the Source form or
documentation, if provided along with the Derivative Works; or,
within a display generated by the Derivative Works, if and
wherever such third-party notices normally appear. The contents
of the NOTICE file are for informational purposes only and
do not modify the License. You may add Your own attribution
notices within Derivative Works that You distribute, alongside
or as an addendum to the NOTICE text from the Work, provided
that such additional attribution notices cannot be construed
as modifying the License.

You may add Your own copyright statement to Your modifications and
may provide additional or different license terms and conditions
for use, reproduction, or distribution of Your modifications, or
for any such Derivative Works as a whole, provided Your use,
reproduction, and distribution of the Work otherwise complies with
the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
any Contribution intentionally submitted for inclusion in the Work
by You to the Licensor shall be under the terms and conditions of
this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify
the terms of any separate license agreement you may have executed
with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
names, trademarks, service marks, or product names of the Licensor,

except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "{}" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright {yyyy} {name of copyright owner}

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.94 slf4j-api-module 1.7.26

1.94.1 Available under license :

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.95 javabeans-activation-framework-api 1.2.2

1.95.1 Available under license :

Copyright (c) 2018 Oracle and/or its affiliates. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

- Redistributions of source code must retain the above copyright
  notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright
  notice, this list of conditions and the following disclaimer in the
  documentation and/or other materials provided with the distribution.

- Neither the name of the Eclipse Foundation, Inc. nor the names of its
  contributors may be used to endorse or promote products derived
  from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

[subs="normal"]
....
Specification: {doctitle}

Version: {revnumber}

ifeval::["{revremark}" != ""]
Status: {revremark}
endif::[]
ifeval::["{revremark}" == ""]
Status: Final Release
endif::[]

Release: {revdate}
....

Copyright (c) 2019 Eclipse Foundation.

=== Eclipse Foundation Specification License

By using and/or copying this document, or the Eclipse Foundation
document from which this statement is linked, you (the licensee) agree
that you have read, understood, and will comply with the following
terms and conditions:

Permission to copy, and distribute the contents of this document, or
the Eclipse Foundation document from which this statement is linked, in
any medium for any purpose and without fee or royalty is hereby
granted, provided that you include the following on ALL copies of the

document, or portions thereof, that you use:

* link or URL to the original Eclipse Foundation document.
* All existing copyright notices, or if one does not exist, a notice
 (hypertext is preferred, but a textual representation is permitted)
 of the form: "Copyright (c) [$date-of-document]
 Eclipse Foundation, Inc. <<url to this license>>"

Inclusion of the full text of this NOTICE must be provided. We
request that authorship attribution be provided in any software,
documents, or other items or products that you create pursuant to the
implementation of the contents of this document, or any portion
thereof.

No right to create modifications or derivatives of Eclipse Foundation
documents is granted pursuant to this license, except anyone may
prepare and distribute derivative works and portions of this document
in software that implements the specification, in supporting materials
accompanying such software, and in documentation of such software,
PROVIDED that all such works include the notice below. HOWEVER, the
publication of derivative works of this document for use as a technical
specification is expressly prohibited.

The notice is:

"Copyright (c) 2018 Eclipse Foundation. This software or
document includes material copied from or derived from [title and URI
of the Eclipse Foundation specification document]."

==== Disclaimers

THIS DOCUMENT IS PROVIDED "AS IS," AND THE COPYRIGHT
HOLDERS AND THE ECLIPSE FOUNDATION MAKE NO REPRESENTATIONS OR
WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE,
NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE DOCUMENT ARE
SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS
WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR
OTHER RIGHTS.

THE COPYRIGHT HOLDERS AND THE ECLIPSE FOUNDATION WILL NOT BE LIABLE
FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT
OF ANY USE OF THE DOCUMENT OR THE PERFORMANCE OR IMPLEMENTATION OF THE
CONTENTS THEREOF.

The name and trademarks of the copyright holders or the Eclipse
Foundation may NOT be used in advertising or publicity pertaining to
this document or its contents without specific, written prior

permission. Title to copyright in this document will at all times
remain with copyright holders.
Notices for Jakarta Activation

This content is produced and maintained by Jakarta Activation project.

* Project home: https://projects.eclipse.org/projects/ee4j.jaf

Copyright

All content is the property of the respective authors or their employers. For
more information regarding authorship of content, please consult the listed
source code repository logs.

Declared Project Licenses

This program and the accompanying materials are made available under the terms
of the Eclipse Distribution License v. 1.0,
which is available at http://www.eclipse.org/org/documents/edl-v10.php.

SPDX-License-Identifier: BSD-3-Clause

Source Code

The project maintains the following source code repositories:

* https://github.com/eclipse-ee4j/jaf

Third-party Content

This project leverages the following third party content.

JUnit (4.12)

* License: Eclipse Public License

1.96 apache-kafka 5.3.1-ccs

1.96.1 Available under license :

No license file was found, but licenses were detected in source scan.

<project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns="http://maven.apache.org/POM/4.0.0"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">

  <modelVersion>4.0.0</modelVersion>

```
<parent>
   <groupId>io.confluent</groupId>
   <artifactId>kafka-schema-registry-parent</artifactId>
   <version>5.3.1</version>
</parent>

<licenses>
   <license>
      <name>Apache License 2.0</name>
      <url>http://www.apache.org/licenses/LICENSE-2.0.html</url>
      <distribution>repo</distribution>
   </license>
</licenses>

<artifactId>kafka-schema-registry-client</artifactId>
<packaging>jar</packaging>
<name>kafka-schema-registry-client</name>

<dependencies>
   <dependency>
      <groupId>org.apache.kafka</groupId>
      <artifactId>kafka-clients</artifactId>
   </dependency>
   <dependency>
      <groupId>io.confluent</groupId>
      <artifactId>common-config</artifactId>
   </dependency>

   <dependency>
      <groupId>org.apache.avro</groupId>
      <artifactId>avro</artifactId>
   </dependency>
   <dependency>
      <groupId>com.fasterxml.jackson.core</groupId>
      <artifactId>jackson-databind</artifactId>
   </dependency>

   <dependency>
      <groupId>org.easymock</groupId>
      <artifactId>easymock</artifactId>
      <scope>test</scope>
   </dependency>
   <dependency>
      <groupId>org.powermock</groupId>
      <artifactId>powermock-module-junit4</artifactId>
      <scope>test</scope>
   </dependency>
   <dependency>
```

```
      <groupId>org.powermock</groupId>
      <artifactId>powermock-api-easymock</artifactId>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Found in path(s):
* /opt/cola/permits/1473561887_1669090334.2604365/0/kafka-schema-registry-client-5-3-1-jar/META-INF/maven/io.confluent/kafka-schema-registry-client/pom.xml

1.97 okhttp 3.14.9

1.97.1 Available under license :

Note that publicsuffixes.gz is compiled from The Public Suffix List:
https://publicsuffix.org/list/public_suffix_list.dat

It is subject to the terms of the Mozilla Public License, v. 2.0:
https://mozilla.org/MPL/2.0/

1.98 junit-platform-junit-platform-engine 1.7.2

1.98.1 Available under license :

```
import java.io.File
import java.net.URI

data class License(val name: String, val url: URI, val headerFile: File)
```

Apache License
==========

Version 2.0, January 2004
<<https://www.apache.org/licenses/>>

Terms and Conditions for use, reproduction, and distribution

1. Definitions

License shall mean the terms and conditions for use, reproduction, and
distribution as defined by Sections 1 through 9 of this document.

Licensor shall mean the copyright owner or entity authorized by the copyright

owner that is granting the License.

Legal Entity shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, control means **(i)** the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or **(ii)** ownership of fifty percent (50%) or more of the outstanding shares, or **(iii)** beneficial ownership of such entity.

You (or Your) shall mean an individual or Legal Entity exercising permissions granted by this License.

Source form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

Object form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

Work shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

Derivative Works shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

Contribution shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, submitted means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as Not a Contribution.

Contributor shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License

Subject to the terms and conditions of this License, each Contributor hereby
grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free,
irrevocable copyright license to reproduce, prepare Derivative Works of,
publicly display, publicly perform, sublicense, and distribute the Work and such
Derivative Works in Source or Object form.

3. Grant of Patent License

Subject to the terms and conditions of this License, each Contributor hereby
grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free,
irrevocable (except as stated in this section) patent license to make, have
made, use, offer to sell, sell, import, and otherwise transfer the Work, where
such license applies only to those patent claims licensable by such Contributor
that are necessarily infringed by their Contribution(s) alone or by combination
of their Contribution(s) with the Work to which such Contribution(s) was
submitted. If You institute patent litigation against any entity (including a
cross-claim or counterclaim in a lawsuit) alleging that the Work or a
Contribution incorporated within the Work constitutes direct or contributory
patent infringement, then any patent licenses granted to You under this License
for that Work shall terminate as of the date such litigation is filed.

4. Redistribution

You may reproduce and distribute copies of the Work or Derivative Works thereof
in any medium, with or without modifications, and in Source or Object form,
provided that You meet the following conditions:

* **(a)** You must give any other recipients of the Work or Derivative Works a copy of
this License; and
* **(b)** You must cause any modified files to carry prominent notices stating that You
changed the files; and
* **(c)** You must retain, in the Source form of any Derivative Works that You distribute,
all copyright, patent, trademark, and attribution notices from the Source form
of the Work, excluding those notices that do not pertain to any part of the
Derivative Works; and
* **(d)** If the Work includes a NOTICE text file as part of its distribution, then any
Derivative Works that You distribute must include a readable copy of the
attribution notices contained within such NOTICE file, excluding those notices
that do not pertain to any part of the Derivative Works, in at least one of the
following places: within a NOTICE text file distributed as part of the
Derivative Works; within the Source form or documentation, if provided along
with the Derivative Works; or, within a display generated by the Derivative
Works, if and wherever such third-party notices normally appear. The contents of
the NOTICE file are for informational purposes only and do not modify the
License. You may add Your own attribution notices within Derivative Works that
You distribute, alongside or as an addendum to the NOTICE text from the Work,

provided that such additional attribution notices cannot be construed as
modifying the License.

You may add Your own copyright statement to Your modifications and may provide
additional or different license terms and conditions for use, reproduction, or
distribution of Your modifications, or for any such Derivative Works as a whole,
provided Your use, reproduction, and distribution of the Work otherwise complies
with the conditions stated in this License.

5. Submission of Contributions

Unless You explicitly state otherwise, any Contribution intentionally submitted
for inclusion in the Work by You to the Licensor shall be under the terms and
conditions of this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify the terms of
any separate license agreement you may have executed with Licensor regarding
such Contributions.

6. Trademarks

This License does not grant permission to use the trade names, trademarks,
service marks, or product names of the Licensor, except as required for
reasonable and customary use in describing the origin of the Work and
reproducing the content of the NOTICE file.

7. Disclaimer of Warranty

Unless required by applicable law or agreed to in writing, Licensor provides the
Work (and each Contributor provides its Contributions) on an AS IS BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied,
including, without limitation, any warranties or conditions of TITLE,
NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are
solely responsible for determining the appropriateness of using or
redistributing the Work and assume any risks associated with Your exercise of
permissions under this License.

8. Limitation of Liability

In no event and under no legal theory, whether in tort (including negligence),
contract, or otherwise, unless required by applicable law (such as deliberate
and grossly negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special, incidental,
or consequential damages of any character arising as a result of this License or
out of the use or inability to use the Work (including but not limited to
damages for loss of goodwill, work stoppage, computer failure or malfunction, or
any and all other commercial damages or losses), even if such Contributor has
been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability

While redistributing the Work or Derivative Works thereof, You may choose to
offer, and charge a fee for, acceptance of support, warranty, indemnity, or
other liability obligations and/or rights consistent with this License. However,
in accepting such obligations, You may act only on Your own behalf and on Your
sole responsibility, not on behalf of any other Contributor, and only if You
agree to indemnify, defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason of your
accepting any such warranty or additional liability.
Eclipse Public License - v 2.0
==========

THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS ECLIPSE PUBLIC LICENSE (AGREEMENT). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.

1. Definitions

Contribution means:
* **a)** in the case of the initial Contributor, the initial content Distributed under this Agreement, and
* **b)** in the case of each subsequent Contributor:
 * **i)** changes to the Program, and
 * **ii)** additions to the Program;
where such changes and/or additions to the Program originate from and are Distributed by that particular Contributor. A Contribution originates from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor's behalf. Contributions do not include changes or additions to the Program that are not Modified Works.

Contributor means any person or entity that Distributes the Program.

Licensed Patents mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program.

Program means the Contributions Distributed in accordance with this Agreement.

Recipient means anyone who receives the Program under this Agreement or any Secondary License (as applicable), including Contributors.

Derivative Works shall mean any work, whether in Source Code or other form, that is based on (or derived from) the Program and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship.

Modified Works shall mean any work in Source Code or other form that results from an addition to, deletion from, or modification of the contents of the Program, including, for purposes of clarity any new file in Source Code form that contains any contents of the Program. Modified Works shall not include works that contain only declarations, interfaces, types, classes, structures, or files of the Program solely in each case in order to link to, bind by name, or subclass the Program or Modified Works thereof.

Distribute means the acts of **a)** distributing or **b)** making available in any manner that enables the transfer of a copy.

Source Code means the form of a Program preferred for making modifications, including but not limited to software source code, documentation source, and configuration files.

Secondary License means either the GNU General Public License, Version 2.0, or any later versions of that license, including any exceptions or additional permissions as identified by the initial Contributor.

2. Grant of Rights

a) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, Distribute and sublicense the Contribution of such Contributor, if any, and such Derivative Works.

b) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in Source Code or other form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder.

c) Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to Distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.

d) Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement.

e) Notwithstanding the terms of any Secondary License, no Contributor makes additional grants to any Recipient (other than those set forth in this Agreement) as a result of such Recipient's receipt of the Program under the terms of a Secondary License (if permitted under the terms of Section 3).

3. Requirements

3.1 If a Contributor Distributes the Program in any form, then:

* **a)** the Program must also be made available as Source Code, in accordance with section 3.2, and the Contributor must accompany the Program with a statement that the Source Code for the Program is available under this Agreement, and informs Recipients how to obtain it in a reasonable manner on or through a medium customarily used for software exchange; and

* **b)** the Contributor may Distribute the Program under a license different than this Agreement, provided that

such license:
 * **i)** effectively disclaims on behalf of all other Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose;
 * **ii)** effectively excludes on behalf of all other Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits;
 * **iii)** does not attempt to limit or alter the recipients' rights in the Source Code under section 3.2; and
 * **iv)** requires any subsequent distribution of the Program by any party to be under a license that satisfies the requirements of this section 3.

3.2 When the Program is Distributed as Source Code:

* **a)** it must be made available under this Agreement, or if the Program **(i)** is combined with other material in a separate file or files made available under a Secondary License, and **(ii)** the initial Contributor attached to the Source Code the notice described in Exhibit A of this Agreement, then the Program may be made available under the terms of such Secondary Licenses, and
* **b)** a copy of this Agreement must be included with each copy of the Program.

3.3 Contributors may not remove or alter any copyright, patent, trademark, attribution notices, disclaimers of warranty, or limitations of liability (notices) contained within the Program from any copy of the Program which they Distribute, provided that Contributors may add their own appropriate notices.

4. Commercial Distribution

Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor (Commercial Contributor) hereby agrees to defend and indemnify every other Contributor (Indemnified Contributor) against any losses, damages and costs (collectively Losses) arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must: **a)** promptly notify the Commercial Contributor in writing of such claim, and **b)** allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those damages.

5. No Warranty

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, AND TO THE EXTENT PERMITTED BY

APPLICABLE LAW, THE PROGRAM IS PROVIDED ON AN AS IS BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

6. Disclaimer of Liability

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, AND TO THE EXTENT PERMITTED BY APPLICABLE LAW, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. General

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. The Eclipse Foundation is the initial Agreement Steward. The Eclipse Foundation may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be Distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to Distribute the Program (including its Contributions) under the new version.

Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All

rights in the Program not expressly granted under this Agreement are reserved. Nothing in this Agreement is intended to be enforceable by any entity that is not a Contributor or Recipient. No third-party beneficiary rights are created under this Agreement.

Exhibit A - Form of Secondary Licenses Notice

> This Source Code may also be made available under the following Secondary Licenses when the conditions for such availability set forth in the Eclipse Public License, v. 2.0 are satisfied: {name license(s), version(s), and exceptions or additional permissions here}.

Simply including a copy of this Agreement, including this Exhibit A is not sufficient to license the Source Code under Secondary Licenses.

If it is not possible or desirable to put the notice in a particular file, then You may include the notice in a location (such as a LICENSE file in a relevant directory) where a recipient would be likely to look for such a notice.

You may add additional accurate notices of copyright ownership.
Apache License
==========

Version 2.0, January 2004
<<https://www.apache.org/licenses/>>

Terms and Conditions for use, reproduction, and distribution

1. Definitions

License shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

Licensor shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

Legal Entity shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, control means **(i)** the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or **(ii)** ownership of fifty percent (50%) or more of the outstanding shares, or **(iii)** beneficial ownership of such entity.

You (or Your) shall mean an individual or Legal Entity exercising permissions granted by this License.

Source form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

Object form shall mean any form resulting from mechanical transformation or

translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

Work shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

Derivative Works shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

Contribution shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, submitted means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as Not a Contribution.

Contributor shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was

submitted. If You institute patent litigation against any entity (including a
cross-claim or counterclaim in a lawsuit) alleging that the Work or a
Contribution incorporated within the Work constitutes direct or contributory
patent infringement, then any patent licenses granted to You under this License
for that Work shall terminate as of the date such litigation is filed.

4. Redistribution

You may reproduce and distribute copies of the Work or Derivative Works thereof
in any medium, with or without modifications, and in Source or Object form,
provided that You meet the following conditions:

* **(a)** You must give any other recipients of the Work or Derivative Works a copy of
this License; and
* **(b)** You must cause any modified files to carry prominent notices stating that You
changed the files; and
* **(c)** You must retain, in the Source form of any Derivative Works that You distribute,
all copyright, patent, trademark, and attribution notices from the Source form
of the Work, excluding those notices that do not pertain to any part of the
Derivative Works; and
* **(d)** If the Work includes a NOTICE text file as part of its distribution, then any
Derivative Works that You distribute must include a readable copy of the
attribution notices contained within such NOTICE file, excluding those notices
that do not pertain to any part of the Derivative Works, in at least one of the
following places: within a NOTICE text file distributed as part of the
Derivative Works; within the Source form or documentation, if provided along
with the Derivative Works; or, within a display generated by the Derivative
Works, if and wherever such third-party notices normally appear. The contents of
the NOTICE file are for informational purposes only and do not modify the
License. You may add Your own attribution notices within Derivative Works that
You distribute, alongside or as an addendum to the NOTICE text from the Work,
provided that such additional attribution notices cannot be construed as
modifying the License.

You may add Your own copyright statement to Your modifications and may provide
additional or different license terms and conditions for use, reproduction, or
distribution of Your modifications, or for any such Derivative Works as a whole,
provided Your use, reproduction, and distribution of the Work otherwise complies
with the conditions stated in this License.

5. Submission of Contributions

Unless You explicitly state otherwise, any Contribution intentionally submitted
for inclusion in the Work by You to the Licensor shall be under the terms and
conditions of this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify the terms of
any separate license agreement you may have executed with Licensor regarding
such Contributions.

6. Trademarks

This License does not grant permission to use the trade names, trademarks,
service marks, or product names of the Licensor, except as required for
reasonable and customary use in describing the origin of the Work and
reproducing the content of the NOTICE file.

7. Disclaimer of Warranty

Unless required by applicable law or agreed to in writing, Licensor provides the
Work (and each Contributor provides its Contributions) on an AS IS BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied,
including, without limitation, any warranties or conditions of TITLE,
NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are
solely responsible for determining the appropriateness of using or
redistributing the Work and assume any risks associated with Your exercise of
permissions under this License.

8. Limitation of Liability

In no event and under no legal theory, whether in tort (including negligence),
contract, or otherwise, unless required by applicable law (such as deliberate
and grossly negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special, incidental,
or consequential damages of any character arising as a result of this License or
out of the use or inability to use the Work (including but not limited to
damages for loss of goodwill, work stoppage, computer failure or malfunction, or
any and all other commercial damages or losses), even if such Contributor has
been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability

While redistributing the Work or Derivative Works thereof, You may choose to
offer, and charge a fee for, acceptance of support, warranty, indemnity, or
other liability obligations and/or rights consistent with this License. However,
in accepting such obligations, You may act only on Your own behalf and on Your
sole responsibility, not on behalf of any other Contributor, and only if You
agree to indemnify, defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason of your
accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work

To apply the Apache License to your work, attach the following boilerplate
notice, with the fields enclosed by brackets `[]` replaced with your own

identifying information. (Don't include the brackets!) The text should be
enclosed in the appropriate comment syntax for the file format. We also
recommend that a file or class name and description of purpose be included on
the same printed page as the copyright notice for easier identification within
third-party archives.

  Copyright [yyyy] [name of copyright owner]

  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    https://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
Open Source Licenses
====================

This product may include a number of subcomponents with separate
copyright notices and license terms. Your use of the source code for
these subcomponents is subject to the terms and conditions of the
subcomponent's license, as noted in the LICENSE-<subcomponent>.md
files.
[[contributors]]
== Contributors

Browse the {junit5-repo}/graphs/contributors[current list of contributors] directly on GitHub.

1.99 netty-transport-native-unix-common 4.1.84.Final

1.99.1 Available under license :

No license file was found, but licenses were detected in source scan.

<!--
 ~ Copyright 2016 The Netty Project
 ~
 ~ The Netty Project licenses this file to you under the Apache License,
 ~ version 2.0 (the "License"); you may not use this file except in compliance
 ~ with the License. You may obtain a copy of the License at:
 ~
 ~   https://www.apache.org/licenses/LICENSE-2.0

~
~ Unless required by applicable law or agreed to in writing, software
~ distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
~ WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
~ License for the specific language governing permissions and limitations
~ under the License.
-->

Found in path(s):
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/META-INF/maven/io.netty/netty-transport-native-unix-common/pom.xml
No license file was found, but licenses were detected in source scan.

/*
* Copyright 2020 The Netty Project
*
* The Netty Project licenses this file to you under the Apache License,
* version 2.0 (the "License"); you may not use this file except in compliance
* with the License. You may obtain a copy of the License at:
*
*   https://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
* WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
* License for the specific language governing permissions and limitations
* under the License.
*/

Found in path(s):
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/netty_unix.h
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/netty_unix.c
No license file was found, but licenses were detected in source scan.

/*
* Copyright 2018 The Netty Project
*
* The Netty Project licenses this file to you under the Apache License,
* version 2.0 (the "License"); you may not use this file except in compliance
* with the License. You may obtain a copy of the License at:
*
*   https://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
* WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the

* License for the specific language governing permissions and limitations
* under the License.
*/

Found in path(s):
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/netty_unix_buffer.h
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/netty_unix_buffer.c
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/PreferredDirectByteBufAllocator.java
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/Buffer.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright 2021 The Netty Project
*
* The Netty Project licenses this file to you under the Apache License,
* version 2.0 (the "License"); you may not use this file except in compliance
* with the License. You may obtain a copy of the License at:
*
*   https://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
* WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
* License for the specific language governing permissions and limitations
* under the License.
*/

Found in path(s):
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/DomainDatagramChannelConfig.java
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/SegmentedDatagramPacket.java
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/DomainDatagramPacket.java
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/DomainDatagramChannel.java
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/DomainDatagramSocketAddress.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright 2015 The Netty Project
*
* The Netty Project licenses this file to you under the Apache License,

* version 2.0 (the "License"); you may not use this file except in compliance
* with the License. You may obtain a copy of the License at:
*
*   https://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
* WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
* License for the specific language governing permissions and limitations
* under the License.
*/

Found in path(s):
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/Socket.java
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/FileDescriptor.java
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/netty_unix_filedescriptor.c
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/netty_unix_socket.c
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/netty_unix_errors.c
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/NativeInetAddress.java
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/UnixChannel.java
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/DatagramSocketAddress.java
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/ServerDomainSocketChannel.java
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/netty_unix_errors.h
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/netty_unix_socket.h
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/DomainSocketChannel.java
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/Errors.java
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/netty_unix_filedescriptor.h
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/DomainSocketReadMode.java
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/DomainSocketAddress.java
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/DomainSocketChannelConfig.java

No license file was found, but licenses were detected in source scan.

/*
* Copyright 2022 The Netty Project
*
* The Netty Project licenses this file to you under the Apache License,
* version 2.0 (the "License"); you may not use this file except in compliance
* with the License. You may obtain a copy of the License at:
*
*   https://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
* WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
* License for the specific language governing permissions and limitations
* under the License.
*/

Found in path(s):
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/GenericUnixChannelOption.java
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/IntegerUnixChannelOption.java
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/RawUnixChannelOption.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright 2014 The Netty Project
*
* The Netty Project licenses this file to you under the Apache License,
* version 2.0 (the "License"); you may not use this file except in compliance
* with the License. You may obtain a copy of the License at:
*
*   https://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
* WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
* License for the specific language governing permissions and limitations
* under the License.
*/

Found in path(s):
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/UnixChannelOption.java
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/Unix.java

* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/package-info.java
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/IovArray.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright 2016 The Netty Project
*
* The Netty Project licenses this file to you under the Apache License,
* version 2.0 (the "License"); you may not use this file except in compliance
* with the License. You may obtain a copy of the License at:
*
*   https://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
* WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
* License for the specific language governing permissions and limitations
* under the License.
*/

Found in path(s):
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/LimitsStaticallyReferencedJniMethods.java
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/netty_unix_util.c
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/PeerCredentials.java
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/ErrorsStaticallyReferencedJniMethods.java
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/netty_unix_util.h
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/netty_unix_limits.c
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/Limits.java
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/netty_unix_limits.h
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/SocketWritableByteChannel.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright 2017 The Netty Project
*
* The Netty Project licenses this file to you under the Apache License,
* version 2.0 (the "License"); you may not use this file except in compliance

* with the License. You may obtain a copy of the License at:
*
*   https://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
* WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
* License for the specific language governing permissions and limitations
* under the License.
*/

Found in path(s):
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/io/netty/channel/unix/UnixChannelUtil.java
* /opt/cola/permits/1470278818_1668107860.126395/0/netty-transport-native-unix-common-4-1-84-final-sources-2-jar/netty_unix_jni.h

1.100 jackson-annotations 2.14.0

1.100.1 Available under license :

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,

including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual,

worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf

of any other Contributor, and only if You agree to indemnify,
defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following
boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.101 apache-zookeeper-jute 3.8.0

1.101.1 Available under license :

This program and the accompanying materials are made available under the
terms of the Eclipse Public License 2.0 which is available at
http://www.eclipse.org/legal/epl-2.0, or the Apache Software License
2.0 which is available at https://www.apache.org/licenses/LICENSE-2.0.

Eclipse Public License - v 1.0

THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS ECLIPSE PUBLIC LICENSE ("AGREEMENT"). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.

1. DEFINITIONS

"Contribution" means:

a) in the case of the initial Contributor, the initial code and documentation distributed under this Agreement, and
b) in the case of each subsequent Contributor:
i) changes to the Program, and
ii) additions to the Program;

where such changes and/or additions to the Program originate from and are distributed by that particular Contributor. A Contribution 'originates' from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor's behalf. Contributions do not include additions to the Program which: (i) are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program.

"Contributor" means any person or entity that distributes the Program.

"Licensed Patents" mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program.

"Program" means the Contributions distributed in accordance with this Agreement.

"Recipient" means anyone who receives the Program under this Agreement, including all Contributors.

2. GRANT OF RIGHTS

a) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, distribute and sublicense the Contribution of such Contributor, if any, and such derivative works, in source code and object code form.
b) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in source code and object code form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder.
c) Recipient understands that although each Contributor grants the licenses

to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.

d) Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement.

3. REQUIREMENTS

A Contributor may choose to distribute the Program in object code form under its own license agreement, provided that:

a) it complies with the terms and conditions of this Agreement; and

b) its license agreement:

i) effectively disclaims on behalf of all Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose;

ii) effectively excludes on behalf of all Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits;

iii) states that any provisions which differ from this Agreement are offered by that Contributor alone and not by any other party; and

iv) states that source code for the Program is available from such Contributor, and informs licensees how to obtain it in a reasonable manner on or through a medium customarily used for software exchange.

When the Program is made available in source code form:

a) it must be made available under this Agreement; and

b) a copy of this Agreement must be included with each copy of the Program. Contributors may not remove or alter any copyright notices contained within the Program.

Each Contributor must identify itself as the originator of its Contribution, if
any, in a manner that reasonably allows subsequent Recipients to identify the originator of the Contribution.

4. COMMERCIAL DISTRIBUTION

Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor ("Commercial Contributor") hereby agrees to defend and indemnify every other Contributor ("Indemnified Contributor") against any losses, damages and costs (collectively "Losses") arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must:
a) promptly notify the Commercial Contributor in writing of such claim, and
b) allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those damages.

5. NO WARRANTY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement , including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

6. DISCLAIMER OF LIABILITY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, NEITHER RECIPIENT NOR ANY

CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. GENERAL

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. The Eclipse Foundation is the initial Agreement Steward. The Eclipse Foundation may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to distribute the Program (including its Contributions) under the new version. Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved.

This Agreement is governed by the laws of the State of New York and the intellectual property laws of the United States of America. No party to this Agreement will bring a legal action under this Agreement more than one year after the cause of action arose. Each party waives its rights to a jury trial in any resulting litigation.

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
Work or Derivative Works thereof in any medium, with or without
modifications, and in Source or Object form, provided that You
meet the following conditions:

(a) You must give any other recipients of the Work or
Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices
stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works
that You distribute, all copyright, patent, trademark, and
attribution notices from the Source form of the Work,
excluding those notices that do not pertain to any part of
the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its
distribution, then any Derivative Works that You distribute must
include a readable copy of the attribution notices contained
within such NOTICE file, excluding those notices that do not
pertain to any part of the Derivative Works, in at least one
of the following places: within a NOTICE text file distributed
as part of the Derivative Works; within the Source form or
documentation, if provided along with the Derivative Works; or,
within a display generated by the Derivative Works, if and
wherever such third-party notices normally appear. The contents
of the NOTICE file are for informational purposes only and
do not modify the License. You may add Your own attribution
notices within Derivative Works that You distribute, alongside
or as an addendum to the NOTICE text from the Work, provided
that such additional attribution notices cannot be construed
as modifying the License.

You may add Your own copyright statement to Your modifications and
may provide additional or different license terms and conditions
for use, reproduction, or distribution of Your modifications, or
for any such Derivative Works as a whole, provided Your use,
reproduction, and distribution of the Work otherwise complies with
the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
any Contribution intentionally submitted for inclusion in the Work
by You to the Licensor shall be under the terms and conditions of
this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify
the terms of any separate license agreement you may have executed

with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate

comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work,

where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or

for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason

of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright 1999-2005 The Apache Software Foundation

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

Copyright (c) 2002, 2004, Christopher Clark
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

* Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.

* Neither the name of the original author; nor the names of any contributors
may be used to endorse or promote products derived from this software
without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS

"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
Net::ZooKeeper - Perl extension for Apache ZooKeeper
Copyright 2009 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,

and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s)

with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
   any Contribution intentionally submitted for inclusion in the Work
   by You to the Licensor shall be under the terms and conditions of
   this License, without any additional terms or conditions.
   Notwithstanding the above, nothing herein shall supersede or modify
   the terms of any separate license agreement you may have executed
   with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright 2010-2012 Coda Hale and Yammer, Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

This contrib module includes software developed under the
COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL) Version 1.0

This contrib depends on binary only jar libraries developed at:

https://jersey.dev.java.net/
https://grizzly.dev.java.net/

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all

other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and

subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed

as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the

Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

This distribution bundles javacc, which is available under the
3-clause BSD License. For details, see a copy of the license in
lib/javacc.LICENSE.txt

This distribution bundles jline 2.14.6, which is available under the
2-clause BSD License. For details, see a copy of the license in
lib/jline-2.14.6.LICENSE.txt

This distribution bundles SLF4J 1.7.30, which is available under the MIT
License. For details, see a copy of the license in
lib/slf4j-1.7.30.LICENSE.txt
Apache ZooKeeper
Copyright 2009-2011 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

---
include/winstdint.h is included only for Windows Client support, as follows:

```
// ISO C9x  compliant stdint.h for Microsoft Visual Studio
// Based on ISO/IEC 9899:TC2 Committee draft (May 6, 2005) WG14/N1124
//
//  Copyright (c) 2006-2008 Alexander Chemeris
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are met:
//
//   1. Redistributions of source code must retain the above copyright notice,
//      this list of conditions and the following disclaimer.
//
//   2. Redistributions in binary form must reproduce the above copyright
//      notice, this list of conditions and the following disclaimer in the
//      documentation and/or other materials provided with the distribution.
//
//   3. The name of the author may be used to endorse or promote products
//      derived from this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS" AND ANY EXPRESS OR IMPLIED
// WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
// MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
// EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
// PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
// OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
// WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
// OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
// ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//
///////////////////////////////////////////////////////////////////////////////
```

---

src/java/com/nitido/utils/toaster/Toaster.java:
This java file is copyright by Daniele Piras ("danielepiras80", no email known) released under the Apache Software License 2.0
It has been downloaded in december 2009 from the CVS web interface of the sourceforge project http://sourceforge.net/projects/jtoaster/ . The web interface to CVS is not available anymore on sourceforge.

The icons in src/main/resources/icons are taken from the Tango project
downloaded from http://tango.freedesktop.org/releases on 2011-09-06.
The Tango project is public domain.

Distribution packagers should not include the icons in the package but rather depend on tango-icon-theme (Debian package name). ZooInspector will then try to get the icons from /usr/share/icons/Tango rather then from its jar file.
Apache ZooKeeper
Copyright 2009-2022 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

This product includes software components originally
developed for Airlift (https://github.com/airlift/airlift),
licensed under the Apache 2.0 license. The licensing terms
for Airlift code can be found at:
https://github.com/airlift/airlift/blob/master/LICENSE

This project also includes some files with the following licenses.

These BSD licensed files:
./zookeeper-client/zookeeper-client-c/src/hashtable/hashtable.c
./zookeeper-client/zookeeper-client-c/src/hashtable/hashtable.h
./zookeeper-client/zookeeper-client-c/src/hashtable/hashtable_itr.c
./zookeeper-client/zookeeper-client-c/src/hashtable/hashtable_itr.h
./zookeeper-client/zookeeper-client-c/src/hashtable/hashtable_private.h
./zookeeper-contrib/zookeeper-contrib-loggraph/src/main/resources/webapp/org/apache/zookeeper/graph/resources/yui-min.js
./zookeeper-docs/src/main/resources/markdown/skin/prototype.js

These MIT licensed files:
./zookeeper-contrib/zookeeper-contrib-loggraph/src/main/resources/webapp/org/apache/zookeeper/graph/resources/date.format.js
./zookeeper-contrib/zookeeper-contrib-loggraph/src/main/resources/webapp/org/apache/zookeeper/graph/resources/g.bar.js
./zookeeper-contrib/zookeeper-contrib-loggraph/src/main/resources/webapp/org/apache/zookeeper/graph/resources/g.dot.js
./zookeeper-contrib/zookeeper-contrib-loggraph/src/main/resources/webapp/org/apache/zookeeper/graph/resources/g.line.js
./zookeeper-contrib/zookeeper-contrib-loggraph/src/main/resources/webapp/org/apache/zookeeper/graph/resources/g.pie.js
./zookeeper-contrib/zookeeper-contrib-

loggraph/src/main/resources/webapp/org/apache/zookeeper/graph/resources/g.raphael.js
./zookeeper-contrib/zookeeper-contrib-loggraph/src/main/resources/webapp/org/apache/zookeeper/graph/resources/raphael.js

This Apache 2.0 licensed file:
./zookeeper-contrib/zookeeper-contrib-zooinspector/src/main/java/com/nitido/utils/toaster/Toaster.java
Apache ZooKeeper
Copyright 2009-2021 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

This product includes software components originally
developed for Airlift (https://github.com/airlift/airlift),
licensed under the Apache 2.0 license. The licensing terms
for Airlift code can be found at:
https://github.com/airlift/airlift/blob/master/LICENSE

This product includes software developed by
The Netty Project (http://netty.io/)
Copyright 2011 The Netty Project

The Netty NOTICE file contains the following items:
This product contains the extensions to Java Collections Framework which has
been derived from the works by JSR-166 EG, Doug Lea, and Jason T. Greene:

* LICENSE:
  * license/LICENSE.jsr166y.txt (Public Domain)
* HOMEPAGE:
  * http://gee.cs.oswego.edu/cgi-bin/viewcvs.cgi/jsr166/
  * http://viewvc.jboss.org/cgi-bin/viewvc.cgi/jbosscache/experimental/jsr166/

This product contains a modified version of Robert Harder's Public Domain
Base64 Encoder and Decoder, which can be obtained at:

* LICENSE:
  * license/LICENSE.base64.txt (Public Domain)
* HOMEPAGE:
  * http://iharder.sourceforge.net/current/java/base64/

This product contains a modified version of 'JZlib', a re-implementation of
zlib in pure Java, which can be obtained at:

* LICENSE:
  * license/LICENSE.jzlib.txt (BSD Style License)
* HOMEPAGE:
  * http://www.jcraft.com/jzlib/

This product contains a modified version of 'Webbit', a Java event based
WebSocket and HTTP server:

 * LICENSE:
   * license/LICENSE.webbit.txt (BSD License)
 * HOMEPAGE:
   * https://github.com/joewalnes/webbit

This product optionally depends on 'Protocol Buffers', Google's data
interchange format, which can be obtained at:

 * LICENSE:
   * license/LICENSE.protobuf.txt (New BSD License)
 * HOMEPAGE:
   * http://code.google.com/p/protobuf/

This product optionally depends on 'Bouncy Castle Crypto APIs' to generate
a temporary self-signed X.509 certificate when the JVM does not provide the
equivalent functionality. It can be obtained at:

 * LICENSE:
   * license/LICENSE.bouncycastle.txt (MIT License)
 * HOMEPAGE:
   * http://www.bouncycastle.org/

This product optionally depends on 'SLF4J', a simple logging facade for Java,
which can be obtained at:

 * LICENSE:
   * license/LICENSE.slf4j.txt (MIT License)
 * HOMEPAGE:
   * http://www.slf4j.org/

This product optionally depends on 'Apache Commons Logging', a logging
framework, which can be obtained at:

 * LICENSE:
   * license/LICENSE.commons-logging.txt (Apache License 2.0)
 * HOMEPAGE:
   * http://commons.apache.org/logging/

This product optionally depends on 'Apache Logback', a logging framework,
which can be obtained at:

 * LICENSE:
   * license/LICENSE.logback.txt (Eclipse Public License 1.0)
 * HOMEPAGE:
   * https://logback.qos.ch/

This product optionally depends on 'JBoss Logging', a logging framework,
which can be obtained at:

 * LICENSE:
   * license/LICENSE.jboss-logging.txt (GNU LGPL 2.1)
 * HOMEPAGE:
   * http://anonsvn.jboss.org/repos/common/common-logging-spi/

This product optionally depends on 'Apache Felix', an open source OSGi
framework implementation, which can be obtained at:

 * LICENSE:
   * license/LICENSE.felix.txt (Apache License 2.0)
 * HOMEPAGE:
   * http://felix.apache.org/

The bundled library Metrics Core NOTICE file reports the following items

Metrics
Copyright 2010-2013 Coda Hale and Yammer, Inc.

This product includes software developed by Coda Hale and Yammer, Inc.

This product includes code derived from the JSR-166 project (ThreadLocalRandom, Striped64,
LongAdder), which was released with the following comments:

    Written by Doug Lea with assistance from members of JCP JSR-166
    Expert Group and released to the public domain, as explained at
    http://creativecommons.org/publicdomain/zero/1.0/

The Nappy Java NOTICE file reports the following items:

This product includes software developed by Google
Snappy: http://code.google.com/p/snappy/ (New BSD License)

This product includes software developed by Apache
PureJavaCrc32C from apache-hadoop-common http://hadoop.apache.org/
(Apache 2.0 license)

This library containd statically linked libstdc++. This inclusion is allowed by
"GCC RUntime Library Exception"
http://gcc.gnu.org/onlinedocs/libstdc++/manual/license.html

== Contributors ==
 * Tatu Saloranta
   * Providing benchmark suite
 * Alec Wysoker

* Performance and memory usage improvement

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity
on behalf of whom a Contribution has been received by Licensor and
subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
copyright license to reproduce, prepare Derivative Works of,
publicly display, publicly perform, sublicense, and distribute the
Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
(except as stated in this section) patent license to make, have made,
use, offer to sell, sell, import, and otherwise transfer the Work,
where such license applies only to those patent claims licensable
by such Contributor that are necessarily infringed by their
Contribution(s) alone or by combination of their Contribution(s)
with the Work to which such Contribution(s) was submitted. If You
institute patent litigation against any entity (including a
cross-claim or counterclaim in a lawsuit) alleging that the Work
or a Contribution incorporated within the Work constitutes direct
or contributory patent infringement, then any patent licenses
granted to You under this License for that Work shall terminate
as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
Work or Derivative Works thereof in any medium, with or without
modifications, and in Source or Object form, provided that You
meet the following conditions:

(a) You must give any other recipients of the Work or

Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
agreed to in writing, Licensor provides the Work (and each
Contributor provides its Contributions) on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied, including, without limitation, any warranties or conditions
of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
PARTICULAR PURPOSE. You are solely responsible for determining the
appropriateness of using or redistributing the Work and assume any
risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
whether in tort (including negligence), contract, or otherwise,
unless required by applicable law (such as deliberate and grossly
negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special,
incidental, or consequential damages of any character arising as a
result of this License or out of the use or inability to use the
Work (including but not limited to damages for loss of goodwill,
work stoppage, computer failure or malfunction, or any and all
other commercial damages or losses), even if such Contributor
has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
the Work or Derivative Works thereof, You may choose to offer,
and charge a fee for, acceptance of support, warranty, indemnity,
or other liability obligations and/or rights consistent with this
License. However, in accepting such obligations, You may act only
on Your own behalf and on Your sole responsibility, not on behalf
of any other Contributor, and only if You agree to indemnify,
defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following
boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

```
================================================================================
=========
===         The following part contains the dual OpenSSL and SSLeay license         ===
===         for OpenSSL versions 1.1.1, 1.1.0, 1.0.2 and all prior releases         ===
===              (see https://www.openssl.org/source/license.html)              ===
================================================================================
=========
```

LICENSE ISSUES
==========

The OpenSSL toolkit stays under a double license, i.e. both the conditions of
the OpenSSL License and the original SSLeay license apply to the toolkit.
See below for the actual license texts.

OpenSSL License

```
/* ====================================================================
* Copyright (c) 1998-2019 The OpenSSL Project.  All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
*
* 1. Redistributions of source code must retain the above copyright
*    notice, this list of conditions and the following disclaimer.
*
* 2. Redistributions in binary form must reproduce the above copyright
*    notice, this list of conditions and the following disclaimer in
*    the documentation and/or other materials provided with the
*    distribution.
*
```

* 3. All advertising materials mentioning features or use of this
*   software must display the following acknowledgment:
*   "This product includes software developed by the OpenSSL Project
*   for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
*
* 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
*   endorse or promote products derived from this software without
*   prior written permission. For written permission, please contact
*   openssl-core@openssl.org.
*
* 5. Products derived from this software may not be called "OpenSSL"
*   nor may "OpenSSL" appear in their names without prior written
*   permission of the OpenSSL Project.
*
* 6. Redistributions of any form whatsoever must retain the following
*   acknowledgment:
*   "This product includes software developed by the OpenSSL Project
*   for use in the OpenSSL Toolkit (http://www.openssl.org/)"
*
* THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
* EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
* PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR
* ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
* NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
* LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
* STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
* ==
*
* This product includes cryptographic software written by Eric Young
* (eay@cryptsoft.com). This product includes software written by Tim
* Hudson (tjh@cryptsoft.com).
*
*/

Original SSLeay License

/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
* All rights reserved.
*
* This package is an SSL implementation written
* by Eric Young (eay@cryptsoft.com).
* The implementation was written so as to conform with Netscapes SSL.

*
* This library is free for commercial and non-commercial use as long as
* the following conditions are aheared to. The following conditions
* apply to all code found in this distribution, be it the RC4, RSA,
* lhash, DES, etc., code; not just the SSL code. The SSL documentation
* included with this distribution is covered by the same copyright terms
* except that the holder is Tim Hudson (tjh@cryptsoft.com).
*
* Copyright remains Eric Young's, and as such any Copyright notices in
* the code are not to be removed.
* If this package is used in a product, Eric Young should be given attribution
* as the author of the parts of the library used.
* This can be in the form of a textual message at program startup or
* in documentation (online or textual) provided with the package.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
* 1. Redistributions of source code must retain the copyright
*   notice, this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright
*   notice, this list of conditions and the following disclaimer in the
*   documentation and/or other materials provided with the distribution.
* 3. All advertising materials mentioning features or use of this software
*   must display the following acknowledgement:
*   "This product includes cryptographic software written by
*   Eric Young (eay@cryptsoft.com)"
*   The word 'cryptographic' can be left out if the rouines from the library
*   being used are not cryptographic related :-).
* 4. If you include any Windows specific code (or a derivative thereof) from
*   the apps directory (application code) you must include an acknowledgement:
*   "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
*
* THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
* ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
* OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
* LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
* OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
*
* The licence and distribution terms for any publically available version or
* derivative of this code cannot be changed. i.e. this code cannot simply be
* copied and put under another distribution licence

* [including the GNU Public Licence.]
*/

==
=========
=== The following part contains the license for the Cyrus SASL 2.x library ===
=== used for optional SASL support ===
==
=========

/* CMU libsasl
* Tim Martin
* Rob Earhart
* Rob Siemborski
*/
/*
* Copyright (c) 1998-2003 Carnegie Mellon University. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
*
* 1. Redistributions of source code must retain the above copyright
*   notice, this list of conditions and the following disclaimer.
*
* 2. Redistributions in binary form must reproduce the above copyright
*   notice, this list of conditions and the following disclaimer in
*   the documentation and/or other materials provided with the
*   distribution.
*
* 3. The name "Carnegie Mellon University" must not be used to
*   endorse or promote products derived from this software without
*   prior written permission. For permission or any other legal
*   details, please contact
*   Office of Technology Transfer
*   Carnegie Mellon University
*   5000 Forbes Avenue
*   Pittsburgh, PA 15213-3890
*   (412) 268-4387, fax: (412) 268-7395
*   tech-transfer@andrew.cmu.edu
*
* 4. Redistributions of any form whatsoever must retain the following
*   acknowledgment:
*   "This product includes software developed by Computing Services
*   at Carnegie Mellon University (http://www.cmu.edu/computing/)."
*
* CARNEGIE MELLON UNIVERSITY DISCLAIMS ALL WARRANTIES WITH REGARD TO
* THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY

* AND FITNESS, IN NO EVENT SHALL CARNEGIE MELLON UNIVERSITY BE LIABLE
* FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
* WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN
* AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING
* OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
*/

http://www.opensource.org/licenses/bsd-license.php

Redistribution and use in source and binary forms, with or
without modification, are permitted provided that the following
conditions are met:

Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer
in the documentation and/or other materials provided with
the distribution.

Neither the name of JLine nor the names of its contributors
may be used to endorse or promote products derived from this
software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING,
BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY,
OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
OF THE POSSIBILITY OF SUCH DAMAGE.

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to

permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a

cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,

any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following
boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.102 java-classmate 1.3.4

1.102.1 Available under license :

Java ClassMate library was originally written by Tatu Saloranta (tatu.saloranta@iki.fi)

Other developers who have contributed code are:

* Brian Langel
This copy of Java ClassMate library is licensed under Apache (Software) License,
version 2.0 ("the License").
See the License for details about distribution rights, and the specific rights regarding derivate works.

You may obtain a copy of the License at:

http://www.apache.org/licenses/LICENSE-2.0

1.103 jackson-datatype-joda jackson-datatype-joda-2.10.3

1.103.1 Available under license :

This copy of Jackson JSON processor streaming parser/generator is licensed under the
Apache (Software) License, version 2.0 ("the License").
See the License for details about distribution rights, and the
specific rights regarding derivate works.

You may obtain a copy of the License at:

http://www.apache.org/licenses/LICENSE-2.0

1.104 fastutil 8.2.3

1.104.1 Available under license :

No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import java.util.Iterator;
import java.util.ListIterator;
import java.util.NoSuchElementException;
import java.util.Objects;

/** A class providing static methods and objects that do useful things with type-specific iterators.
*
* @see Iterator
*/

public final class ITERATORS {

 private ITERATORS() {}
```

```
/** A class returning no elements and a type-specific iterator interface.
 *
 * <p>This class may be useful to implement your own in case you subclass
 * a type-specific iterator.
 */

public static class EmptyIterator KEY_GENERIC implements KEY_LIST_ITERATOR KEY_GENERIC,
java.io.Serializable, Cloneable {

 private static final long serialVersionUID = -7046029254386353129L;

 protected EmptyIterator() {}

 @Override
 public boolean hasNext() { return false; }

 @Override
 public boolean hasPrevious() { return false; }

 @Override
 public KEY_GENERIC_TYPE NEXT_KEY() { throw new NoSuchElementException(); }

 @Override
 public KEY_GENERIC_TYPE PREV_KEY() { throw new NoSuchElementException(); }

 @Override
 public int nextIndex() { return 0; }

 @Override
 public int previousIndex() { return -1; }

 @Override
 public int skip(int n) { return 0; };

 @Override
 public int back(int n) { return 0; };

 @Override
 public Object clone() { return EMPTY_ITERATOR; }

 private Object readResolve() { return EMPTY_ITERATOR; }
}

/** An empty iterator. It is serializable and cloneable.
 *
 * <p>The class of this objects represent an abstract empty iterator
 * that can iterate as a type-specific (list) iterator.
```

```
 */

 SUPPRESS_WARNINGS_KEY_RAWTYPES
 public static final EmptyIterator EMPTY_ITERATOR = new EmptyIterator();

#if KEYS_REFERENCE
 /** Returns an empty iterator. It is serializable and cloneable.
  *
  * <p>The class of the object returned represent an abstract empty iterator
  * that can iterate as a type-specific (list) iterator.
  *
  * <p>This method provides a typesafe access to {@link #EMPTY_ITERATOR}.
  * @return an empty iterator.
  */
 @SuppressWarnings("unchecked")
 public static KEY_GENERIC KEY_ITERATOR KEY_GENERIC emptyIterator() { return EMPTY_ITERATOR; }
#endif


 /** An iterator returning a single element. */

 private static class SingletonIterator KEY_GENERIC implements KEY_LIST_ITERATOR KEY_GENERIC {
  private final KEY_GENERIC_TYPE element;
  private int curr;

  public SingletonIterator(final KEY_GENERIC_TYPE element) {
   this.element = element;
  }

  @Override
  public boolean hasNext() { return curr == 0; }

  @Override
  public boolean hasPrevious() { return curr == 1; }

  @Override
  public KEY_GENERIC_TYPE NEXT_KEY() {
   if (! hasNext()) throw new NoSuchElementException();
   curr = 1;
   return element;
  }

  @Override
  public KEY_GENERIC_TYPE PREV_KEY() {
   if (! hasPrevious()) throw new NoSuchElementException();
   curr = 0;
   return element;
  }
```

```
 @Override
 public int nextIndex() {
  return curr;
 }

 @Override
 public int previousIndex() {
  return curr - 1;
 }
}


/** Returns an immutable iterator that iterates just over the given element.
 *
 * @param element the only element to be returned by a type-specific list iterator.
 * @return an immutable iterator that iterates just over {@code element}.
 */
 public static KEY_GENERIC KEY_LIST_ITERATOR KEY_GENERIC singleton(final KEY_GENERIC_TYPE
element) {
  return new SingletonIterator KEY_GENERIC_DIAMOND(element);
 }


/** A class to wrap arrays in iterators. */

 private static class ArrayIterator KEY_GENERIC implements KEY_LIST_ITERATOR KEY_GENERIC {
  private final KEY_GENERIC_TYPE[] array;
  private final int offset, length;
  private int curr;

  public ArrayIterator(final KEY_GENERIC_TYPE[] array, final int offset, final int length) {
   this.array = array;
   this.offset = offset;
   this.length = length;
  }

  @Override
  public boolean hasNext() { return curr < length; }

  @Override
  public boolean hasPrevious() { return curr > 0; }

  @Override
  public KEY_GENERIC_TYPE NEXT_KEY() {
   if (! hasNext()) throw new NoSuchElementException();
   return array[offset + curr++];
  }
```

```
@Override
public KEY_GENERIC_TYPE PREV_KEY() {
 if (! hasPrevious()) throw new NoSuchElementException();
 return array[offset + --curr];
}

@Override
public int skip(int n) {
 if (n <= length - curr) {
  curr += n;
  return n;
 }
 n = length - curr;
 curr = length;
 return n;
}

@Override
public int back(int n) {
 if (n <= curr) {
  curr -= n;
  return n;
 }
 n = curr;
 curr = 0;
 return n;
}

@Override
public int nextIndex() {
 return curr;
}

@Override
public int previousIndex() {
 return curr - 1;
 }
}


/** Wraps the given part of an array into a type-specific list iterator.
 *
 * <p>The type-specific list iterator returned by this method will iterate
 * {@code length} times, returning consecutive elements of the given
 * array starting from the one with index {@code offset}.
 *
 * @param array an array to wrap into a type-specific list iterator.
```

```
 * @param offset the first element of the array to be returned.
 * @param length the number of elements to return.
 * @return an iterator that will return {@code length} elements of {@code array} starting at position {@code
offset}.
 */
 public static KEY_GENERIC KEY_LIST_ITERATOR KEY_GENERIC wrap(final KEY_GENERIC_TYPE[]
array, final int offset, final int length) {
  ARRAYS.ensureOffsetLength(array, offset, length);
  return new ArrayIterator KEY_GENERIC_DIAMOND(array, offset, length);
 }

 /** Wraps the given array into a type-specific list iterator.
  *
  * <p>The type-specific list iterator returned by this method will return
  * all elements of the given array.
  *
  * @param array an array to wrap into a type-specific list iterator.
  * @return an iterator that will the elements of {@code array}.
  */
 public static KEY_GENERIC KEY_LIST_ITERATOR KEY_GENERIC wrap(final KEY_GENERIC_TYPE[]
array) {
  return new ArrayIterator KEY_GENERIC_DIAMOND(array, 0, array.length);
 }


 /** Unwraps an iterator into an array starting at a given offset for a given number of elements.
  *
  * <p>This method iterates over the given type-specific iterator and stores the elements
  * returned, up to a maximum of {@code length}, in the given array starting at {@code offset}.
  * The number of actually unwrapped elements is returned (it may be less than {@code max} if
  * the iterator emits less than {@code max} elements).
  *
  * @param i a type-specific iterator.
  * @param array an array to contain the output of the iterator.
  * @param offset the first element of the array to be returned.
  * @param max the maximum number of elements to unwrap.
  * @return the number of elements unwrapped.
  */
 public static KEY_GENERIC int unwrap(final STD_KEY_ITERATOR KEY_EXTENDS_GENERIC i, final
KEY_GENERIC_TYPE array[], int offset, final int max) {
  if (max < 0) throw new IllegalArgumentException("The maximum number of elements (" + max + ") is negative");
  if (offset < 0 || offset + max > array.length) throw new IllegalArgumentException();
  int j = max;
  while(j-- != 0 && i.hasNext()) array[offset++] = i.NEXT_KEY();
  return max - j - 1;
 }

 /** Unwraps an iterator into an array.
```

```
 *
 * <p>This method iterates over the given type-specific iterator and stores the
 * elements returned in the given array. The iteration will stop when the
 * iterator has no more elements or when the end of the array has been reached.
 *
 * @param i a type-specific iterator.
 * @param array an array to contain the output of the iterator.
 * @return the number of elements unwrapped.
 */
 public static KEY_GENERIC int unwrap(final STD_KEY_ITERATOR KEY_EXTENDS_GENERIC i, final
KEY_GENERIC_TYPE array[]) {
  return unwrap(i, array, 0, array.length);
 }

 /** Unwraps an iterator, returning an array, with a limit on the number of elements.
 *
 * <p>This method iterates over the given type-specific iterator and returns an array
 * containing the elements returned by the iterator. At most {@code max} elements
 * will be returned.
 *
 * @param i a type-specific iterator.
 * @param max the maximum number of elements to be unwrapped.
 * @return an array containing the elements returned by the iterator (at most {@code max}).
 */
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public static KEY_GENERIC KEY_GENERIC_TYPE[] unwrap(final STD_KEY_ITERATOR
KEY_EXTENDS_GENERIC i, int max) {
  if (max < 0) throw new IllegalArgumentException("The maximum number of elements (" + max + ") is negative");
  KEY_GENERIC_TYPE array[] = KEY_GENERIC_ARRAY_CAST new KEY_TYPE[16];
  int j = 0;

  while(max-- != 0 && i.hasNext()) {
   if (j == array.length) array = ARRAYS.grow(array, j + 1);
   array[j++] = i.NEXT_KEY();
  }

  return ARRAYS.trim(array, j);
 }



 /** Unwraps an iterator, returning an array.
 *
 * <p>This method iterates over the given type-specific iterator and returns an array
 * containing the elements returned by the iterator.
 *
 * @param i a type-specific iterator.
 * @return an array containing the elements returned by the iterator.
 */
```

```
 public static KEY_GENERIC KEY_GENERIC_TYPE[] unwrap(final STD_KEY_ITERATOR
KEY_EXTENDS_GENERIC i) {
  return unwrap(i, Integer.MAX_VALUE);
 }


 /** Unwraps an iterator into a type-specific collection, with a limit on the number of elements.
  *
  * <p>This method iterates over the given type-specific iterator and stores the elements
  * returned, up to a maximum of {@code max}, in the given type-specific collection.
  * The number of actually unwrapped elements is returned (it may be less than {@code max} if
  * the iterator emits less than {@code max} elements).
  *
  * @param i a type-specific iterator.
  * @param c a type-specific collection array to contain the output of the iterator.
  * @param max the maximum number of elements to unwrap.
  * @return the number of elements unwrapped. Note that
  * this is the number of elements returned by the iterator, which is not necessarily the number
  * of elements that have been added to the collection (because of duplicates).
  */
 public static KEY_GENERIC int unwrap(final STD_KEY_ITERATOR KEY_GENERIC i, final COLLECTION
KEY_SUPER_GENERIC c, final int max) {
  if (max < 0) throw new IllegalArgumentException("The maximum number of elements (" + max + ") is negative");
  int j = max;
  while(j-- != 0 && i.hasNext()) c.add(i.NEXT_KEY());
  return max - j - 1;
 }

 /** Unwraps an iterator into a type-specific collection.
  *
  * <p>This method iterates over the given type-specific iterator and stores the
  * elements returned in the given type-specific collection. The returned count on the number
  * unwrapped elements is a long, so that it will work also with very large collections.
  *
  * @param i a type-specific iterator.
  * @param c a type-specific collection to contain the output of the iterator.
  * @return the number of elements unwrapped. Note that
  * this is the number of elements returned by the iterator, which is not necessarily the number
  * of elements that have been added to the collection (because of duplicates).
  */
 public static KEY_GENERIC long unwrap(final STD_KEY_ITERATOR KEY_GENERIC i, final COLLECTION
KEY_SUPER_GENERIC c) {
  long n = 0;
  while(i.hasNext()) {
   c.add(i.NEXT_KEY());
   n++;
  }
```

```
 return n;
 }


 /** Pours an iterator into a type-specific collection, with a limit on the number of elements.
  *
  * <p>This method iterates over the given type-specific iterator and adds
  * the returned elements to the given collection (up to {@code max}).
  *
  * @param i a type-specific iterator.
  * @param s a type-specific collection.
  * @param max the maximum number of elements to be poured.
  * @return the number of elements poured. Note that
  * this is the number of elements returned by the iterator, which is not necessarily the number
  * of elements that have been added to the collection (because of duplicates).
  */

 public static KEY_GENERIC int pour(final STD_KEY_ITERATOR KEY_GENERIC i, final COLLECTION
KEY_SUPER_GENERIC s, final int max) {
  if (max < 0) throw new IllegalArgumentException("The maximum number of elements (" + max + ") is negative");
  int j = max;
  while(j-- != 0 && i.hasNext()) s.add(i.NEXT_KEY());
  return max - j - 1;
 }

 /** Pours an iterator into a type-specific collection.
  *
  * <p>This method iterates over the given type-specific iterator and adds
  * the returned elements to the given collection.
  *
  * @param i a type-specific iterator.
  * @param s a type-specific collection.
  * @return the number of elements poured. Note that
  * this is the number of elements returned by the iterator, which is not necessarily the number
  * of elements that have been added to the collection (because of duplicates).
  */

 public static KEY_GENERIC int pour(final STD_KEY_ITERATOR KEY_GENERIC i, final COLLECTION
KEY_SUPER_GENERIC s) {
  return pour(i, s, Integer.MAX_VALUE);
 }

 /** Pours an iterator, returning a type-specific list, with a limit on the number of elements.
  *
  * <p>This method iterates over the given type-specific iterator and returns
  * a type-specific list containing the returned elements (up to {@code max}). Iteration
  * on the returned list is guaranteed to produce the elements in the same order
  * in which they appeared in the iterator.
```

```
 *
 *
 * @param i a type-specific iterator.
 * @param max the maximum number of elements to be poured.
 * @return a type-specific list containing the returned elements, up to {@code max}.
 */

 public static KEY_GENERIC LIST KEY_GENERIC pour(final STD_KEY_ITERATOR KEY_GENERIC i, int
max) {
 final ARRAY_LIST KEY_GENERIC l = new ARRAY_LIST KEY_GENERIC_DIAMOND();
 pour(i, l, max);
 l.trim();
 return l;
 }

 /** Pours an iterator, returning a type-specific list.
 *
 * <p>This method iterates over the given type-specific iterator and returns
 * a list containing the returned elements. Iteration
 * on the returned list is guaranteed to produce the elements in the same order
 * in which they appeared in the iterator.
 *
 * @param i a type-specific iterator.
 * @return a type-specific list containing the returned elements.
 */

 public static KEY_GENERIC LIST KEY_GENERIC pour(final STD_KEY_ITERATOR KEY_GENERIC i) {
 return pour(i, Integer.MAX_VALUE);
 }

 private static class IteratorWrapper KEY_GENERIC implements KEY_ITERATOR KEY_GENERIC {
 final Iterator<KEY_GENERIC_CLASS> i;

 public IteratorWrapper(final Iterator<KEY_GENERIC_CLASS> i) {
  this.i = i;
 }

 @Override
 public boolean hasNext() { return i.hasNext(); }
 @Override
 public void remove() { i.remove(); }
 @Override
 public KEY_GENERIC_TYPE NEXT_KEY() { return KEY_CLASS2TYPE(i.next()); }
 }

 /** Wraps a standard iterator into a type-specific iterator.
 *
 * <p>This method wraps a standard iterator into a type-specific one which will handle the
```

```
 * type conversions for you. Of course, any attempt to wrap an iterator returning the
 * instances of the wrong class will generate a {@link ClassCastException}. The
 * returned iterator is backed by {@code i}: changes to one of the iterators
 * will affect the other, too.
 *
 * <p>If {@code i} is already type-specific, it will returned and no new object
 * will be generated.
 *
 * @param i an iterator.
 * @return a type-specific iterator  backed by {@code i}.
 */
#if KEYS_PRIMITIVE
 @SuppressWarnings({"unchecked","rawtypes"})
#endif
 public static KEY_GENERIC KEY_ITERATOR KEY_GENERIC AS_KEY_ITERATOR(final Iterator
KEY_GENERIC i) {
  if (i instanceof KEY_ITERATOR) return (KEY_ITERATOR KEY_GENERIC)i;
  return new IteratorWrapper KEY_GENERIC_DIAMOND(i);
 }


 private static class ListIteratorWrapper KEY_GENERIC implements KEY_LIST_ITERATOR KEY_GENERIC {
  final ListIterator<KEY_GENERIC_CLASS> i;

  public ListIteratorWrapper(final ListIterator<KEY_GENERIC_CLASS> i) {
   this.i = i;
  }

  @Override
  public boolean hasNext() { return i.hasNext(); }

  @Override
  public boolean hasPrevious() { return i.hasPrevious(); }

  @Override
  public int nextIndex() { return i.nextIndex(); }

  @Override
  public int previousIndex() { return i.previousIndex(); }

  @Override
  public void set(KEY_GENERIC_TYPE k) { i.set(KEY2OBJ(k)); }

  @Override
  public void add(KEY_GENERIC_TYPE k) { i.add(KEY2OBJ(k)); }

  @Override
  public void remove() { i.remove(); }
```

```
 @Override
 public KEY_GENERIC_TYPE NEXT_KEY() { return KEY_CLASS2TYPE(i.next()); }

 @Override
 public KEY_GENERIC_TYPE PREV_KEY() { return KEY_CLASS2TYPE(i.previous()); }
}

/** Wraps a standard list iterator into a type-specific list iterator.
 *
 * <p>This method wraps a standard list iterator into a type-specific one
 * which will handle the type conversions for you. Of course, any attempt
 * to wrap an iterator returning the instances of the wrong class will
 * generate a {@link ClassCastException}. The
 * returned iterator is backed by {@code i}: changes to one of the iterators
 * will affect the other, too.
 *
 * <p>If {@code i} is already type-specific, it will returned and no new object
 * will be generated.
 *
 * @param i a list iterator.
 * @return a type-specific list iterator backed by {@code i}.
 */
#if KEYS_PRIMITIVE
 @SuppressWarnings({"unchecked","rawtypes"})
#endif
 public static KEY_GENERIC KEY_LIST_ITERATOR KEY_GENERIC AS_KEY_ITERATOR(final ListIterator
KEY_GENERIC i) {
  if (i instanceof KEY_LIST_ITERATOR) return (KEY_LIST_ITERATOR KEY_GENERIC)i;
  return new ListIteratorWrapper KEY_GENERIC_DIAMOND(i);
 }

#ifdef JDK_PRIMITIVE_PREDICATE
 public static boolean any(final KEY_ITERATOR iterator, final JDK_PRIMITIVE_PREDICATE predicate) {
#else
 public static KEY_GENERIC boolean any(final KEY_ITERATOR KEY_GENERIC iterator, final
java.util.function.Predicate<? super KEY_GENERIC_CLASS> predicate) {
#endif
  return indexOf(iterator, predicate) != -1;
 }

#ifdef JDK_PRIMITIVE_PREDICATE
 public static boolean all(final KEY_ITERATOR iterator, final JDK_PRIMITIVE_PREDICATE predicate) {
#else
 public static KEY_GENERIC boolean all(final KEY_ITERATOR KEY_GENERIC iterator, final
java.util.function.Predicate<? super KEY_GENERIC_CLASS> predicate) {
#endif
  Objects.requireNonNull(predicate);
```

```
  do {
   if (!iterator.hasNext()) return true;
#if KEY_CLASS_Boolean
  } while (predicate.test(Boolean.valueOf(iterator.nextBoolean())));
#else
  } while (predicate.test(iterator.NEXT_KEY()));
#endif
  return false;
 }

#ifdef JDK_PRIMITIVE_PREDICATE
 public static int indexOf(final KEY_ITERATOR iterator, final JDK_PRIMITIVE_PREDICATE predicate) {
#else
 public static KEY_GENERIC int indexOf(final KEY_ITERATOR KEY_GENERIC iterator, final
java.util.function.Predicate<? super KEY_GENERIC_CLASS> predicate) {
#endif
  Objects.requireNonNull(predicate);

  for (int i = 0; iterator.hasNext(); ++i) {
#if KEY_CLASS_Boolean
   if (predicate.test(Boolean.valueOf(iterator.nextBoolean()))) return i;
#else
   if (predicate.test(iterator.NEXT_KEY())) return i;
#endif
  }

  return -1;
 }

#if KEY_CLASS_Integer || KEY_CLASS_Byte || KEY_CLASS_Short || KEY_CLASS_Character ||
KEY_CLASS_Long

#if KEY_CLASS_Long
 private static class IntervalIterator implements KEY_BIDI_ITERATOR {
#else
 private static class IntervalIterator implements KEY_LIST_ITERATOR {
#endif
  private final KEY_TYPE from, to;
  KEY_TYPE curr;

  public IntervalIterator(final KEY_TYPE from, final KEY_TYPE to) {
   this.from = this.curr = from;
   this.to = to;
  }

  @Override
  public boolean hasNext() { return curr < to; }
```

```
  @Override
  public boolean hasPrevious() { return curr > from; }

  @Override
  public KEY_TYPE NEXT_KEY() {
   if (! hasNext()) throw new NoSuchElementException();
   return curr++;
  }

  @Override
  public KEY_TYPE PREV_KEY() {
   if (! hasPrevious()) throw new NoSuchElementException();
   return --curr;
  }

#if ! KEY_CLASS_Long
  @Override
  public int nextIndex() { return curr - from; }

  @Override
  public int previousIndex() { return curr - from - 1; }
#endif

  @Override
  public int skip(int n) {
   if (curr + n <= to) {
    curr += n;
    return n;
   }
#if ! KEY_CLASS_Long
   n = to - curr;
#else
   n = (int)(to - curr);
#endif
   curr = to;
   return n;
  }

  @Override
  public int back(int n) {
   if (curr - n >= from) {
    curr -= n;
    return n;
   }
#if ! KEY_CLASS_Long
   n = curr - from ;
#else
```

```
   n = (int)(curr - from);
#endif
   curr = from;
   return n;
  }
 }

#if KEY_CLASS_Long
 /** Creates a type-specific bidirectional iterator over an interval.
  *
  * <p>The type-specific bidirectional iterator returned by this method will return the
  * elements {@code from}, {@code from+1},&hellip;, {@code to-1}.
  *
  * <p>Note that all other type-specific interval iterator are <em>list</em>
  * iterators. Of course, this is not possible with longs as the index
  * returned by {@link java.util.ListIterator#nextIndex() nextIndex()}/{@link
  * java.util.ListIterator#previousIndex() previousIndex()} would exceed an integer.
  *
  * @param from the starting element (inclusive).
  * @param to the ending element (exclusive).
  * @return a type-specific bidirectional iterator enumerating the elements from {@code from} to {@code to}.
  */
 public static KEY_BIDI_ITERATOR fromTo(final KEY_TYPE from, final KEY_TYPE to) {
  return new IntervalIterator(from, to);
 }
#else

 /** Creates a type-specific list iterator over an interval.
  *
  * <p>The type-specific list iterator returned by this method will return the
  * elements {@code from}, {@code from+1},&hellip;, {@code to-1}.
  *
  * @param from the starting element (inclusive).
  * @param to the ending element (exclusive).
  * @return a type-specific list iterator enumerating the elements from {@code from} to {@code to}.
  */
 public static KEY_LIST_ITERATOR fromTo(final KEY_TYPE from, final KEY_TYPE to) {
  return new IntervalIterator(from, to);
 }

#endif

#endif

 private static class IteratorConcatenator KEY_GENERIC implements KEY_ITERATOR KEY_GENERIC {
  final KEY_ITERATOR KEY_EXTENDS_GENERIC a[];
  int offset, length, lastOffset = -1;
```

```
public IteratorConcatenator(final KEY_ITERATOR KEY_EXTENDS_GENERIC a[], int offset, int length) {
 this.a = a;
 this.offset = offset;
 this.length = length;
 advance();
}

private void advance() {
 while(length != 0) {
  if (a[offset].hasNext()) break;
  length--;
  offset++;
 }

 return;
}

@Override
public boolean hasNext() {
 return length > 0;
}

@Override
public KEY_GENERIC_TYPE NEXT_KEY() {
 if (! hasNext()) throw new NoSuchElementException();
 KEY_GENERIC_TYPE next = a[lastOffset = offset].NEXT_KEY();
 advance();
 return next;
}

@Override
public void remove() {
 if (lastOffset == -1) throw new IllegalStateException();
 a[lastOffset].remove();
}

@Override
public int skip(int n) {
 lastOffset = -1;

 int skipped = 0;

 while(skipped < n && length != 0) {
  skipped += a[offset].skip(n - skipped);
  if (a[offset].hasNext()) break;
  length--;
  offset++;
 }
```

```
  return skipped;
 }
}


/** Concatenates all iterators contained in an array.
 *
 * <p>This method returns an iterator that will enumerate in order the elements returned
 * by all iterators contained in the given array.
 *
 * @param a an array of iterators.
 * @return an iterator obtained by concatenation.
 */

public static KEY_GENERIC KEY_ITERATOR KEY_GENERIC concat(final KEY_ITERATOR
KEY_EXTENDS_GENERIC a[]) {
 return concat(a, 0, a.length);
}


/** Concatenates a sequence of iterators contained in an array.
 *
 * <p>This method returns an iterator that will enumerate in order the elements returned
 * by {@code a[offset]}, then those returned
 * by {@code a[offset + 1]}, and so on up to
 * {@code a[offset + length - 1]}.
 *
 * @param a an array of iterators.
 * @param offset the index of the first iterator to concatenate.
 * @param length the number of iterators to concatenate.
 * @return an iterator obtained by concatenation of {@code length} elements of {@code a} starting at {@code
offset}.
 */

public static KEY_GENERIC KEY_ITERATOR KEY_GENERIC concat(final KEY_ITERATOR
KEY_EXTENDS_GENERIC a[], final int offset, final int length) {
 return new IteratorConcatenator KEY_GENERIC_DIAMOND(a, offset, length);
}


 /** An unmodifiable wrapper class for iterators. */


public static class UnmodifiableIterator KEY_GENERIC implements KEY_ITERATOR KEY_GENERIC {
 protected final KEY_ITERATOR KEY_GENERIC i;

 public UnmodifiableIterator(final KEY_ITERATOR KEY_GENERIC i) {
```

```
 this.i = i;
 }

 @Override
 public boolean hasNext() { return i.hasNext(); }

 @Override
 public KEY_GENERIC_TYPE NEXT_KEY() { return i.NEXT_KEY(); }
 }


 /** Returns an unmodifiable iterator backed by the specified iterator.
  *
  * @param i the iterator to be wrapped in an unmodifiable iterator.
  * @return an unmodifiable view of the specified iterator.
  */
 public static KEY_GENERIC KEY_ITERATOR KEY_GENERIC unmodifiable(final KEY_ITERATOR
KEY_GENERIC i) { return new UnmodifiableIterator KEY_GENERIC_DIAMOND(i); }



  /** An unmodifiable wrapper class for bidirectional iterators. */

 public static class UnmodifiableBidirectionalIterator KEY_GENERIC implements KEY_BIDI_ITERATOR
KEY_GENERIC {
 protected final KEY_BIDI_ITERATOR KEY_GENERIC i;

 public UnmodifiableBidirectionalIterator(final KEY_BIDI_ITERATOR KEY_GENERIC i) {
  this.i = i;
 }

 @Override
 public boolean hasNext() { return i.hasNext(); }

 @Override
 public boolean hasPrevious() { return i.hasPrevious(); }

 @Override
 public KEY_GENERIC_TYPE NEXT_KEY() { return i.NEXT_KEY(); }

 @Override
 public KEY_GENERIC_TYPE PREV_KEY() { return i.PREV_KEY(); }
 }


 /** Returns an unmodifiable bidirectional iterator backed by the specified bidirectional iterator.
  *
  * @param i the bidirectional iterator to be wrapped in an unmodifiable bidirectional iterator.
```

```
 * @return an unmodifiable view of the specified bidirectional iterator.
 */
 public static KEY_GENERIC KEY_BIDI_ITERATOR KEY_GENERIC unmodifiable(final
KEY_BIDI_ITERATOR KEY_GENERIC i) { return new UnmodifiableBidirectionalIterator
KEY_GENERIC_DIAMOND(i); }


 /** An unmodifiable wrapper class for list iterators. */

 public static class UnmodifiableListIterator KEY_GENERIC implements KEY_LIST_ITERATOR
KEY_GENERIC {
 protected final KEY_LIST_ITERATOR KEY_GENERIC i;

 public UnmodifiableListIterator(final KEY_LIST_ITERATOR KEY_GENERIC i) {
  this.i = i;
 }

 @Override
 public boolean hasNext() { return i.hasNext(); }

 @Override
 public boolean hasPrevious() { return i.hasPrevious(); }

 @Override
 public KEY_GENERIC_TYPE NEXT_KEY() { return i.NEXT_KEY(); }

 @Override
 public KEY_GENERIC_TYPE PREV_KEY() { return i.PREV_KEY(); }

 @Override
 public int nextIndex() { return i.nextIndex(); }

 @Override
 public int previousIndex() { return i.previousIndex(); }
}

 /** Returns an unmodifiable list iterator backed by the specified list iterator.
  *
  * @param i the list iterator to be wrapped in an unmodifiable list iterator.
  * @return an unmodifiable view of the specified list iterator.
  */
 public static KEY_GENERIC KEY_LIST_ITERATOR KEY_GENERIC unmodifiable(final
KEY_LIST_ITERATOR KEY_GENERIC i) { return new UnmodifiableListIterator
KEY_GENERIC_DIAMOND(i); }

#if SMALL_TYPES && (KEY_CLASS_Short || KEY_CLASS_Integer || KEY_CLASS_Long ||
KEY_CLASS_Float || KEY_CLASS_Double)
```

```
 /** A wrapper promoting the results of a ByteIterator. */

protected static class ByteIteratorWrapper implements KEY_ITERATOR {
 final it.unimi.dsi.fastutil.bytes.ByteIterator iterator;

 public ByteIteratorWrapper(final it.unimi.dsi.fastutil.bytes.ByteIterator iterator) {
  this.iterator = iterator;
 }

 @Override
 public boolean hasNext() { return iterator.hasNext(); }

 @Deprecated
 @Override
 public KEY_GENERIC_CLASS next() { return KEY_GENERIC_CLASS.valueOf(iterator.nextByte()); }

 @Override
 public KEY_TYPE NEXT_KEY() { return iterator.nextByte(); }

 @Override
 public void remove() { iterator.remove(); }

 @Override
 public int skip(final int n) { return iterator.skip(n); }
}

 /** Returns an iterator backed by the specified byte iterator.
  * @param iterator a byte iterator.
  * @return an iterator backed by the specified byte iterator.
  */
 public static KEY_ITERATOR wrap(final it.unimi.dsi.fastutil.bytes.ByteIterator iterator) {
  return new ByteIteratorWrapper(iterator);
 }
#endif

#if SMALL_TYPES && (KEY_CLASS_Integer || KEY_CLASS_Long || KEY_CLASS_Float ||
KEY_CLASS_Double)

 /** A wrapper promoting the results of a ShortIterator. */

 protected static class ShortIteratorWrapper implements KEY_ITERATOR {
  final it.unimi.dsi.fastutil.shorts.ShortIterator iterator;

  public ShortIteratorWrapper(final it.unimi.dsi.fastutil.shorts.ShortIterator iterator) {
   this.iterator = iterator;
  }

  @Override
```

```
 public boolean hasNext() { return iterator.hasNext(); }

 @Deprecated
 @Override
 public KEY_GENERIC_CLASS next() { return KEY_GENERIC_CLASS.valueOf(iterator.nextShort()); }

 @Override
 public KEY_TYPE NEXT_KEY() { return iterator.nextShort(); }

 @Override
 public void remove() { iterator.remove(); }

 @Override
 public int skip(final int n) { return iterator.skip(n); }
}

/** Returns an iterator backed by the specified short iterator.
 * @param iterator a short iterator.
 * @return an iterator backed by the specified short iterator.
 */
public static KEY_ITERATOR wrap(final it.unimi.dsi.fastutil.shorts.ShortIterator iterator) {
 return new ShortIteratorWrapper(iterator);
}

#endif

#if KEY_CLASS_Long || KEY_CLASS_Double

 /** A wrapper promoting the results of an IntIterator. */

protected static class IntIteratorWrapper implements KEY_ITERATOR {
 final it.unimi.dsi.fastutil.ints.IntIterator iterator;

 public IntIteratorWrapper(final it.unimi.dsi.fastutil.ints.IntIterator iterator) {
  this.iterator = iterator;
 }

 @Override
 public boolean hasNext() { return iterator.hasNext(); }

 @Deprecated
 @Override
 public KEY_GENERIC_CLASS next() { return KEY_GENERIC_CLASS.valueOf(iterator.nextInt()); }

 @Override
 public KEY_TYPE NEXT_KEY() { return iterator.nextInt(); }

 @Override
```

```
 public void remove() { iterator.remove(); }

 @Override
 public int skip(final int n) { return iterator.skip(n); }
}

/** Returns an iterator backed by the specified integer iterator.
 * @param iterator an integer iterator.
 * @return an iterator backed by the specified integer iterator.
 */

public static KEY_ITERATOR wrap(final it.unimi.dsi.fastutil.ints.IntIterator iterator) {
 return new IntIteratorWrapper(iterator);
}

#endif

#if SMALL_TYPES && KEY_CLASS_Double

 /** A wrapper promoting the results of a FloatIterator. */

protected static class FloatIteratorWrapper implements KEY_ITERATOR {
 final it.unimi.dsi.fastutil.floats.FloatIterator iterator;

 public FloatIteratorWrapper(final it.unimi.dsi.fastutil.floats.FloatIterator iterator) {
  this.iterator = iterator;
 }

 @Override
 public boolean hasNext() { return iterator.hasNext(); }

 @Deprecated
 @Override
 public KEY_GENERIC_CLASS next() { return KEY_GENERIC_CLASS.valueOf(iterator.nextFloat()); }

 @Override
 public KEY_TYPE NEXT_KEY() { return iterator.nextFloat(); }

 @Override
 public void remove() { iterator.remove(); }

 @Override
 public int skip(final int n) { return iterator.skip(n); }
}

/** Returns an iterator backed by the specified float iterator.
 * @param iterator a float iterator.
 * @return an iterator backed by the specified float iterator.
```

```
 */
 public static KEY_ITERATOR wrap(final it.unimi.dsi.fastutil.floats.FloatIterator iterator) {
  return new FloatIteratorWrapper(iterator);
 }
#endif
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/Iterators.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

#if KEYS_PRIMITIVE
import java.util.Objects;
#endif
import java.util.function.Consumer;

/** A type-specific {@link Consumer}; provides methods to consume a primitive type both as object
* and as primitive.
*
* @see Consumer
* @since 8.0.0
*/

@FunctionalInterface
#ifdef JDK_PRIMITIVE_KEY_CONSUMER
public interface KEY_CONSUMER KEY_GENERIC extends Consumer<KEY_GENERIC_CLASS>,
JDK_PRIMITIVE_KEY_CONSUMER {
#else
```

```
public interface KEY_CONSUMER KEY_GENERIC extends Consumer<KEY_GENERIC_CLASS> {
#endif

#if KEYS_PRIMITIVE
#if !defined JDK_PRIMITIVE_KEY_CONSUMER || KEY_WIDENED
 void accept(KEY_TYPE t);
#endif

#if KEY_WIDENED
 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 default void accept(final KEY_TYPE_WIDENED t) {
  accept(KEY_NARROWING(t));
 }
#endif

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 default void accept(final KEY_CLASS t) {
  this.accept(t.KEY_VALUE());
 }

#if !defined JDK_PRIMITIVE_KEY_CONSUMER || KEY_WIDENED
 default KEY_CONSUMER andThen(final KEY_CONSUMER after) {
  Objects.requireNonNull(after);
  return (KEY_TYPE t) -> { accept(t); after.accept(t); };
 }
#endif

#ifdef JDK_PRIMITIVE_KEY_CONSUMER
#if KEY_WIDENED
 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
#endif
 @Override
 default KEY_CONSUMER andThen(final JDK_PRIMITIVE_KEY_CONSUMER after) {
  Objects.requireNonNull(after);
  return (KEY_TYPE t) -> { accept(t); after.accept(t); };
 }
#endif

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
```

```
 @Deprecated
 @Override
 default Consumer<KEY_GENERIC_CLASS> andThen(final Consumer<? super KEY_GENERIC_CLASS> after)
{
  return Consumer.super.andThen(after);
 }
#endif
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/Consumer.drv
No license file was found, but licenses were detected in source scan.


```
package PACKAGE;

import VALUE_PACKAGE.VALUE_COLLECTION;
import VALUE_PACKAGE.VALUE_ABSTRACT_COLLECTION;
import VALUE_PACKAGE.VALUE_ITERATOR;
import it.unimi.dsi.fastutil.objects.ObjectBidirectionalIterator;

#if KEYS_REFERENCE
import java.util.Comparator;
#endif

/** An abstract class providing basic methods for sorted maps implementing a type-specific interface. */

public abstract class ABSTRACT_SORTED_MAP KEY_VALUE_GENERIC extends ABSTRACT_MAP KEY_VALUE_GENERIC implements SORTED_MAP KEY_VALUE_GENERIC {

 private static final long serialVersionUID = -1773560792952436569L;
```

```
 protected ABSTRACT_SORTED_MAP() {}

 /** {@inheritDoc}
  *
  * <p>The view is backed by the sorted set returned by {@link java.util.Map#entrySet()}. Note that
  * <em>no attempt is made at caching the result of this method</em>, as this would
  * require adding some attributes that lightweight implementations would
  * not need. Subclasses may easily override this policy by calling
  * this method and caching the result, but implementors are encouraged to
  * write more efficient ad-hoc implementations.
  *
  * @return a sorted set view of the keys of this map; it may be safely cast to a type-specific interface.
  */
 @Override
 public SORTED_SET KEY_GENERIC keySet() {
  return new KeySet();
 }

 /** A wrapper exhibiting the keys of a map. */

 protected class KeySet extends ABSTRACT_SORTED_SET KEY_GENERIC {
  @Override
  public boolean contains(final KEY_TYPE k) { return containsKey(k); }

  @Override
  public int size() { return ABSTRACT_SORTED_MAP.this.size(); }

  @Override
  public void clear() { ABSTRACT_SORTED_MAP.this.clear(); }

  @Override
  public KEY_COMPARATOR KEY_SUPER_GENERIC comparator() { return
 ABSTRACT_SORTED_MAP.this.comparator(); }

  @Override
  public KEY_GENERIC_TYPE FIRST() { return FIRST_KEY(); }

  @Override
  public KEY_GENERIC_TYPE LAST() { return LAST_KEY(); }

  @Override
  public SORTED_SET KEY_GENERIC headSet(final KEY_GENERIC_TYPE to) { return headMap(to).keySet(); }

  @Override
  public SORTED_SET KEY_GENERIC tailSet(final KEY_GENERIC_TYPE from) { return
 tailMap(from).keySet(); }
```

```
 @Override
 public SORTED_SET KEY_GENERIC subSet(final KEY_GENERIC_TYPE from, final KEY_GENERIC_TYPE
to) { return subMap(from, to).keySet(); }

 @Override
 public KEY_BIDI_ITERATOR KEY_GENERIC iterator(final KEY_GENERIC_TYPE from) { return new
KeySetIterator KEY_VALUE_GENERIC_DIAMOND(ENTRYSET().iterator(new BasicEntry
KEY_VALUE_GENERIC_DIAMOND(from, VALUE_NULL))); }

 @Override
 public KEY_BIDI_ITERATOR KEY_GENERIC iterator() { return new KeySetIterator
KEY_VALUE_GENERIC_DIAMOND(SORTED_MAPS.fastIterator(ABSTRACT_SORTED_MAP.this)); }
 }

 /** A wrapper exhibiting a map iterator as an iterator on keys.
  *
  * <p>To provide an iterator on keys, just create an instance of this
  * class using the corresponding iterator on entries.
  */

 protected static class KeySetIterator KEY_VALUE_GENERIC implements KEY_BIDI_ITERATOR
KEY_GENERIC {
 protected final ObjectBidirectionalIterator<MAP.Entry KEY_VALUE_GENERIC> i;

 public KeySetIterator(ObjectBidirectionalIterator<MAP.Entry KEY_VALUE_GENERIC> i) {
  this.i = i;
 }

 @Override
 public KEY_GENERIC_TYPE NEXT_KEY() { return i.next().ENTRY_GET_KEY(); };

 @Override
 public KEY_GENERIC_TYPE PREV_KEY() { return i.previous().ENTRY_GET_KEY(); };

 @Override
 public boolean hasNext() { return i.hasNext(); }

 @Override
 public boolean hasPrevious() { return i.hasPrevious(); }
 }

 /** {@inheritDoc}
  *
  * <p>The view is backed by the sorted set returned by {@link java.util.Map#entrySet()}. Note that
  * <em>no attempt is made at caching the result of this method</em>, as this would
  * require adding some attributes that lightweight implementations would
  * not need. Subclasses may easily override this policy by calling
  * this method and caching the result, but implementors are encouraged to
```

```
 * write more efficient ad-hoc implementations.
 *
 * @return a type-specific collection view of the values contained in this map.
 */
@Override
public VALUE_COLLECTION VALUE_GENERIC values() {
 return new ValuesCollection();
}

/** A wrapper exhibiting the values of a map. */
protected class ValuesCollection extends VALUE_ABSTRACT_COLLECTION VALUE_GENERIC {
 @Override
 public VALUE_ITERATOR VALUE_GENERIC iterator() { return new ValuesIterator
KEY_VALUE_GENERIC_DIAMOND(SORTED_MAPS.fastIterator(ABSTRACT_SORTED_MAP.this)); }

 @Override
 public boolean contains(final VALUE_TYPE k) { return containsValue(k); }

 @Override
 public int size() { return ABSTRACT_SORTED_MAP.this.size(); }

 @Override
 public void clear() { ABSTRACT_SORTED_MAP.this.clear(); }
}

/** A wrapper exhibiting a map iterator as an iterator on values.
 *
 * <p>To provide an iterator on values, just create an instance of this
 * class using the corresponding iterator on entries.
 */

protected static class ValuesIterator KEY_VALUE_GENERIC implements VALUE_ITERATOR
VALUE_GENERIC {
 protected final ObjectBidirectionalIterator<MAP.Entry KEY_VALUE_GENERIC> i;

 public ValuesIterator(ObjectBidirectionalIterator<MAP.Entry KEY_VALUE_GENERIC> i) {
  this.i = i;
 }

 @Override
 public VALUE_GENERIC_TYPE NEXT_VALUE() { return i.next().ENTRY_GET_VALUE(); };

 @Override
 public boolean hasNext() { return i.hasNext(); }
}
}
```

Found in path(s):

* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/AbstractSortedMap.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2003-2017 Paolo Boldi and Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

#if KEY_CLASS_Object
import java.util.Arrays;
import java.util.Comparator;

import it.unimi.dsi.fastutil.PriorityQueue;
#else
import java.util.Iterator;
#endif

import java.util.Collection;
import java.util.NoSuchElementException;


/** A type-specific heap-based priority queue.
*
* <p>Instances of this class represent a priority queue using a heap. The heap is enlarged as needed, but
* it is never shrunk. Use the {@link #trim()} method to reduce its size, if necessary.
*/

public class HEAP_PRIORITY_QUEUE KEY_GENERIC implements PRIORITY_QUEUE KEY_GENERIC, java.io.Serializable {
 private static final long serialVersionUID = 1L;

 /** The heap array. */
 SUPPRESS_WARNINGS_KEY_UNCHECKED
```

```
 protected transient KEY_GENERIC_TYPE[] heap = KEY_GENERIC_ARRAY_CAST
ARRAYS.EMPTY_ARRAY;

 /** The number of elements in this queue. */
 protected int size;

 /** The type-specific comparator used in this queue. */
 protected KEY_COMPARATOR KEY_SUPER_GENERIC c;

 /** Creates a new empty queue with a given capacity and comparator.
  *
  * @param capacity the initial capacity of this queue.
  * @param c the comparator used in this queue, or {@code null} for the natural order.
  */
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public HEAP_PRIORITY_QUEUE(int capacity, KEY_COMPARATOR KEY_SUPER_GENERIC c) {
  if (capacity > 0) this.heap = KEY_GENERIC_ARRAY_CAST new KEY_TYPE[capacity];
  this.c = c;
 }

 /** Creates a new empty queue with a given capacity and using the natural order.
  *
  * @param capacity the initial capacity of this queue.
  */
 public HEAP_PRIORITY_QUEUE(int capacity) {
  this(capacity, null);
 }

 /** Creates a new empty queue with a given comparator.
  *
  * @param c the comparator used in this queue, or {@code null} for the natural order.
  */
 public HEAP_PRIORITY_QUEUE(KEY_COMPARATOR KEY_SUPER_GENERIC c) {
  this(0, c);
 }

 /** Creates a new empty queue using the natural order.
  */
 public HEAP_PRIORITY_QUEUE() {
  this(0, null);
 }

 /** Wraps a given array in a queue using a given comparator.
  *
  * <p>The queue returned by this method will be backed by the given array.
  * The first {@code size} element of the array will be rearranged so to form a heap (this is
  * more efficient than enqueing the elements of {@code a} one by one).
  *
```

```
 * @param a an array.
 * @param size the number of elements to be included in the queue.
 * @param c the comparator used in this queue, or {@code null} for the natural order.
 */
 public HEAP_PRIORITY_QUEUE(final KEY_GENERIC_TYPE[] a, int size, final KEY_COMPARATOR
KEY_SUPER_GENERIC c) {
 this(c);
 this.heap = a;
 this.size = size;
 HEAPS.makeHeap(a, size, c);
 }


 /** Wraps a given array in a queue using a given comparator.
 *
 * <p>The queue returned by this method will be backed by the given array.
 * The elements of the array will be rearranged so to form a heap (this is
 * more efficient than enqueing the elements of {@code a} one by one).
 *
 * @param a an array.
 * @param c the comparator used in this queue, or {@code null} for the natural order.
 */
 public HEAP_PRIORITY_QUEUE(final KEY_GENERIC_TYPE[] a, final KEY_COMPARATOR
KEY_SUPER_GENERIC c) {
 this(a, a.length, c);
 }

 /** Wraps a given array in a queue using the natural order.
 *
 * <p>The queue returned by this method will be backed by the given array.
 * The first {@code size} element of the array will be rearranged so to form a heap (this is
 * more efficient than enqueing the elements of {@code a} one by one).
 *
 * @param a an array.
 * @param size the number of elements to be included in the queue.
 */
 public HEAP_PRIORITY_QUEUE(final KEY_GENERIC_TYPE[] a, int size) {
 this(a, size, null);
 }


 /** Wraps a given array in a queue using the natural order.
 *
 * <p>The queue returned by this method will be backed by the given array.
 * The elements of the array will be rearranged so to form a heap (this is
 * more efficient than enqueing the elements of {@code a} one by one).
 *
 * @param a an array.
```

```
 */
 public HEAP_PRIORITY_QUEUE(final KEY_GENERIC_TYPE[] a) {
  this(a, a.length);
 }

#if KEYS_PRIMITIVE

 /** Creates a queue using the elements in a type-specific collection using a given comparator.
  *
  * <p>This constructor is more efficient than enqueing the elements of {@code collection} one by one.
  *
  * @param collection a collection; its elements will be used to initialize the queue.
  * @param c the comparator used in this queue, or {@code null} for the natural order.
  */
 public HEAP_PRIORITY_QUEUE(final COLLECTION KEY_EXTENDS_GENERIC collection, final
KEY_COMPARATOR KEY_SUPER_GENERIC c) {
  this(collection.TO_KEY_ARRAY(), c);
 }

 /** Creates a queue using the elements in a type-specific collection using the natural order.
  *
  * <p>This constructor is
  * more efficient than enqueing the elements of {@code collection} one by one.
  *
  * @param collection a collection; its elements will be used to initialize the queue.
  */
 public HEAP_PRIORITY_QUEUE(final COLLECTION KEY_EXTENDS_GENERIC collection) {
  this(collection, null);
 }

 /** Creates a queue using the elements in a collection using a given comparator.
  *
  * <p>This constructor is more efficient than enqueing the elements of {@code collection} one by one.
  *
  * @param collection a collection; its elements will be used to initialize the queue.
  * @param c the comparator used in this queue, or {@code null} for the natural order.
  */
 public HEAP_PRIORITY_QUEUE(final Collection<? extends KEY_GENERIC_CLASS> collection, final
KEY_COMPARATOR KEY_SUPER_GENERIC c) {
  this(collection.size(), c);
  final Iterator<? extends KEY_GENERIC_CLASS> iterator = collection.iterator();
  final int size = collection.size();
  for(int i = 0 ; i < size; i++) heap[i] = KEY_OBJ2TYPE(iterator.next());
 }

 /** Creates a queue using the elements in a collection using the natural order.
  *
  * <p>This constructor is
```

```
 * more efficient than enqueing the elements of {@code collection} one by one.
 *
 * @param collection a collection; its elements will be used to initialize the queue.
 */
 public HEAP_PRIORITY_QUEUE(final Collection<? extends KEY_GENERIC_CLASS> collection) {
  this(collection, null);
 }
#else
 /** Creates a queue using the elements in a collection using a given comparator.
 *
 * <p>This constructor is more efficient than enqueing the elements of {@code collection} one by one.
 *
 * @param collection a collection; its elements will be used to initialize the queue.
 * @param c the comparator used in this queue, or {@code null} for the natural order.
 */
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public HEAP_PRIORITY_QUEUE(final Collection<? extends KEY_GENERIC_CLASS> collection, final
KEY_COMPARATOR KEY_SUPER_GENERIC c) {
  this(KEY_GENERIC_ARRAY_CAST collection.toArray(), c);
 }

 /** Creates a queue using the elements in a collection using the natural order.
 *
 * <p>This constructor is
 * more efficient than enqueing the elements of {@code collection} one by one.
 *
 * @param collection a collection; its elements will be used to initialize the queue.
 */
 public HEAP_PRIORITY_QUEUE(final Collection<? extends KEY_GENERIC_CLASS> collection) {
  this(collection, null);
 }
#endif

 @Override
  public void enqueue(KEY_GENERIC_TYPE x) {
  if (size == heap.length) heap = ARRAYS.grow(heap, size + 1);

  heap[size++] = x;
  HEAPS.upHeap(heap, size, size - 1, c);
 }

 @Override
 public KEY_GENERIC_TYPE DEQUEUE() {
  if (size == 0) throw new NoSuchElementException();

  final KEY_GENERIC_TYPE result = heap[0];
  heap[0] = heap[--size];
#if KEY_CLASS_Object
```

```
 heap[size] = null;
#endif
 if (size != 0) HEAPS.downHeap(heap, size, 0, c);
 return result;
 }

 @Override
 public KEY_GENERIC_TYPE FIRST() {
 if (size == 0) throw new NoSuchElementException();
 return heap[0];
 }

 @Override
 public void changed() {
 HEAPS.downHeap(heap, size, 0, c);
 }

 @Override
 public int size() { return size; }

 @Override
 public void clear() {
#if KEY_CLASS_Object
 Arrays.fill(heap, 0, size, null);
#endif
 size = 0;
 }

 /** Trims the underlying heap array so that it has exactly {@link #size()} elements. */

 public void trim() {
 heap = ARRAYS.trim(heap, size);
 }

 @Override
 public KEY_COMPARATOR KEY_SUPER_GENERIC comparator() { return c; }

 private void writeObject(java.io.ObjectOutputStream s) throws java.io.IOException {
 s.defaultWriteObject();
 s.writeInt(heap.length);
 for(int i = 0; i < size; i++) s.WRITE_KEY(heap[i]);
 }

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 private void readObject(java.io.ObjectInputStream s) throws java.io.IOException, ClassNotFoundException {
 s.defaultReadObject();
 heap = KEY_GENERIC_ARRAY_CAST new KEY_TYPE[s.readInt()];
 for(int i = 0; i < size; i++) heap[i] = KEY_GENERIC_CAST s.READ_KEY();
```

```
 }


#ifdef TEST

 private static long seed = System.currentTimeMillis();
 private static java.util.Random r = new java.util.Random(seed);

 private static KEY_TYPE genKey() {
#if KEY_CLASS_Byte || KEY_CLASS_Short || KEY_CLASS_Character
  return (KEY_TYPE)(r.nextInt());
#elif KEYS_PRIMITIVE
  return r.NEXT_KEY();
#elif KEY_CLASS_Object
  return Integer.toBinaryString(r.nextInt());
#else
  return new java.io.Serializable() {};
#endif
 }

 private static java.text.NumberFormat format = new java.text.DecimalFormat("#,###.00");
 private static java.text.FieldPosition p = new java.text.FieldPosition(0);

 private static String format(double d) {
  StringBuffer s = new StringBuffer();
  return format.format(d, s, p).toString();
 }

 private static void speedTest(int n, boolean comp) {
  System.out.println("There are presently no speed tests for this class.");
 }


 private static void fatal(String msg) {
  System.out.println(msg);
  System.exit(1);
 }

 private static void ensure(boolean cond, String msg) {
  if (cond) return;
  fatal(msg);
 }

 private static boolean heapEqual(KEY_TYPE[] a, KEY_TYPE[] b, int sizea, int sizeb) {
  if (sizea != sizeb) return false;
  KEY_TYPE[] aa = (KEY_TYPE[])a.clone();
  KEY_TYPE[] bb = (KEY_TYPE[])b.clone();
  java.util.Arrays.sort(aa, 0, sizea);
```

```
 java.util.Arrays.sort(bb, 0, sizeb);
 while(sizea-- != 0) if (! KEY_EQUALS(aa[sizea], bb[sizea])) return false;
 return true;
 }

 private static KEY_TYPE k[];

 protected static void runTest(int n) {
 long ms;
 Exception mThrowsIllegal, tThrowsIllegal, mThrowsOutOfBounds, tThrowsOutOfBounds, mThrowsNoElement,
tThrowsNoElement;
 KEY_TYPE rm = KEY_NULL, rt = KEY_NULL;
 k = new KEY_TYPE[n];

 for(int i = 0; i < n; i++) k[i] = genKey();

 HEAP_PRIORITY_QUEUE m = new
HEAP_PRIORITY_QUEUE(COMPARATORS.NATURAL_COMPARATOR);
 ARRAY_PRIORITY_QUEUE t = new
ARRAY_PRIORITY_QUEUE(COMPARATORS.NATURAL_COMPARATOR);

 /* We add pairs to t. */
 for(int i = 0; i < n / 2;  i++) {
  t.enqueue(k[i]);
  m.enqueue(k[i]);
 }

 ensure(heapEqual(m.heap, t.array, m.size(), t.size()), "Error (" + seed + "): m and t differ after creation (" + m + ", "
+ t + ")");

 if (m.size() != 0) {
  ensure(KEY_EQUALS(m.FIRST(), t.FIRST()), "Error (" + seed + "): m and t differ in first element after creation ("
+ m.FIRST() + ", " + t.FIRST() + ")");
 }

 /* Now we add and remove random data in m and t, checking that the result is the same. */

 for(int i=0; i<2*n;  i++) {

  if (r.nextDouble() < 0.01) {
   t.clear();
   m.clear();
   for(int j = 0; j < n / 2;  j++) {
    t.enqueue(k[j]);
    m.enqueue(k[j]);
   }
  }
```

```
  KEY_TYPE T = genKey();

  mThrowsNoElement = tThrowsNoElement = mThrowsOutOfBounds = tThrowsOutOfBounds = mThrowsIllegal =
tThrowsIllegal = null;

  try {
   m.enqueue(T);
  }
  catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }
  catch (IllegalArgumentException e) { mThrowsIllegal = e; }

  try {
   t.enqueue(T);
  }
  catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }
  catch (IllegalArgumentException e) { tThrowsIllegal = e; }

  ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + seed + "): enqueue()
divergence in IndexOutOfBoundsException for " + T + " (" + mThrowsOutOfBounds + ", " + tThrowsOutOfBounds
+ ")");
  ensure((mThrowsIllegal == null) == (tThrowsIllegal == null), "Error (" + seed + "): enqueue() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");

  ensure(heapEqual(m.heap, t.array, m.size(), t.size()), "Error (" + seed + "): m and t differ after enqueue (" + m + ", "
+ t + ")");

  if (m.size() != 0) {
   ensure(KEY_EQUALS(m.FIRST(), t.FIRST()), "Error (" + seed + "): m and t differ in first element after enqueue ("
+ m.FIRST() + ", " + t.FIRST() + ")");
  }

  mThrowsNoElement = tThrowsNoElement = mThrowsOutOfBounds = tThrowsOutOfBounds = mThrowsIllegal =
tThrowsIllegal = null;

  try {
   rm = m.DEQUEUE();
  }
  catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }
  catch (IllegalArgumentException e) { mThrowsIllegal = e; }
  catch (NoSuchElementException e) { mThrowsNoElement = e; }

  try {
   rt = t.DEQUEUE();
  }
  catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }
  catch (IllegalArgumentException e) { tThrowsIllegal = e; }
  catch (NoSuchElementException e) { tThrowsNoElement = e; }
```

```
  ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + seed + "): dequeue() divergence in IndexOutOfBoundsException (" + mThrowsOutOfBounds + ", " + tThrowsOutOfBounds + ")");
  ensure((mThrowsIllegal == null) == (tThrowsIllegal == null), "Error (" + seed + "): dequeue() divergence in IllegalArgumentException  (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
  ensure((mThrowsNoElement == null) == (tThrowsNoElement == null), "Error (" + seed + "): dequeue() divergence in NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
  if (mThrowsOutOfBounds == null) ensure(KEY_EQUALS(rt, rm) , "Error (" + seed + "): divergence in dequeue() between t and m (" + rt + ", " + rm + ")");


  ensure(heapEqual(m.heap, t.array, m.size(), t.size()), "Error (" + seed + "): m and t differ after dequeue (" + m + ", " + t + ")");

  if (m.size() != 0) {
   ensure(KEY_EQUALS(m.FIRST(), t.FIRST()), "Error (" + seed + "): m and t differ in first element after dequeue (" + m.FIRST() + ", " + t.FIRST() + ")");
  }

  /* Now we save and read m. */

  try {
   java.io.File ff = new java.io.File("it.unimi.dsi.fastutil.test");
   java.io.OutputStream os = new java.io.FileOutputStream(ff);
   java.io.ObjectOutputStream oos = new java.io.ObjectOutputStream(os);

   oos.writeObject(m);
   oos.close();

   java.io.InputStream is = new java.io.FileInputStream(ff);
   java.io.ObjectInputStream ois = new java.io.ObjectInputStream(is);

   m = (HEAP_PRIORITY_QUEUE)ois.readObject();
   ois.close();
   ff.delete();
  }
  catch(Exception e) {
   e.printStackTrace();
   System.exit(1);
  }

  ensure(heapEqual(m.heap, t.array, m.size(), t.size()), "Error (" + seed + "): m and t differ after save/read");

  HEAP_PRIORITY_QUEUE m2 = new HEAP_PRIORITY_QUEUE(t.array, t.size());
  ARRAY_PRIORITY_QUEUE t2 = new ARRAY_PRIORITY_QUEUE(m.heap, m.size());
  m = m2;
  t = t2;

  ensure(heapEqual(m.heap, t.array, m.size(), t.size()), "Error (" + seed + "): m and t differ after wrap (" + m + ", " + t
```

```
+ ")");

   if (m.size() != 0) {
    ensure(KEY_EQUALS(m.FIRST(), t.FIRST()), "Error (" + seed + "): m and t differ in first element after wrap (" +
m.FIRST() + ", " + t.FIRST() + ")");
   }

   if (m.size() != 0 && ((new OPEN_HASH_SET(m.heap, 0, m.size())).size() == m.size())) {

    int j = t.size(), M = --j;
#if KEYS_PRIMITIVE
    while(j-- != 0) if (KEY_LESS(t.array[j], t.array[M])) M = j;
#else
    while(j-- != 0) if (((Comparable)t.array[j]).compareTo(t.array[M])< 0) M = j;
#endif

    m.heap[0] = t.array[M] = genKey();

    m.changed();
    t.changed();

    ensure(heapEqual(m.heap, t.array, m.size(), t.size()), "Error (" + seed + "): m and t differ after change (" + m + ", " +
t + ")");

    if (m.size() != 0) {
     ensure(KEY_EQUALS(m.FIRST(), t.FIRST()), "Error (" + seed + "): m and t differ in first element after change ("
+ m.FIRST() + ", " + t.FIRST() + ")");
    }
   }
  }


  /* Now we check that m actually holds the same data. */

  m.clear();
  ensure(m.isEmpty(), "Error (" + seed + "): m is not empty after clear()");

  System.out.println("Test OK");
 }



 public static void main(String args[]) {
  int n  = Integer.parseInt(args[1]);
  if (args.length > 2) r = new java.util.Random(seed = Long.parseLong(args[2]));


  try {
```

```
  if ("speedTest".equals(args[0]) || "speedComp".equals(args[0])) speedTest(n, "speedComp".equals(args[0]));
  else if ("test".equals(args[0])) runTest(n);
  } catch(Throwable e) {
  e.printStackTrace(System.err);
  System.err.println("seed: " + seed);
  }
 }

#endif

}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/HeapPriorityQueue.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2009-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*
*
*
* Copyright (C) 1999 CERN - European Organization for Nuclear Research.
*
*   Permission to use, copy, modify, distribute and sell this software and
*   its documentation for any purpose is hereby granted without fee,
*   provided that the above copyright notice appear in all copies and that
*   both that copyright notice and this permission notice appear in
*   supporting documentation. CERN makes no representations about the
*   suitability of this software for any purpose. It is provided "as is"
*   without expressed or implied warranty.
*/

package PACKAGE;

import java.util.Arrays;
```

```
import java.util.Random;

import it.unimi.dsi.fastutil.BigArrays;
import it.unimi.dsi.fastutil.Hash;
import static it.unimi.dsi.fastutil.BigArrays.ensureLength;
import static it.unimi.dsi.fastutil.BigArrays.start;
import static it.unimi.dsi.fastutil.BigArrays.segment;
import static it.unimi.dsi.fastutil.BigArrays.displacement;
import static it.unimi.dsi.fastutil.BigArrays.SEGMENT_MASK;
import static it.unimi.dsi.fastutil.BigArrays.SEGMENT_SHIFT;
import static it.unimi.dsi.fastutil.BigArrays.SEGMENT_SIZE;

#if KEYS_PRIMITIVE

#if ! KEY_CLASS_Byte && ! KEY_CLASS_Boolean
import it.unimi.dsi.fastutil.bytes.ByteBigArrays;
#endif

/** A class providing static methods and objects that do useful things with {@linkplain BigArrays big arrays}.
*
* <p>In particular, the {@code forceCapacity()}, {@code ensureCapacity()}, {@code grow()},
* {@code trim()} and {@code setLength()} methods allow to handle
* big arrays much like array lists.
*
* <p>Note that {@link it.unimi.dsi.fastutil.io.BinIO} and {@link it.unimi.dsi.fastutil.io.TextIO}
* contain several methods that make it possible to load and save big arrays of primitive types as sequences
* of elements in {@link java.io.DataInput} format (i.e., not as objects) or as sequences of lines of text.
*
* @see BigArrays
*/

public final class BIG_ARRAYS {

#else

import java.util.Comparator;

/** A class providing static methods and objects that do useful things with {@linkplain BigArrays big arrays}.
*
* <p>In particular, the {@code ensureCapacity()}, {@code grow()},
* {@code trim()} and {@code setLength()} methods allow to handle
* arrays much like array lists.
*
* <p>Note that {@link it.unimi.dsi.fastutil.io.BinIO} and {@link it.unimi.dsi.fastutil.io.TextIO}
* contain several methods make it possible to load and save big arrays of primitive types as sequences
* of elements in {@link java.io.DataInput} format (i.e., not as objects) or as sequences of lines of text.
*
* <p><strong>Warning:</strong> creating arrays
```

```
* using {@linkplain java.lang.reflect.Array#newInstance(Class,int) reflection}, as it
* happens in {@link #ensureCapacity(Object[][],long,long)} and {@link #grow(Object[][],long,long)},
* is <em>significantly slower</em> than using {@code new}. This phenomenon is particularly
* evident in the first growth phases of an array reallocated with doubling (or similar) logic.
*
* @see BigArrays
*/

public final class BIG_ARRAYS {

#endif
 private BIG_ARRAYS() {}

 /** A static, final, empty big array. */
 public static final KEY_TYPE[][] EMPTY_BIG_ARRAY = {};

 /** A static, final, empty big array to be used as default big array in allocations. An
  * object distinct from {@link #EMPTY_BIG_ARRAY} makes it possible to have different
  * behaviors depending on whether the user required an empty allocation, or we are
  * just lazily delaying allocation.
  *
  * @see java.util.ArrayList
  */
 public static final KEY_TYPE[][] DEFAULT_EMPTY_BIG_ARRAY = {};

 /** Returns the element of the given big array of specified index.
  *
  * @param array a big array.
  * @param index a position in the big array.
  * @return the element of the big array at the specified position.
  */
 public static KEY_GENERIC KEY_GENERIC_TYPE get(final KEY_GENERIC_TYPE[][] array, final long index)
{
  return array[segment(index)][displacement(index)];
 }

 /** Sets the element of the given big array of specified index.
  *
  * @param array a big array.
  * @param index a position in the big array.
  * @param value the new value for the array element at the specified position.
  */
 public static KEY_GENERIC void set(final KEY_GENERIC_TYPE[][] array, final long index,
KEY_GENERIC_TYPE value) {
  array[segment(index)][displacement(index)] = value;
 }

 /** Swaps the element of the given big array of specified indices.
```

```
 *
 * @param array a big array.
 * @param first a position in the big array.
 * @param second a position in the big array.
 */
public static KEY_GENERIC void swap(final KEY_GENERIC_TYPE[][] array, final long first, final long second)
{
 final KEY_GENERIC_TYPE t = array[segment(first)][displacement(first)];
 array[segment(first)][displacement(first)] = array[segment(second)][displacement(second)];
 array[segment(second)][displacement(second)] = t;
}

#if KEYS_PRIMITIVE && ! KEY_CLASS_Boolean
 /** Adds the specified increment the element of the given big array of specified index.
 *
 * @param array a big array.
 * @param index a position in the big array.
 * @param incr the increment
 */
public static void add(final KEY_GENERIC_TYPE[][] array, final long index, KEY_GENERIC_TYPE incr) {
 array[segment(index)][displacement(index)] += incr;
}

 /** Multiplies by the specified factor the element of the given big array of specified index.
 *
 * @param array a big array.
 * @param index a position in the big array.
 * @param factor the factor
 */
public static void mul(final KEY_GENERIC_TYPE[][] array, final long index, KEY_GENERIC_TYPE factor) {
 array[segment(index)][displacement(index)] *= factor;
}

 /** Increments the element of the given big array of specified index.
 *
 * @param array a big array.
 * @param index a position in the big array.
 */
public static void incr(final KEY_GENERIC_TYPE[][] array, final long index) {
 array[segment(index)][displacement(index)]++;
}

 /** Decrements the element of the given big array of specified index.
 *
 * @param array a big array.
 * @param index a position in the big array.
 */
public static void decr(final KEY_GENERIC_TYPE[][] array, final long index) {
```

```
 array[segment(index)][displacement(index)]--;
 }


#endif


 /** Returns the length of the given big array.
  *
  * @param array a big array.
  * @return the length of the given big array.
  */
 public static KEY_GENERIC long length(final KEY_GENERIC_TYPE[][] array) {
  final int length = array.length;
  return length == 0 ? 0 : start(length - 1) + array[length - 1].length;
 }

 /** Copies a big array from the specified source big array, beginning at the specified position, to the specified position of the destination big array.
  * Handles correctly overlapping regions of the same big array.
  *
  * @param srcArray the source big array.
  * @param srcPos the starting position in the source big array.
  * @param destArray the destination big array.
  * @param destPos the starting position in the destination data.
  * @param length the number of elements to be copied.
  */
 public static KEY_GENERIC void copy(final KEY_GENERIC_TYPE[][] srcArray, final long srcPos, final KEY_GENERIC_TYPE[][] destArray, final long destPos, long length) {
  if (destPos <= srcPos) {
   int srcSegment = segment(srcPos);
   int destSegment = segment(destPos);
   int srcDispl = displacement(srcPos);
   int destDispl = displacement(destPos);
   int l;
   while(length > 0) {
    l = (int)Math.min(length, Math.min(srcArray[srcSegment].length - srcDispl, destArray[destSegment].length - destDispl));
    System.arraycopy(srcArray[srcSegment], srcDispl, destArray[destSegment], destDispl, l);
    if ((srcDispl += l) == SEGMENT_SIZE) {
     srcDispl = 0;
     srcSegment++;
    }
    if ((destDispl += l) == SEGMENT_SIZE) {
     destDispl = 0;
     destSegment++;
    }
    length -= l;
```

```
  }
 }
 else {
  int srcSegment = segment(srcPos + length);
  int destSegment = segment(destPos + length);
  int srcDispl = displacement(srcPos + length);
  int destDispl = displacement(destPos + length);
  int l;
  while(length > 0) {
   if (srcDispl == 0) {
    srcDispl = SEGMENT_SIZE;
    srcSegment--;
   }
   if (destDispl == 0) {
    destDispl = SEGMENT_SIZE;
    destSegment--;
   }
   l = (int)Math.min(length, Math.min(srcDispl, destDispl));
   System.arraycopy(srcArray[srcSegment], srcDispl - l, destArray[destSegment], destDispl - l, l);
   srcDispl -= l;
   destDispl -= l;
   length -= l;
  }
 }
}

 /** Copies a big array from the specified source big array, beginning at the specified position, to the specified position of the destination array.
  *
  * @param srcArray the source big array.
  * @param srcPos the starting position in the source big array.
  * @param destArray the destination array.
  * @param destPos the starting position in the destination data.
  * @param length the number of elements to be copied.
  */
 public static KEY_GENERIC void copyFromBig(final KEY_GENERIC_TYPE[][] srcArray, final long srcPos, final KEY_GENERIC_TYPE[] destArray, int destPos, int length) {
  int srcSegment = segment(srcPos);
  int srcDispl = displacement(srcPos);
  int l;
  while(length > 0) {
   l = Math.min(srcArray[srcSegment].length - srcDispl, length);
   System.arraycopy(srcArray[srcSegment], srcDispl, destArray, destPos, l);
   if ((srcDispl += l) == SEGMENT_SIZE) {
    srcDispl = 0;
    srcSegment++;
   }
   destPos += l;
```

```
  length -= l;
 }
}

 /** Copies an array from the specified source array, beginning at the specified position, to the specified position of the destination big array.
  *
  * @param srcArray the source array.
  * @param srcPos the starting position in the source array.
  * @param destArray the destination big array.
  * @param destPos the starting position in the destination data.
  * @param length the number of elements to be copied.
  */
 public static KEY_GENERIC void copyToBig(final KEY_GENERIC_TYPE[] srcArray, int srcPos, final KEY_GENERIC_TYPE[][] destArray, final long destPos, long length) {
  int destSegment = segment(destPos);
  int destDispl = displacement(destPos);
  int l;
  while(length > 0) {
   l = (int)Math.min(destArray[destSegment].length - destDispl, length);
   System.arraycopy(srcArray, srcPos, destArray[destSegment], destDispl, l);
   if ((destDispl += l) == SEGMENT_SIZE) {
    destDispl = 0;
    destSegment++;
   }
   srcPos += l;
   length -= l;
  }
 }

#if KEY_CLASS_Object
 /** Creates a new big array using the given one as prototype.
  *
  * <p>This method returns a new big array of the given length whose element
  * are of the same class as of those of {@code prototype}. In case
  * of an empty big array, it tries to return {@link #EMPTY_BIG_ARRAY}, if possible.
  *
  * @param prototype a big array that will be used to type the new one.
  * @param length the length of the new big array.
  * @return a new big array of given type and length.
  */

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public static <K> K[][] newBigArray(final K[][] prototype, final long length) {
  return (K[][])newBigArray(prototype.getClass().getComponentType(), length);
 }

 /** Creates a new big array using a the given one as component type.
```

```
 *
 * <p>This method returns a new big array whose segments
 * are of class {@code componentType}. In case
 * of an empty big array, it tries to return {@link #EMPTY_BIG_ARRAY}, if possible.
 *
 * @param componentType a class representing the type of segments of the array to be created.
 * @param length the length of the new big array.
 * @return a new big array of given type and length.
 */

 private static Object[][] newBigArray(Class<?> componentType, final long length) {
  if (length == 0 && componentType == Object[].class) return EMPTY_BIG_ARRAY;
  ensureLength(length);
  final int baseLength = (int)((length + SEGMENT_MASK) >>> SEGMENT_SHIFT);
  Object[][] base = (Object[][])java.lang.reflect.Array.newInstance(componentType, baseLength);
  final int residual = (int)(length & SEGMENT_MASK);
  if (residual != 0) {
   for(int i = 0; i < baseLength - 1; i++) base[i] =
(Object[])java.lang.reflect.Array.newInstance(componentType.getComponentType(), SEGMENT_SIZE);
   base[baseLength - 1] = (Object[])java.lang.reflect.Array.newInstance(componentType.getComponentType(),
residual);
  }
  else for(int i = 0; i < baseLength; i++) base[i] =
(Object[])java.lang.reflect.Array.newInstance(componentType.getComponentType(), SEGMENT_SIZE);

  return base;
 }
#endif

 /** Creates a new big array.
  *
  * @param length the length of the new big array.
  * @return a new big array of given length.
  */

 public static KEY_TYPE[][] newBigArray(final long length) {
  if (length == 0) return EMPTY_BIG_ARRAY;
  ensureLength(length);
  final int baseLength = (int)((length + SEGMENT_MASK) >>> SEGMENT_SHIFT);
  KEY_TYPE[][] base = new KEY_TYPE[baseLength][];
  final int residual = (int)(length & SEGMENT_MASK);
  if (residual != 0) {
   for(int i = 0; i < baseLength - 1; i++) base[i] = new KEY_TYPE[SEGMENT_SIZE];
   base[baseLength - 1] = new KEY_TYPE[residual];
  }
  else for(int i = 0; i < baseLength; i++) base[i] = new KEY_TYPE[SEGMENT_SIZE];

  return base;
```

```
 }

#if KEY_CLASS_Object
 /** Turns a standard array into a big array.
  *
  * <p>Note that the returned big array might contain as a segment the original array.
  *
  * @param array an array.
  * @return a new big array with the same length and content of {@code array}.
  */

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public static <K> K[][] wrap(final K[] array) {
  if (array.length == 0 && array.getClass() == Object[].class) return KEY_GENERIC_BIG_ARRAY_CAST
EMPTY_BIG_ARRAY;
  if (array.length <= SEGMENT_SIZE) {
   final K[][] bigArray = (K[][])java.lang.reflect.Array.newInstance(array.getClass(), 1);
   bigArray[0] = array;
   return bigArray;
  }
  final K[][] bigArray = (K[][])newBigArray(array.getClass(), array.length);
  for(int i = 0; i < bigArray.length; i++) System.arraycopy(array, (int)start(i), bigArray[i], 0, bigArray[i].length);
  return bigArray;
 }

#else
 /** Turns a standard array into a big array.
  *
  * <p>Note that the returned big array might contain as a segment the original array.
  *
  * @param array an array.
  * @return a new big array with the same length and content of {@code array}.
  */

 public static KEY_TYPE[][] wrap(final KEY_TYPE[] array) {
  if (array.length == 0) return EMPTY_BIG_ARRAY;
  if (array.length <= SEGMENT_SIZE) return new KEY_TYPE[][] { array };
  final KEY_TYPE[][] bigArray = newBigArray(array.length);
  for(int i = 0; i < bigArray.length; i++) System.arraycopy(array, (int)start(i), bigArray[i], 0, bigArray[i].length);
  return bigArray;
 }
#endif
 /** Ensures that a big array can contain the given number of entries.
  *
  * <p>If you cannot foresee whether this big array will need again to be
  * enlarged, you should probably use {@code grow()} instead.
  *
  * <p><strong>Warning:</strong> the returned array might use part of the segments of the original
```

```
 * array, which must be considered read-only after calling this method.
 *
 * @param array a big array.
 * @param length the new minimum length for this big array.
 * @return {@code array}, if it contains {@code length} entries or more; otherwise,
 * a big array with {@code length} entries whose first {@code length(array)}
 * entries are the same as those of {@code array}.
 */
 public static KEY_GENERIC KEY_GENERIC_TYPE[][] ensureCapacity(final KEY_GENERIC_TYPE[][] array,
final long length) {
  return ensureCapacity(array, length, length(array));
 }

#if KEY_CLASS_Object

 /** Forces a big array to contain the given number of entries, preserving just a part of the big array.
 *
 * <p>This method returns a new big array of the given length whose element
 * are of the same class as of those of {@code array}.
 *
 * <p><strong>Warning:</strong> the returned array might use part of the segments of the original
 * array, which must be considered read-only after calling this method.
 *
 * @param array a big array.
 * @param length the new minimum length for this big array.
 * @param preserve the number of elements of the big array that must be preserved in case a new allocation is
necessary.
 * @return a big array with {@code length} entries whose first {@code preserve}
 * entries are the same as those of {@code array}.
 */
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public static KEY_GENERIC KEY_GENERIC_TYPE[][] forceCapacity(final KEY_GENERIC_TYPE[][] array,
final long length, final long preserve) {
  ensureLength(length);
  final int valid = array.length - (array.length == 0 || array.length > 0 && array[array.length - 1].length ==
SEGMENT_SIZE ? 0 : 1);
  final int baseLength = (int)((length + SEGMENT_MASK) >>> SEGMENT_SHIFT);
  final KEY_GENERIC_TYPE[][] base = Arrays.copyOf(array, baseLength);
  final Class<?> componentType = array.getClass().getComponentType();
  final int residual = (int)(length & SEGMENT_MASK);
  if (residual != 0) {
   for(int i = valid; i < baseLength - 1; i++) base[i] =
(KEY_GENERIC_TYPE[])java.lang.reflect.Array.newInstance(componentType.getComponentType(),
SEGMENT_SIZE);
   base[baseLength - 1] =
(KEY_GENERIC_TYPE[])java.lang.reflect.Array.newInstance(componentType.getComponentType(), residual);
  }
  else for(int i = valid; i < baseLength; i++) base[i] =
```

```
(KEY_GENERIC_TYPE[])java.lang.reflect.Array.newInstance(componentType.getComponentType(), SEGMENT_SIZE);

  if (preserve - (valid * (long)SEGMENT_SIZE) > 0) copy(array, valid * (long)SEGMENT_SIZE, base, valid * (long)SEGMENT_SIZE, preserve - (valid * (long)SEGMENT_SIZE));
  return base;
 }

 /** Ensures that a big array can contain the given number of entries, preserving just a part of the big array.
  *
  * <p>This method returns a new big array of the given length whose element
  * are of the same class as of those of {@code array}.
  *
  * <p><strong>Warning:</strong> the returned array might use part of the segments of the original
  * array, which must be considered read-only after calling this method.
  *
  * @param array a big array.
  * @param length the new minimum length for this big array.
  * @param preserve the number of elements of the big array that must be preserved in case a new allocation is necessary.
  * @return {@code array}, if it can contain {@code length} entries or more; otherwise,
  * a big array with {@code length} entries whose first {@code preserve}
  * entries are the same as those of {@code array}.
  */
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public static KEY_GENERIC KEY_GENERIC_TYPE[][] ensureCapacity(final KEY_GENERIC_TYPE[][] array, final long length, final long preserve) {
  return length > length(array) ? forceCapacity(array, length, preserve) : array;
 }

#else

 /** Forces a big array to contain the given number of entries, preserving just a part of the big array.
  *
  * <p><strong>Warning:</strong> the returned array might use part of the segments of the original
  * array, which must be considered read-only after calling this method.
  *
  * @param array a big array.
  * @param length the new minimum length for this big array.
  * @param preserve the number of elements of the big array that must be preserved in case a new allocation is necessary.
  * @return a big array with {@code length} entries whose first {@code preserve}
  * entries are the same as those of {@code array}.
  */
 public static KEY_TYPE[][] forceCapacity(final KEY_TYPE[][] array, final long length, final long preserve) {
  ensureLength(length);
  final int valid = array.length - (array.length == 0 || array.length > 0 && array[array.length - 1].length == SEGMENT_SIZE ? 0 : 1);
```

```
  final int baseLength = (int)((length + SEGMENT_MASK) >>> SEGMENT_SHIFT);
  final KEY_TYPE[][] base = Arrays.copyOf(array, baseLength);
  final int residual = (int)(length & SEGMENT_MASK);
  if (residual != 0) {
   for(int i = valid; i < baseLength - 1; i++) base[i] = new KEY_TYPE[SEGMENT_SIZE];
   base[baseLength - 1] = new KEY_TYPE[residual];
  }
  else for(int i = valid; i < baseLength; i++) base[i] = new KEY_TYPE[SEGMENT_SIZE];

  if (preserve - (valid * (long)SEGMENT_SIZE) > 0) copy(array, valid * (long)SEGMENT_SIZE, base, valid *
(long)SEGMENT_SIZE, preserve - (valid * (long)SEGMENT_SIZE));
  return base;
 }

 /** Ensures that a big array can contain the given number of entries, preserving just a part of the big array.
  *
  * <p><strong>Warning:</strong> the returned array might use part of the segments of the original
  * array, which must be considered read-only after calling this method.
  *
  * @param array a big array.
  * @param length the new minimum length for this big array.
  * @param preserve the number of elements of the big array that must be preserved in case a new allocation is
necessary.
  * @return {@code array}, if it can contain {@code length} entries or more; otherwise,
  * a big array with {@code length} entries whose first {@code preserve}
  * entries are the same as those of {@code array}.
  */
 public static KEY_TYPE[][] ensureCapacity(final KEY_TYPE[][] array, final long length, final long preserve) {
  return length > length(array) ? forceCapacity(array, length, preserve) : array;
 }

#endif

 /** Grows the given big array to the maximum between the given length and
  * the current length increased by 50%, provided that the given
  * length is larger than the current length.
  *
  * <p>If you want complete control on the big array growth, you
  * should probably use {@code ensureCapacity()} instead.
  *
  * <p><strong>Warning:</strong> the returned array might use part of the segments of the original
  * array, which must be considered read-only after calling this method.
  *
  * @param array a big array.
  * @param length the new minimum length for this big array.
  * @return {@code array}, if it can contain {@code length}
  * entries; otherwise, a big array with
  * max({@code length},{@code length(array)}/&phi;) entries whose first
```

```
 * {@code length(array)} entries are the same as those of {@code array}.
 * */

 public static KEY_GENERIC KEY_GENERIC_TYPE[][] grow(final KEY_GENERIC_TYPE[][] array, final long
length) {
  final long oldLength = length(array);
  return length > oldLength ? grow(array, length, oldLength) : array;
 }

 /** Grows the given big array to the maximum between the given length and
  * the current length increased by 50%, provided that the given
  * length is larger than the current length, preserving just a part of the big array.
  *
  * <p>If you want complete control on the big array growth, you
  * should probably use {@code ensureCapacity()} instead.
  *
  * <p><strong>Warning:</strong> the returned array might use part of the segments of the original
  * array, which must be considered read-only after calling this method.
  *
  * @param array a big array.
  * @param length the new minimum length for this big array.
  * @param preserve the number of elements of the big array that must be preserved in case a new allocation is
necessary.
  * @return {@code array}, if it can contain {@code length}
  * entries; otherwise, a big array with
  * max({@code length},{@code length(array)}/&phi;) entries whose first
  * {@code preserve} entries are the same as those of {@code array}.
  * */

 public static KEY_GENERIC KEY_GENERIC_TYPE[][] grow(final KEY_GENERIC_TYPE[][] array, final long
length, final long preserve) {
  final long oldLength = length(array);
  return length > oldLength ? ensureCapacity(array, Math.max(oldLength + (oldLength >> 1), length), preserve) :
array;
 }

#if KEY_CLASS_Object

 /** Trims the given big array to the given length.
  *
  * <p><strong>Warning:</strong> the returned array might use part of the segments of the original
  * array, which must be considered read-only after calling this method.
  *
  * @param array a big array.
  * @param length the new maximum length for the big array.
  * @return {@code array}, if it contains {@code length}
  * entries or less; otherwise, a big array with
  * {@code length} entries whose entries are the same as
```

```
 * the first {@code length} entries of {@code array}.
 *
 */

 public static KEY_GENERIC KEY_GENERIC_TYPE[][] trim(final KEY_GENERIC_TYPE[][] array, final long
length) {
  ensureLength(length);
  final long oldLength = length(array);
  if (length >= oldLength) return array;
  final int baseLength = (int)((length + SEGMENT_MASK) >>> SEGMENT_SHIFT);
  final KEY_GENERIC_TYPE[][] base = Arrays.copyOf(array, baseLength);
  final int residual = (int)(length & SEGMENT_MASK);
  if (residual != 0) base[baseLength - 1] = ARRAYS.trim(base[baseLength - 1], residual);
  return base;
 }

#else

 /** Trims the given big array to the given length.
  *
  * <p><strong>Warning:</strong> the returned array might use part of the segments of the original
  * array, which must be considered read-only after calling this method.
  *
  * @param array a big array.
  * @param length the new maximum length for the big array.
  * @return {@code array}, if it contains {@code length}
  * entries or less; otherwise, a big array with
  * {@code length} entries whose entries are the same as
  * the first {@code length} entries of {@code array}.
  *
  */

 public static KEY_GENERIC KEY_GENERIC_TYPE[][] trim(final KEY_GENERIC_TYPE[][] array, final long
length) {
  ensureLength(length);
  final long oldLength = length(array);
  if (length >= oldLength) return array;
  final int baseLength = (int)((length + SEGMENT_MASK) >>> SEGMENT_SHIFT);
  final KEY_TYPE[][] base = Arrays.copyOf(array, baseLength);
  final int residual = (int)(length & SEGMENT_MASK);
  if (residual != 0) base[baseLength - 1] = ARRAYS.trim(base[baseLength - 1], residual);
  return base;
 }

#endif

 /** Sets the length of the given big array.
  *
```

```
 * <p><strong>Warning:</strong> the returned array might use part of the segments of the original
 * array, which must be considered read-only after calling this method.
 *
 * @param array a big array.
 * @param length the new length for the big array.
 * @return {@code array}, if it contains exactly {@code length}
 * entries; otherwise, if it contains <em>more</em> than
 * {@code length} entries, a big array with {@code length} entries
 * whose entries are the same as the first {@code length} entries of
 * {@code array}; otherwise, a big array with {@code length} entries
 * whose first {@code length(array)} entries are the same as those of
 * {@code array}.
 *
 */

 public static KEY_GENERIC KEY_GENERIC_TYPE[][] setLength(final KEY_GENERIC_TYPE[][] array, final
long length) {
  final long oldLength = length(array);
  if (length == oldLength) return array;
  if (length < oldLength) return trim(array, length);
  return ensureCapacity(array, length);
 }

 /** Returns a copy of a portion of a big array.
  *
  * @param array a big array.
  * @param offset the first element to copy.
  * @param length the number of elements to copy.
  * @return a new big array containing {@code length} elements of {@code array} starting at {@code offset}.
  */

 public static KEY_GENERIC KEY_GENERIC_TYPE[][] copy(final KEY_GENERIC_TYPE[][] array, final long
offset, final long length) {
  ensureOffsetLength(array, offset, length);
  final KEY_GENERIC_TYPE[][] a =
#if KEY_CLASS_Object
   newBigArray(array, length);
#else
   newBigArray(length);
#endif
  copy(array, offset, a, 0, length);
  return a;
 }

 /** Returns a copy of a big array.
  *
  * @param array a big array.
  * @return a copy of {@code array}.
```

```
 */

public static KEY_GENERIC KEY_GENERIC_TYPE[][] copy(final KEY_GENERIC_TYPE[][] array) {
 final KEY_GENERIC_TYPE[][] base = array.clone();
 for(int i = base.length; i-- != 0;) base[i] = array[i].clone();
 return base;
}

/** Fills the given big array with the given value.
 *
 * <p>This method uses a backward loop. It is significantly faster than the corresponding
 * method in {@link java.util.Arrays}.
 *
 * @param array a big array.
 * @param value the new value for all elements of the big array.
 */

public static KEY_GENERIC void fill(final KEY_GENERIC_TYPE[][] array, final KEY_GENERIC_TYPE value)
{
 for(int i = array.length; i-- != 0;) Arrays.fill(array[i], value);
}

/** Fills a portion of the given big array with the given value.
 *
 * <p>If possible (i.e., {@code from} is 0) this method uses a
 * backward loop. In this case, it is significantly faster than the
 * corresponding method in {@link java.util.Arrays}.
 *
 * @param array a big array.
 * @param from the starting index of the portion to fill.
 * @param to the end index of the portion to fill.
 * @param value the new value for all elements of the specified portion of the big array.
 */

public static KEY_GENERIC void fill(final KEY_GENERIC_TYPE[][] array, final long from, long to, final
KEY_GENERIC_TYPE value) {
 final long length = length(array);
 BigArrays.ensureFromTo(length, from, to);
 if (length == 0) return; // To avoid addressing array[0]
 int fromSegment = segment(from);
 int toSegment = segment(to);
 int fromDispl = displacement(from);
 int toDispl = displacement(to);
 if (fromSegment == toSegment) {
  Arrays.fill(array[fromSegment], fromDispl, toDispl, value);
  return;
 }
```

```
 if (toDispl != 0) Arrays.fill(array[toSegment], 0, toDispl, value);
 while(--toSegment > fromSegment) Arrays.fill(array[toSegment], value);
 Arrays.fill(array[fromSegment], fromDispl, SEGMENT_SIZE, value);
 }


 /** Returns true if the two big arrays are elementwise equal.
  *
  * <p>This method uses a backward loop. It is significantly faster than the corresponding
  * method in {@link java.util.Arrays}.
  *
  * @param a1 a big array.
  * @param a2 another big array.
  * @return true if the two big arrays are of the same length, and their elements are equal.
  */

 public static KEY_GENERIC boolean equals(final KEY_GENERIC_TYPE[][] a1, final KEY_GENERIC_TYPE
a2[][]) {
  if (length(a1) != length(a2)) return false;
  int i = a1.length, j;
  KEY_GENERIC_TYPE[] t, u;
  while(i-- != 0) {
   t = a1[i];
   u = a2[i];
   j = t.length;
   while(j-- != 0) if (! KEY_EQUALS(t[j], u[j])) return false;
  }
  return true;
 }

 /* Returns a string representation of the contents of the specified big array.
  *
  * The string representation consists of a list of the big array's elements, enclosed in square brackets ("[]"). Adjacent
elements are separated by the characters ", " (a comma followed by a space). Returns "null" if {@code a} is null.
  * @param a the big array whose string representation to return.
  * @return the string representation of {@code a}.
  */

 public static KEY_GENERIC String toString(final KEY_GENERIC_TYPE[][] a) {
  if (a == null) return "null";
  final long last = length(a) - 1;
  if (last == - 1) return "[]";
  final StringBuilder b = new StringBuilder();
  b.append('[');
  for (long i = 0; ; i++) {
   b.append(String.valueOf(get(a, i)));
   if (i == last) return b.append(']').toString();
   b.append(", ");
```

```
 }
}


 /** Ensures that a range given by its first (inclusive) and last (exclusive) elements fits a big array.
  *
  * <p>This method may be used whenever a big array range check is needed.
  *
  * @param a a big array.
  * @param from a start index (inclusive).
  * @param to an end index (inclusive).
  * @throws IllegalArgumentException if {@code from} is greater than {@code to}.
  * @throws ArrayIndexOutOfBoundsException if {@code from} or {@code to} are greater than the big array
length or negative.
  */
 public static KEY_GENERIC void ensureFromTo(final KEY_GENERIC_TYPE[][] a, final long from, final long to)
{
  BigArrays.ensureFromTo(length(a), from, to);
 }

 /** Ensures that a range given by an offset and a length fits a big array.
  *
  * <p>This method may be used whenever a big array range check is needed.
  *
  * @param a a big array.
  * @param offset a start index.
  * @param length a length (the number of elements in the range).
  * @throws IllegalArgumentException if {@code length} is negative.
  * @throws ArrayIndexOutOfBoundsException if {@code offset} is negative or {@code offset}+{@code length} is
greater than the big array length.
  */
 public static KEY_GENERIC void ensureOffsetLength(final KEY_GENERIC_TYPE[][] a, final long offset, final
long length) {
  BigArrays.ensureOffsetLength(length(a), offset, length);
 }


 /** A type-specific content-based hash strategy for big arrays. */

 private static final class BigArrayHashStrategy KEY_GENERIC implements
Hash.Strategy<KEY_GENERIC_TYPE[][]>, java.io.Serializable {
  private static final long serialVersionUID = -7046029254386353129L;

  @Override
  public int hashCode(final KEY_GENERIC_TYPE[][] o) { return java.util.Arrays.deepHashCode(o); }

  @Override
  public boolean equals(final KEY_GENERIC_TYPE[][] a, final KEY_GENERIC_TYPE[][] b) { return
```

```
BIG_ARRAYS.equals(a, b); }
 }

 /** A type-specific content-based hash strategy for big arrays.
  *
  * <p>This hash strategy may be used in custom hash collections whenever keys are
  * big arrays, and they must be considered equal by content. This strategy
  * will handle {@code null} correctly, and it is serializable.
  */

 @SuppressWarnings({"rawtypes"})
 public static final Hash.Strategy HASH_STRATEGY = new BigArrayHashStrategy();

 private static final int SMALL = 7;
 private static final int MEDIUM = 40;

 private static KEY_GENERIC void vecSwap(final KEY_GENERIC_TYPE[][] x, long a, long b, final long n) {
  for(int i = 0; i < n; i++, a++, b++) swap(x, a, b);
 }

 private static KEY_GENERIC long med3(final KEY_GENERIC_TYPE x[][], final long a, final long b, final long c,
KEY_COMPARATOR KEY_GENERIC comp) {
  int ab = comp.compare(get(x, a), get(x, b));
  int ac = comp.compare(get(x, a), get(x, c));
  int bc = comp.compare(get(x, b), get(x, c));
  return (ab < 0 ?
   (bc < 0 ? b : ac < 0 ? c : a) :
   (bc > 0 ? b : ac > 0 ? c : a));
 }

 private static KEY_GENERIC void selectionSort(final KEY_GENERIC_TYPE[][] a, final long from, final long to,
final KEY_COMPARATOR KEY_GENERIC comp) {
  for(long i = from; i < to - 1; i++) {
   long m = i;
   for(long j = i + 1; j < to; j++) if (comp.compare(BIG_ARRAYS.get(a, j), BIG_ARRAYS.get(a, m)) < 0) m = j;
   if (m != i) swap(a, i, m);
  }
 }

 /** Sorts the specified range of elements according to the order induced by the specified
  * comparator using quicksort.
  *
  * <p>The sorting algorithm is a tuned quicksort adapted from Jon L. Bentley and M. Douglas
  * McIlroy, &amp;ldquo;Engineering a Sort Function&amp;rdquo;, <i>Software: Practice and Experience</i>, 23(11), pages
  * 1249&amp;minus;1265, 1993.
  *
  * @param x the big array to be sorted.
  * @param from the index of the first element (inclusive) to be sorted.
```

```
 * @param to the index of the last element (exclusive) to be sorted.
 * @param comp the comparator to determine the sorting order.
 */
 public static KEY_GENERIC void quickSort(final KEY_GENERIC_TYPE[][] x, final long from, final long to,
final KEY_COMPARATOR KEY_GENERIC comp) {
 final long len = to - from;

 // Selection sort on smallest arrays
 if (len < SMALL) {
  selectionSort(x, from, to, comp);
  return;
 }

 // Choose a partition element, v
 long m = from + len / 2;  // Small arrays, middle element
 if (len > SMALL) {
  long l = from;
  long n = to - 1;
  if (len > MEDIUM) {  // Big arrays, pseudomedian of 9
   long s = len / 8;
   l = med3(x, l, l + s, l + 2 * s, comp);
   m = med3(x, m - s, m, m + s, comp);
   n = med3(x, n - 2 * s, n - s, n, comp);
  }
  m = med3(x, l, m, n, comp); // Mid-size, med of 3
 }

 final KEY_GENERIC_TYPE v = get(x, m);

 // Establish Invariant: v* (<v)* (>v)* v*
 long a = from, b = a, c = to - 1, d = c;
 while(true) {
  int comparison;
  while (b <= c && (comparison = comp.compare(get(x, b), v)) <= 0) {
   if (comparison == 0) swap(x, a++, b);
   b++;
  }
  while (c >= b && (comparison = comp.compare(get(x, c), v)) >=0) {
   if (comparison == 0) swap(x, c, d--);
   c--;
  }
  if (b > c) break;
  swap(x, b++, c--);
 }

 // Swap partition elements back to middle
 long s, n = to;
 s = Math.min(a - from, b - a);
```

```
 vecSwap(x, from, b - s, s);
 s = Math.min(d - c, n - d- 1);
 vecSwap(x, b, n - s, s);

 // Recursively sort non-partition-elements
 if ((s = b - a) > 1) quickSort(x, from, from + s, comp);
 if ((s = d - c) > 1) quickSort(x, n - s, n, comp);

}

SUPPRESS_WARNINGS_KEY_UNCHECKED
private static KEY_GENERIC long med3(final KEY_GENERIC_TYPE x[][], final long a, final long b, final long c)
{
 int ab = KEY_CMP(get(x, a), get(x, b));
 int ac = KEY_CMP(get(x, a), get(x, c));
 int bc = KEY_CMP(get(x, b), get(x, c));
 return (ab < 0 ?
  (bc < 0 ? b : ac < 0 ? c : a) :
  (bc > 0 ? b : ac > 0 ? c : a));
}


SUPPRESS_WARNINGS_KEY_UNCHECKED
private static KEY_GENERIC void selectionSort(final KEY_GENERIC_TYPE[][] a, final long from, final long to)
{
 for(long i = from; i < to - 1; i++) {
  long m = i;
  for(long j = i + 1; j < to; j++) if (KEY_LESS(BIG_ARRAYS.get(a, j), BIG_ARRAYS.get(a, m))) m = j;
  if (m != i) swap(a, i, m);
 }
}

/** Sorts the specified big array according to the order induced by the specified
 * comparator using quicksort.
 *
 * <p>The sorting algorithm is a tuned quicksort adapted from Jon L. Bentley and M. Douglas
 * McIlroy, &ldquo;Engineering a Sort Function&rdquo;, <i>Software: Practice and Experience</i>, 23(11), pages
 * 1249&minus;1265, 1993.
 *
 * @param x the big array to be sorted.
 * @param comp the comparator to determine the sorting order.
 *
 */
public static KEY_GENERIC void quickSort(final KEY_GENERIC_TYPE[][] x, final KEY_COMPARATOR
KEY_GENERIC comp) {
 quickSort(x, 0, BIG_ARRAYS.length(x), comp);
 }
```

```
/** Sorts the specified range of elements according to the natural ascending order using quicksort.
 *
 * <p>The sorting algorithm is a tuned quicksort adapted from Jon L. Bentley and M. Douglas
 * McIlroy, &ldquo;Engineering a Sort Function&rdquo;, <i>Software: Practice and Experience</i>, 23(11), pages
 * 1249&minus;1265, 1993.
 *
 * @param x the big array to be sorted.
 * @param from the index of the first element (inclusive) to be sorted.
 * @param to the index of the last element (exclusive) to be sorted.
 */

SUPPRESS_WARNINGS_KEY_UNCHECKED
public static KEY_GENERIC void quickSort(final KEY_GENERIC_TYPE[][] x, final long from, final long to) {
 final long len = to - from;

 // Selection sort on smallest arrays
 if (len < SMALL) {
  selectionSort(x, from, to);
  return;
 }

 // Choose a partition element, v
 long m = from + len / 2;  // Small arrays, middle element
 if (len > SMALL) {
  long l = from;
  long n = to - 1;
  if (len > MEDIUM) {  // Big arrays, pseudomedian of 9
   long s = len / 8;
   l = med3(x, l, l + s, l + 2 * s);
   m = med3(x, m - s, m, m + s);
   n = med3(x, n - 2 * s, n - s, n);
  }
  m = med3(x, l, m, n); // Mid-size, med of 3
 }

 final KEY_GENERIC_TYPE v = get(x, m);

 // Establish Invariant: v* (<v)* (>v)* v*
 long a = from, b = a, c = to - 1, d = c;
 while(true) {
  int comparison;
  while (b <= c && (comparison = KEY_CMP(get(x, b), v)) <= 0) {
   if (comparison == 0) swap(x, a++, b);
   b++;
  }
  while (c >= b && (comparison = KEY_CMP(get(x, c), v)) >=0) {
   if (comparison == 0) swap(x, c, d--);
   c--;
```

```
 }
 if (b > c) break;
 swap(x, b++, c--);
 }

 // Swap partition elements back to middle
 long s, n = to;
 s = Math.min(a - from, b - a);
 vecSwap(x, from, b - s, s);
 s = Math.min(d - c, n - d- 1);
 vecSwap(x, b, n - s, s);

 // Recursively sort non-partition-elements
 if ((s = b - a) > 1) quickSort(x, from, from + s);
 if ((s = d - c) > 1) quickSort(x, n - s, n);

}


/** Sorts the specified big array according to the natural ascending order using quicksort.
 *
 * <p>The sorting algorithm is a tuned quicksort adapted from Jon L. Bentley and M. Douglas
 * McIlroy, &ldquo;Engineering a Sort Function&rdquo;, <i>Software: Practice and Experience</i>, 23(11), pages
 * 1249&minus;1265, 1993.
 *
 * @param x the big array to be sorted.
 */

public static KEY_GENERIC void quickSort(final KEY_GENERIC_TYPE[][] x) {
 quickSort(x, 0, BIG_ARRAYS.length(x));
}




#if ! KEY_CLASS_Boolean

/**
 * Searches a range of the specified big array for the specified value using
 * the binary search algorithm. The range must be sorted prior to making this call.
 * If it is not sorted, the results are undefined. If the range contains multiple elements with
 * the specified value, there is no guarantee which one will be found.
 *
 * @param a the big array to be searched.
 * @param from  the index of the first element (inclusive) to be searched.
 * @param to  the index of the last element (exclusive) to be searched.
 * @param key the value to be searched for.
 * @return index of the search key, if it is contained in the big array;
```

```
 *            otherwise, <tt>(-(<i>insertion point</i>) - 1)</tt>.  The <i>insertion
 *            point</i> is defined as the the point at which the value would
 *            be inserted into the big array: the index of the first
 *            element greater than the key, or the length of the big array, if all
 *            elements in the big array are less than the specified key.  Note
 *            that this guarantees that the return value will be &gt;= 0 if
 *            and only if the key is found.
 * @see java.util.Arrays
 */
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public static KEY_GENERIC long binarySearch(final KEY_GENERIC_TYPE[][] a, long from, long to, final
KEY_GENERIC_TYPE key) {
  KEY_GENERIC_TYPE midVal;
  to--;
  while (from <= to) {
   final long mid = (from + to) >>> 1;
   midVal = get(a, mid);
#if KEYS_PRIMITIVE
   if (midVal < key) from = mid + 1;
   else if (midVal > key) to = mid - 1;
   else return mid;
#else
   final int cmp = ((Comparable KEY_SUPER_GENERIC)midVal).compareTo(key);
   if (cmp < 0) from = mid + 1;
   else if (cmp > 0) to = mid - 1;
   else return mid;
#endif
  }
  return -(from + 1);
 }

 /**
  * Searches a big array for the specified value using
  * the binary search algorithm. The range must be sorted prior to making this call.
  * If it is not sorted, the results are undefined. If the range contains multiple elements with
  * the specified value, there is no guarantee which one will be found.
  *
  * @param a the big array to be searched.
  * @param key the value to be searched for.
  * @return index of the search key, if it is contained in the big array;
  *            otherwise, <tt>(-(<i>insertion point</i>) - 1)</tt>.  The <i>insertion
  *            point</i> is defined as the the point at which the value would
  *            be inserted into the big array: the index of the first
  *            element greater than the key, or the length of the big array, if all
  *            elements in the big array are less than the specified key.  Note
  *            that this guarantees that the return value will be &gt;= 0 if
  *            and only if the key is found.
  * @see java.util.Arrays
```

```
 */
 public static KEY_GENERIC long binarySearch(final KEY_GENERIC_TYPE[][] a, final KEY_TYPE key) {
  return binarySearch(a, 0, BIG_ARRAYS.length(a), key);
 }

 /**
  * Searches a range of the specified big array for the specified value using
  * the binary search algorithm and a specified comparator. The range must be sorted following the comparator prior to making this call.
  * If it is not sorted, the results are undefined. If the range contains multiple elements with
  * the specified value, there is no guarantee which one will be found.
  *
  * @param a the big array to be searched.
  * @param from  the index of the first element (inclusive) to be searched.
  * @param to  the index of the last element (exclusive) to be searched.
  * @param key the value to be searched for.
  * @param c a comparator.
  * @return index of the search key, if it is contained in the big array;
  *           otherwise, <tt>(-(<i>insertion point</i>) - 1)</tt>.  The <i>insertion
  *           point</i> is defined as the the point at which the value would
  *           be inserted into the big array: the index of the first
  *           element greater than the key, or the length of the big array, if all
  *           elements in the big array are less than the specified key.  Note
  *           that this guarantees that the return value will be &gt;= 0 if
  *           and only if the key is found.
  * @see java.util.Arrays
  */
 public static KEY_GENERIC long binarySearch(final KEY_GENERIC_TYPE[][] a, long from, long to, final KEY_GENERIC_TYPE key, final KEY_COMPARATOR KEY_GENERIC c) {
  KEY_GENERIC_TYPE midVal;
  to--;
  while (from <= to) {
   final long mid = (from + to) >>> 1;
   midVal = get(a, mid);
   final int cmp = c.compare(midVal, key);
   if (cmp < 0) from = mid + 1;
   else if (cmp > 0) to = mid - 1;
   else return mid; // key found
  }
  return -(from + 1);
 }

 /**
  * Searches a big array for the specified value using
  * the binary search algorithm and a specified comparator. The range must be sorted following the comparator prior to making this call.
  * If it is not sorted, the results are undefined. If the range contains multiple elements with
  * the specified value, there is no guarantee which one will be found.
```

```
 *
 * @param a the big array to be searched.
 * @param key the value to be searched for.
 * @param c a comparator.
 * @return index of the search key, if it is contained in the big array;
 *         otherwise, <tt>(-(<i>insertion point</i>) - 1)</tt>.  The <i>insertion
 *         point</i> is defined as the the point at which the value would
 *         be inserted into the big array: the index of the first
 *         element greater than the key, or the length of the big array, if all
 *         elements in the big array are less than the specified key.  Note
 *         that this guarantees that the return value will be &gt;= 0 if
 *         and only if the key is found.
 * @see java.util.Arrays
 */
 public static KEY_GENERIC long binarySearch(final KEY_GENERIC_TYPE[][] a, final KEY_GENERIC_TYPE
key, final KEY_COMPARATOR KEY_GENERIC c) {
  return binarySearch(a, 0, BIG_ARRAYS.length(a), key, c);
 }


#if KEYS_PRIMITIVE
 /** The size of a digit used during radix sort (must be a power of 2). */
 private static final int DIGIT_BITS = 8;
 /** The mask to extract a digit of {@link #DIGIT_BITS} bits. */
 private static final int DIGIT_MASK = (1 << DIGIT_BITS) - 1;
 /** The number of digits per element. */
 private static final int DIGITS_PER_ELEMENT = KEY_CLASS.SIZE / DIGIT_BITS;

 /** This method fixes negative numbers so that the combination exponent/significand is lexicographically sorted. */
#if KEY_CLASS_Double
 private static final long fixDouble(final double d) {
  final long l = Double.doubleToRawLongBits(d);
  return l >= 0 ? l : l ^ 0x7FFFFFFFFFFFFFFFL;
 }
#elif KEY_CLASS_Float
 private static final long fixFloat(final float f) {
  final long i = Float.floatToRawIntBits(f);
  return i >= 0 ? i : i ^ 0x7FFFFFFF;
 }
#endif


 /** Sorts the specified big array using radix sort.
  *
  * <p>The sorting algorithm is a tuned radix sort adapted from Peter M. McIlroy, Keith Bostic and M. Douglas
  * McIlroy, &ldquo;Engineering radix sort&rdquo;, <i>Computing Systems</i>, 6(1), pages 5&minus;27 (1993),
  * and further improved using the digit-oracle idea described by
  * Juha K&auml;rkk&auml;inen and Tommi Rantala in &ldquo;Engineering radix sort for strings&rdquo;,
```

```
 * <i>String Processing and Information Retrieval, 15th International Symposium</i>, volume 5280 of
 * Lecture Notes in Computer Science, pages 3&minus;14, Springer (2008).
 *
 * <p>This implementation is significantly faster than quicksort
 * already at small sizes (say, more than 10000 elements), but it can only
 * sort in ascending order.
 * It will allocate a support array of bytes with the same number of elements as the array to be sorted.
 *
 * @param a the big array to be sorted.
 */
public static void radixSort(final KEY_TYPE[][] a) {
 radixSort(a, 0, BIG_ARRAYS.length(a));
}

/** Sorts the specified big array using radix sort.
 *
 * <p>The sorting algorithm is a tuned radix sort adapted from Peter M. McIlroy, Keith Bostic and M. Douglas
 * McIlroy, &ldquo;Engineering radix sort&rdquo;, <i>Computing Systems</i>, 6(1), pages 5&minus;27 (1993),
 * and further improved using the digit-oracle idea described by
 * Juha K&auml;rkk&auml;inen and Tommi Rantala in &ldquo;Engineering radix sort for strings&rdquo;,
 * <i>String Processing and Information Retrieval, 15th International Symposium</i>, volume 5280 of
 * Lecture Notes in Computer Science, pages 3&minus;14, Springer (2008).
 *
 * <p>This implementation is significantly faster than quicksort
 * already at small sizes (say, more than 10000 elements), but it can only
 * sort in ascending order.
 * It will allocate a support array of bytes with the same number of elements as the array to be sorted.
 *
 * @param a the big array to be sorted.
 * @param from the index of the first element (inclusive) to be sorted.
 * @param to the index of the last element (exclusive) to be sorted.
 */
public static void radixSort(final KEY_TYPE[][] a, final long from, final long to) {
 final int maxLevel = DIGITS_PER_ELEMENT - 1;

 final int stackSize = ((1 << DIGIT_BITS) - 1) * (DIGITS_PER_ELEMENT - 1) + 1;
 final long[] offsetStack = new long[stackSize];
 int offsetPos = 0;
 final long[] lengthStack = new long[stackSize];
 int lengthPos = 0;
 final int[] levelStack = new int[stackSize];
 int levelPos = 0;

 offsetStack[offsetPos++] = from;
 lengthStack[lengthPos++] = to - from;
 levelStack[levelPos++] = 0;

 final long[] count = new long[1 << DIGIT_BITS];
```

```
  final long[] pos = new long[1 << DIGIT_BITS];
  final byte[][] digit = ByteBigArrays.newBigArray(to - from);

  while(offsetPos > 0) {
   final long first = offsetStack[--offsetPos];
   final long length = lengthStack[--lengthPos];
   final int level = levelStack[--levelPos];
#if KEY_CLASS_Character
   final int signMask = 0;
#else
   final int signMask = level % DIGITS_PER_ELEMENT == 0 ? 1 << DIGIT_BITS - 1 : 0;
#endif

   if (length < MEDIUM) {
    selectionSort(a, first, first + length);
    continue;
   }

   final int shift = (DIGITS_PER_ELEMENT - 1 - level % DIGITS_PER_ELEMENT) * DIGIT_BITS; // This is the shift that extract the right byte from a key

   // Count keys.

   for(long i = length; i-- != 0;) ByteBigArrays.set(digit, i, (byte)(((KEY2LEXINT(BIG_ARRAYS.get(a, first + i)) >>> shift) & DIGIT_MASK) ^ signMask));
   for(long i = length; i-- != 0;) count[ByteBigArrays.get(digit, i) & 0xFF]++;
   // Compute cumulative distribution and push non-singleton keys on stack.
   int lastUsed = -1;

   long p = 0;
   for(int i = 0; i < 1 << DIGIT_BITS; i++) {
    if (count[i] != 0) {
     lastUsed = i;
     if (level < maxLevel && count[i] > 1){
      //System.err.println(" Pushing " + new StackEntry(first + pos[i - 1], first + pos[i], level + 1));
      offsetStack[offsetPos++] = p + first;
      lengthStack[lengthPos++] = count[i];
      levelStack[levelPos++] = level + 1;
     }
    }
    pos[i] = (p += count[i]);
   }

   // When all slots are OK, the last slot is necessarily OK.
   final long end = length - count[lastUsed];
   count[lastUsed] = 0;

   // i moves through the start of each block
```

```
  int c = -1;
  for(long i = 0, d; i < end; i += count[c], count[c] = 0) {
   KEY_TYPE t = BIG_ARRAYS.get(a, i +first);
   c = ByteBigArrays.get(digit, i) & 0xFF;
   while((d = --pos[c]) > i) {
    final KEY_TYPE z = t;
    final int zz = c;
    t = BIG_ARRAYS.get(a, d + first);
    c = ByteBigArrays.get(digit, d) & 0xFF;
    BIG_ARRAYS.set(a, d + first, z);
    ByteBigArrays.set(digit, d, (byte)zz);
   }

   BIG_ARRAYS.set(a, i + first, t);
  }
 }
}


 private static void selectionSort(final KEY_TYPE[][] a, final KEY_TYPE[][] b, final long from, final long to) {
  for(long i = from; i < to - 1; i++) {
   long m = i;
   for(long j = i + 1; j < to; j++)
    if (KEY_LESS(BIG_ARRAYS.get(a, j), BIG_ARRAYS.get(a, m)) || KEY_CMP_EQ(BIG_ARRAYS.get(a, j),
BIG_ARRAYS.get(a, m)) && KEY_LESS(BIG_ARRAYS.get(b, j), BIG_ARRAYS.get(b, m))) m = j;

   if (m != i) {
    KEY_TYPE t = BIG_ARRAYS.get(a, i);
    BIG_ARRAYS.set(a, i, BIG_ARRAYS.get(a, m));
    BIG_ARRAYS.set(a, m, t);
    t = BIG_ARRAYS.get(b, i);
    BIG_ARRAYS.set(b, i, BIG_ARRAYS.get(b, m));
    BIG_ARRAYS.set(b, m, t);
   }
  }
 }

 /** Sorts the specified pair of big arrays lexicographically using radix sort.
  * <p>The sorting algorithm is a tuned radix sort adapted from Peter M. McIlroy, Keith Bostic and M. Douglas
  * McIlroy, &ldquo;Engineering radix sort&rdquo;, <i>Computing Systems</i>, 6(1), pages 5&minus;27 (1993),
  * and further improved using the digit-oracle idea described by
  * Juha K&auml;rkk&auml;inen and Tommi Rantala in &ldquo;Engineering radix sort for strings&rdquo;,
  * <i>String Processing and Information Retrieval, 15th International Symposium</i>, volume 5280 of
  * Lecture Notes in Computer Science, pages 3&minus;14, Springer (2008).
  *
  * <p>This method implements a <em>lexicographical</em> sorting of the arguments. Pairs of elements
  * in the same position in the two provided arrays will be considered a single key, and permuted
  * accordingly. In the end, either {@code a[i] &lt; a[i + 1]} or {@code a[i] == a[i + 1]} and {@code b[i] &lt;= b[i +
```

```
1]}.
 *
 * <p>This implementation is significantly faster than quicksort
 * already at small sizes (say, more than 10000 elements), but it can only
 * sort in ascending order. It will allocate a support array of bytes with the same number of elements as the arrays to be sorted.
 *
 * @param a the first big array to be sorted.
 * @param b the second big array to be sorted.
 */

public static void radixSort(final KEY_TYPE[][] a, final KEY_TYPE[][] b) {
 radixSort(a, b, 0, BIG_ARRAYS.length(a));
}

/** Sorts the specified pair of big arrays lexicographically using radix sort.
 *
 * <p>The sorting algorithm is a tuned radix sort adapted from Peter M. McIlroy, Keith Bostic and M. Douglas
 * McIlroy, &ldquo;Engineering radix sort&rdquo;, <i>Computing Systems</i>, 6(1), pages 5&minus;27 (1993),
 * and further improved using the digit-oracle idea described by
 * Juha K&auml;rkk&auml;inen and Tommi Rantala in &ldquo;Engineering radix sort for strings&rdquo;,
 * <i>String Processing and Information Retrieval, 15th International Symposium</i>, volume 5280 of
 * Lecture Notes in Computer Science, pages 3&minus;14, Springer (2008).
 *
 * <p>This method implements a <em>lexicographical</em> sorting of the arguments. Pairs of elements
 * in the same position in the two provided arrays will be considered a single key, and permuted
 * accordingly. In the end, either {@code a[i] &lt; a[i + 1]} or {@code a[i] == a[i + 1]} and {@code b[i] &lt;= b[i + 1]}.
 *
 * <p>This implementation is significantly faster than quicksort
 * already at small sizes (say, more than 10000 elements), but it can only
 * sort in ascending order. It will allocate a support array of bytes with the same number of elements as the arrays to be sorted.
 *
 * @param a the first big array to be sorted.
 * @param b the second big array to be sorted.
 * @param from the index of the first element (inclusive) to be sorted.
 * @param to the index of the last element (exclusive) to be sorted.
 */
public static void radixSort(final KEY_TYPE[][] a, final KEY_TYPE[][] b, final long from, final long to) {
 final int layers = 2;
 if (BIG_ARRAYS.length(a) != BIG_ARRAYS.length(b)) throw new IllegalArgumentException("Array size mismatch.");
 final int maxLevel = DIGITS_PER_ELEMENT * layers - 1;

 final int stackSize = ((1 << DIGIT_BITS) - 1) * (layers * DIGITS_PER_ELEMENT - 1) + 1;
 final long[] offsetStack = new long[stackSize];
 int offsetPos = 0;
```

```
  final long[] lengthStack = new long[stackSize];
  int lengthPos = 0;
  final int[] levelStack = new int[stackSize];
  int levelPos = 0;

  offsetStack[offsetPos++] = from;
  lengthStack[lengthPos++] = to - from;
  levelStack[levelPos++] = 0;

  final long[] count = new long[1 << DIGIT_BITS];
  final long[] pos = new long[1 << DIGIT_BITS];
  final byte[][] digit = ByteBigArrays.newBigArray(to - from);

  while(offsetPos > 0) {
   final long first = offsetStack[--offsetPos];
   final long length = lengthStack[--lengthPos];
   final int level = levelStack[--levelPos];
#if KEY_CLASS_Character
   final int signMask = 0;
#else
   final int signMask = level % DIGITS_PER_ELEMENT == 0 ? 1 << DIGIT_BITS - 1 : 0;
#endif

   if (length < MEDIUM) {
    selectionSort(a, b, first, first + length);
    continue;
   }

   final KEY_TYPE[][] k = level < DIGITS_PER_ELEMENT ? a : b; // This is the key array
   final int shift = (DIGITS_PER_ELEMENT - 1 - level % DIGITS_PER_ELEMENT) * DIGIT_BITS; // This is the
shift that extract the right byte from a key

   // Count keys.
   for(long i = length; i-- != 0;) ByteBigArrays.set(digit, i, (byte)(((KEY2LEXINT(BIG_ARRAYS.get(k, first + i))
>>> shift) & DIGIT_MASK) ^ signMask));
   for(long i = length; i-- != 0;) count[ByteBigArrays.get(digit, i) & 0xFF]++;
   // Compute cumulative distribution and push non-singleton keys on stack.
   int lastUsed = -1;

   long p = 0;
   for(int i = 0; i < 1 << DIGIT_BITS; i++) {
    if (count[i] != 0) {
     lastUsed = i;
     if (level < maxLevel && count[i] > 1){
      offsetStack[offsetPos++] = p + first;
      lengthStack[lengthPos++] = count[i];
      levelStack[levelPos++] = level + 1;
     }
```

```
   }
   pos[i] = (p += count[i]);
   }

   // When all slots are OK, the last slot is necessarily OK.
   final long end = length - count[lastUsed];
   count[lastUsed] = 0;

   // i moves through the start of each block
   int c = -1;
   for(long i = 0, d; i < end; i += count[c], count[c] = 0) {
    KEY_TYPE t = BIG_ARRAYS.get(a, i + first);
    KEY_TYPE u = BIG_ARRAYS.get(b, i + first);
    c = ByteBigArrays.get(digit, i) & 0xFF;
    while((d = --pos[c]) > i) {
     KEY_TYPE z = t;
     final int zz = c;
     t = BIG_ARRAYS.get(a, d + first);
     BIG_ARRAYS.set(a, d + first, z);
     z = u;
     u = BIG_ARRAYS.get(b, d + first);
     BIG_ARRAYS.set(b, d + first, z);
     c = ByteBigArrays.get(digit, d) & 0xFF;
     ByteBigArrays.set(digit, d, (byte)zz);
    }

    BIG_ARRAYS.set(a, i + first, t);
    BIG_ARRAYS.set(b, i + first, u);
   }
  }
 }

#endif

#endif


 /** Shuffles the specified big array fragment using the specified pseudorandom number generator.
  *
  * @param a the big array to be shuffled.
  * @param from the index of the first element (inclusive) to be shuffled.
  * @param to the index of the last element (exclusive) to be shuffled.
  * @param random a pseudorandom number generator.
  * @return {@code a}.
  */
 public static KEY_GENERIC KEY_GENERIC_TYPE[][] shuffle(final KEY_GENERIC_TYPE[][] a, final long
from, final long to, final Random random) {
  for(long i = to - from; i-- != 0;) {
```

```
  final long p = (random.nextLong() & 0x7FFFFFFFFFFFFFFFL) % (i + 1);
  final KEY_GENERIC_TYPE t = get(a, from + i);
  set(a, from + i, get(a, from + p));
  set(a, from + p, t);
 }
 return a;
}

/** Shuffles the specified big array using the specified pseudorandom number generator.
 *
 * @param a the big array to be shuffled.
 * @param random a pseudorandom number generator.
 * @return {@code a}.
 */
public static KEY_GENERIC KEY_GENERIC_TYPE[][] shuffle(final KEY_GENERIC_TYPE[][] a, final Random
random) {
 for(long i = length(a); i-- != 0;) {
  final long p = (random.nextLong() & 0x7FFFFFFFFFFFFFFFL) % (i + 1);
  final KEY_GENERIC_TYPE t = get(a, i);
  set(a, i, get(a, p));
  set(a, p, t);
 }
 return a;
}


#if KEY_CLASS_Integer
#ifdef TEST

private static long seed = System.currentTimeMillis();
private static java.util.Random r = new java.util.Random(seed);

private static KEY_TYPE genKey() {
#if KEY_CLASS_Byte || KEY_CLASS_Short || KEY_CLASS_Character
 return (KEY_TYPE)(r.nextInt());
#elif KEYS_PRIMITIVE
 return r.NEXT_KEY();
#elif KEY_CLASS_Object
 return Integer.toBinaryString(r.nextInt());
#else
 return new java.io.Serializable() {};
#endif
}

private static Object[] k, v, nk;
private static KEY_TYPE kt[];
private static KEY_TYPE nkt[];
private static BIG_ARRAY_BIG_LIST topList;
```

```
 protected static void speedTest(int n, boolean b) {}

 protected static void runTest(int n) {
  KEY_TYPE[][] a = BIG_ARRAYS.newBigArray(n);
  for(int i = 0; i < n; i++) set(a, i, i);
  BIG_ARRAYS.copy(a, 0, a, 1, n - 2);
  assert a[0][0] == 0;
  for(int i = 0; i < n - 2; i++) assert get(a, i + 1) == i;

  for(int i = 0; i < n; i++) set(a, i, i);
  BIG_ARRAYS.copy(a, 1, a, 0, n - 1);
  for(int i = 0; i < n - 1; i++) assert get(a, i) == i + 1;

  for(int i = 0; i < n; i++) set(a, i, i);
  KEY_TYPE[] b = new KEY_TYPE[n];
  for(int i = 0; i < n; i++) b[i] = i;

  assert equals(wrap(b), a);

  System.out.println("Test OK");
  return;

 }


 public static void main(String args[]) {
  int n  = Integer.parseInt(args[1]);
  if (args.length > 2) r = new java.util.Random(seed = Long.parseLong(args[2]));

  try {
   if ("speedTest".equals(args[0]) || "speedComp".equals(args[0])) speedTest(n, "speedComp".equals(args[0]));
   else if ("test".equals(args[0])) runTest(n);
  } catch(Throwable e) {
   e.printStackTrace(System.err);
   System.err.println("seed: " + seed);
  }
 }

#endif
#endif

}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/BigArrays.drv

No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2007-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import java.util.Map;
import java.util.NoSuchElementException;
import it.unimi.dsi.fastutil.objects.AbstractObjectSet;
import it.unimi.dsi.fastutil.objects.ObjectIterator;

import VALUE_PACKAGE.VALUE_COLLECTION;
import VALUE_PACKAGE.VALUE_ABSTRACT_COLLECTION;
import VALUE_PACKAGE.VALUE_ARRAYS;

/** A simple, brute-force implementation of a map based on two parallel backing arrays.
*
* <p>The main purpose of this
* implementation is that of wrapping cleanly the brute-force approach to the storage of a very
* small number of pairs: just put them into two parallel arrays and scan linearly to find an item.
*/

public class ARRAY_MAP KEY_VALUE_GENERIC extends ABSTRACT_MAP KEY_VALUE_GENERIC
implements java.io.Serializable, Cloneable {

 private static final long serialVersionUID = 1L;
 /** The keys (valid up to {@link #size}, excluded). */
 private transient KEY_TYPE[] key;
 /** The values (parallel to {@link #key}). */
 private transient VALUE_TYPE[] value;
 /** The number of valid entries in {@link #key} and {@link #value}. */
 private int size;
```

```
 /** Creates a new empty array map with given key and value backing arrays. The resulting map will have as many entries as the given arrays.
  *
  * <p>It is responsibility of the caller that the elements of {@code key} are distinct.
  *
  * @param key the key array.
  * @param value the value array (it <em>must</em> have the same length as {@code key}).
  */
 public ARRAY_MAP(final KEY_TYPE[] key, final VALUE_TYPE[] value) {
  this.key = key;
  this.value = value;
  size = key.length;
  if(key.length != value.length) throw new IllegalArgumentException("Keys and values have different lengths (" + key.length + ", " + value.length + ")");
 }

 /** Creates a new empty array map.
  */
 public ARRAY_MAP() {
  this.key = ARRAYS.EMPTY_ARRAY;
  this.value = VALUE_ARRAYS.EMPTY_ARRAY;
 }

 /** Creates a new empty array map of given capacity.
  *
  * @param capacity the initial capacity.
  */
 public ARRAY_MAP(final int capacity) {
  this.key = new KEY_TYPE[capacity];
  this.value = new VALUE_TYPE[capacity];
 }

 /** Creates a new empty array map copying the entries of a given map.
  *
  * @param m a map.
  */
 public ARRAY_MAP(final MAP KEY_VALUE_GENERIC m) {
  this(m.size());
  putAll(m);
 }

 /** Creates a new empty array map copying the entries of a given map.
  *
  * @param m a map.
  */
 public ARRAY_MAP(final Map<? extends KEY_GENERIC_CLASS, ? extends VALUE_GENERIC_CLASS> m) {
  this(m.size());
```

```
 putAll(m);
 }

 /** Creates a new array map with given key and value backing arrays, using the given number of elements.
  *
  * <p>It is responsibility of the caller that the first {@code size} elements of {@code key} are distinct.
  *
  * @param key the key array.
  * @param value the value array (it <em>must</em> have the same length as {@code key}).
  * @param size the number of valid elements in {@code key} and {@code value}.
  */
 public ARRAY_MAP(final KEY_TYPE[] key, final VALUE_TYPE[] value, final int size) {
  this.key = key;
  this.value = value;
  this.size = size;
  if(key.length != value.length) throw new IllegalArgumentException("Keys and values have different lengths (" +
key.length + ", " + value.length + ")");
  if (size > key.length) throw new IllegalArgumentException("The provided size (" + size + ") is larger than or equal
to the backing-arrays size (" + key.length + ")");
 }

 private final class EntrySet extends AbstractObjectSet<MAP.Entry KEY_VALUE_GENERIC> implements
FastEntrySet KEY_VALUE_GENERIC {

  @Override
  public ObjectIterator<MAP.Entry KEY_VALUE_GENERIC> iterator() {
   return new ObjectIterator<MAP.Entry KEY_VALUE_GENERIC>() {
    int curr = -1, next = 0;

    @Override
    public boolean hasNext() { return next < size; }

    @Override
    SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
    public Entry KEY_VALUE_GENERIC next() {
     if (! hasNext()) throw new NoSuchElementException();
     return new ABSTRACT_MAP.BasicEntry KEY_VALUE_GENERIC_DIAMOND(KEY_GENERIC_CAST
key[curr = next], VALUE_GENERIC_CAST value[next++]);
    }

    @Override
    public void remove() {
     if (curr == -1) throw new IllegalStateException();
     curr = -1;
     final int tail = size-- - next--;
     System.arraycopy(key, next + 1, key, next, tail);
     System.arraycopy(value, next + 1, value, next, tail);
#if KEYS_REFERENCE
```

```
    key[size] = null;
#endif
#if VALUES_REFERENCE
    value[size] = null;
#endif
   }
  };
 }

 @Override
 public ObjectIterator<MAP.Entry KEY_VALUE_GENERIC> fastIterator() {
  return new ObjectIterator<MAP.Entry KEY_VALUE_GENERIC>() {
   int next = 0, curr = -1;
   final BasicEntry KEY_VALUE_GENERIC entry = new BasicEntry KEY_VALUE_GENERIC_DIAMOND ();

   @Override
   public boolean hasNext() { return next < size; }

   @Override
   SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
   public Entry KEY_VALUE_GENERIC next() {
    if (! hasNext()) throw new NoSuchElementException();
    entry.key = KEY_GENERIC_CAST key[curr = next];
    entry.value = VALUE_GENERIC_CAST value[next++];
    return entry;
   }

   @Override
   public void remove() {
    if (curr == -1) throw new IllegalStateException();
    curr = -1;
    final int tail = size-- - next--;
    System.arraycopy(key, next + 1, key, next, tail);
    System.arraycopy(value, next + 1, value, next, tail);
#if KEYS_REFERENCE
    key[size] = null;
#endif
#if VALUES_REFERENCE
    value[size] = null;
#endif
   }
  };
 }

 @Override
 public int size() { return size; }

 @Override
```

```
  SUPPRESS_WARNINGS_KEY_UNCHECKED
  public boolean contains(Object o) {
   if (! (o instanceof Map.Entry)) return false;
   final Map.Entry<?,?> e = (Map.Entry<?,?>)o;
#if KEYS_PRIMITIVE
   if (e.getKey() == null || ! (e.getKey() instanceof KEY_CLASS)) return false;
#endif
#if VALUES_PRIMITIVE
   if (e.getValue() == null || ! (e.getValue() instanceof VALUE_CLASS)) return false;
#endif
   final KEY_GENERIC_TYPE k = KEY_OBJ2TYPE(KEY_GENERIC_CAST e.getKey());
   return ARRAY_MAP.this.containsKey(k) && VALUE_EQUALS(ARRAY_MAP.this.GET_VALUE(k),
VALUE_OBJ2TYPE(e.getValue()));
  }

  @Override
  SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
  public boolean remove(final Object o) {
   if (!(o instanceof Map.Entry)) return false;
   final Map.Entry<?,?> e = (Map.Entry<?,?>)o;
#if KEYS_PRIMITIVE
   if (e.getKey() == null || ! (e.getKey() instanceof KEY_CLASS)) return false;
#endif
#if VALUES_PRIMITIVE
   if (e.getValue() == null || ! (e.getValue() instanceof VALUE_CLASS)) return false;
#endif
   final KEY_GENERIC_TYPE k = KEY_OBJ2TYPE(KEY_GENERIC_CAST e.getKey());
   final VALUE_GENERIC_TYPE v = VALUE_OBJ2TYPE(VALUE_GENERIC_CAST e.getValue());

   final int oldPos = ARRAY_MAP.this.findKey(k);
   if (oldPos == -1 || ! VALUE_EQUALS(v, ARRAY_MAP.this.value[oldPos])) return false;
   final int tail = size - oldPos - 1;
   System.arraycopy(ARRAY_MAP.this.key, oldPos + 1, ARRAY_MAP.this.key, oldPos, tail);
   System.arraycopy(ARRAY_MAP.this.value, oldPos + 1, ARRAY_MAP.this.value, oldPos, tail);
   ARRAY_MAP.this.size--;
#if KEYS_REFERENCE
   ARRAY_MAP.this.key[size] = null;
#endif
#if VALUES_REFERENCE
   ARRAY_MAP.this.value[size] = null;
#endif
   return true;
  }
 }

 @Override
 public FastEntrySet KEY_VALUE_GENERIC ENTRYSET() { return new EntrySet(); }
```

```
private int findKey(final KEY_TYPE k) {
 final KEY_TYPE[] key = this.key;
 for(int i = size; i-- != 0;) if (KEY_EQUALS(key[i], k)) return i;
 return -1;
}

@Override
SUPPRESS_WARNINGS_VALUE_UNCHECKED
public VALUE_GENERIC_TYPE GET_VALUE(final KEY_TYPE k) {
 final KEY_TYPE[] key = this.key;
 for(int i = size; i-- != 0;) if (KEY_EQUALS(key[i], k)) return VALUE_GENERIC_CAST value[i];
 return defRetValue;
}

@Override
public int size() { return size; }

@Override
public void clear() {
#if KEYS_REFERENCE || VALUES_REFERENCE
 for(int i = size; i-- != 0;) {
#if KEYS_REFERENCE
  key[i] = null;
#endif
#if VALUES_REFERENCE
  value[i] = null;
#endif
 }
#endif
 size = 0;
}

@Override
public boolean containsKey(final KEY_TYPE k) { return findKey(k) != -1; }

@Override
public boolean containsValue(VALUE_TYPE v) {
 for(int i = size; i-- != 0;) if (VALUE_EQUALS(value[i], v)) return true;
 return false;
}

@Override
public boolean isEmpty() { return size == 0; }

@Override
SUPPRESS_WARNINGS_VALUE_UNCHECKED
public VALUE_GENERIC_TYPE put(KEY_GENERIC_TYPE k, VALUE_GENERIC_TYPE v) {
 final int oldKey = findKey(k);
```

```
 if (oldKey != -1) {
  final VALUE_GENERIC_TYPE oldValue = VALUE_GENERIC_CAST value[oldKey];
  value[oldKey] = v;
  return oldValue;
 }
 if (size == key.length) {
  final KEY_TYPE[] newKey = new KEY_TYPE[size == 0 ? 2 : size * 2];
  final VALUE_TYPE[] newValue = new VALUE_TYPE[size == 0 ? 2 : size * 2];
  for(int i = size; i-- != 0;) {
   newKey[i] = key[i];
   newValue[i] = value[i];
  }
  key = newKey;
  value = newValue;
 }
 key[size] = k;
 value[size] = v;
 size++;
 return defRetValue;
}

@Override
SUPPRESS_WARNINGS_VALUE_UNCHECKED
public VALUE_GENERIC_TYPE REMOVE_VALUE(final KEY_TYPE k) {
 final int oldPos = findKey(k);
 if (oldPos == -1) return defRetValue;
 final VALUE_GENERIC_TYPE oldValue = VALUE_GENERIC_CAST value[oldPos];
 final int tail = size - oldPos - 1;
 System.arraycopy(key, oldPos + 1, key, oldPos, tail);
 System.arraycopy(value, oldPos + 1, value, oldPos, tail);
 size--;
#if KEYS_REFERENCE
 key[size] = null;
#endif
#if VALUES_REFERENCE
 value[size] = null;
#endif
 return oldValue;
}

@Override
public SET KEY_GENERIC keySet() {
 return new ABSTRACT_SET KEY_GENERIC() {
  @Override
  public boolean contains(final KEY_TYPE k) {
   return findKey(k) != -1;
  }
```

```
@Override
public boolean remove(final KEY_TYPE k) {
  final int oldPos = findKey(k);
  if (oldPos == -1) return false;
  final int tail = size - oldPos - 1;
  System.arraycopy(key, oldPos + 1, key, oldPos, tail);
  System.arraycopy(value, oldPos + 1, value, oldPos, tail);
  size--;
  return true;
 }

 @Override
 public KEY_ITERATOR KEY_GENERIC iterator() {
  return new KEY_ITERATOR KEY_GENERIC() {
   int pos = 0;
   @Override
   public boolean hasNext() {
    return pos < size;
   }

   @Override
   SUPPRESS_WARNINGS_KEY_UNCHECKED
   public KEY_GENERIC_TYPE NEXT_KEY() {
    if (! hasNext()) throw new NoSuchElementException();
    return KEY_GENERIC_CAST key[pos++];
   }

   @Override
   public void remove() {
    if (pos == 0) throw new IllegalStateException();
    final int tail = size - pos;
    System.arraycopy(key, pos, key, pos - 1, tail);
    System.arraycopy(value, pos, value, pos - 1, tail);
    size--;
   }
  };
 }

 @Override
 public int size() {
  return size;
 }

 @Override
 public void clear() {
  ARRAY_MAP.this.clear();
 }
};
```

```
}

@Override
public VALUE_COLLECTION VALUE_GENERIC values() {
 return new VALUE_ABSTRACT_COLLECTION VALUE_GENERIC() {

  @Override
  public boolean contains(final VALUE_TYPE v) {
   return containsValue(v);
  }

  @Override
  public VALUE_PACKAGE.VALUE_ITERATOR VALUE_GENERIC iterator() {
   return new VALUE_PACKAGE.VALUE_ITERATOR VALUE_GENERIC() {
    int pos = 0;
    @Override
    public boolean hasNext() {
     return pos < size;
    }

    @Override
    SUPPRESS_WARNINGS_VALUE_UNCHECKED
    public VALUE_GENERIC_TYPE NEXT_VALUE() {
     if (! hasNext()) throw new NoSuchElementException();
     return VALUE_GENERIC_CAST value[pos++];
    }

    @Override
    public void remove() {
     if (pos == 0) throw new IllegalStateException();
     final int tail = size - pos;
     System.arraycopy(key, pos, key, pos - 1, tail);
     System.arraycopy(value, pos, value, pos - 1, tail);
     size--;
    }
   };
  }

  @Override
  public int size() {
   return size;
  }

  @Override
  public void clear() {
   ARRAY_MAP.this.clear();
  }
 };
```

```
}

/** Returns a deep copy of this map.
 *
 * <p>This method performs a deep copy of this hash map; the data stored in the
 * map, however, is not cloned. Note that this makes a difference only for object keys.
 *
 *  @return a deep copy of this map.
 */
@Override
SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
public ARRAY_MAP KEY_VALUE_GENERIC clone() {
 ARRAY_MAP KEY_VALUE_GENERIC c;
 try {
  c = (ARRAY_MAP KEY_VALUE_GENERIC)super.clone();
 }
 catch(CloneNotSupportedException cantHappen) {
  throw new InternalError();
 }
 c.key = key.clone();
 c.value = value.clone();
 return c;
}

private void writeObject(java.io.ObjectOutputStream s) throws java.io.IOException {
 s.defaultWriteObject();
 for(int i = 0; i < size; i++) {
  s.WRITE_KEY(key[i]);
  s.WRITE_VALUE(value[i]);
 }
}

private void readObject(java.io.ObjectInputStream s) throws java.io.IOException, ClassNotFoundException {
 s.defaultReadObject();
 key = new KEY_TYPE[size];
 value = new VALUE_TYPE[size];
 for(int i = 0; i < size; i++) {
  key[i] = s.READ_KEY();
  value[i] = s.READ_VALUE();
 }
}
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/ArrayMap.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2010-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

#if KEY_CLASS_Object
import java.util.Arrays;
import java.util.Comparator;
#endif

import java.io.Serializable;
import it.unimi.dsi.fastutil.HashCommon;
import it.unimi.dsi.fastutil.PriorityQueue;

import java.util.NoSuchElementException;

/** A type-specific array-based FIFO queue, supporting also deque operations.
*
* <p>Instances of this class represent a FIFO queue using a backing
* array in a circular way. The array is enlarged and shrunk as needed. You can use the {@link #trim()} method
* to reduce its memory usage, if necessary.
*
* <p>This class provides additional methods that implement a <em>deque</em> (double-ended queue).
*/

public class ARRAY_FIFO_QUEUE KEY_GENERIC implements PRIORITY_QUEUE KEY_GENERIC,
Serializable {
 private static final long serialVersionUID = 0L;

 /** The standard initial capacity of a queue. */
 public static final int INITIAL_CAPACITY = 4;

 /** The backing array. */
 protected transient KEY_GENERIC_TYPE array[];
```

```
 /** The current (cached) length of {@link #array}. */
 protected transient int length;

 /** The start position in {@link #array}. It is always strictly smaller than {@link #length}.*/
 protected transient int start;

 /** The end position in {@link #array}. It is always strictly smaller than {@link #length}.
  *  Might be actually smaller than {@link #start} because {@link #array} is used cyclically. */
 protected transient int end;

 /** Creates a new empty queue with given capacity.
  *
  * @param capacity the initial capacity of this queue.
  */
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public ARRAY_FIFO_QUEUE(final int capacity) {
  if (capacity < 0) throw new IllegalArgumentException("Initial capacity (" + capacity + ") is negative");
  array = KEY_GENERIC_ARRAY_CAST new KEY_TYPE[Math.max(1, capacity)]; // Never build a queue with
zero-sized backing array.
  length = array.length;
 }

 /** Creates a new empty queue with standard {@linkplain #INITIAL_CAPACITY initial capacity}. */
 public ARRAY_FIFO_QUEUE() {
  this(INITIAL_CAPACITY);
 }

 /** {@inheritDoc}
  * <p>This implementation returns {@code null} (FIFO queues have no comparator). */
 @Override
 public KEY_COMPARATOR KEY_SUPER_GENERIC comparator() {
  return null;
 }

 @Override
 public KEY_GENERIC_TYPE DEQUEUE() {
  if (start == end) throw new NoSuchElementException();
  final KEY_GENERIC_TYPE t = array[start];
#if KEYS_REFERENCE
  array[start] = null; // Clean-up for the garbage collector.
#endif
  if (++start == length) start = 0;
  reduce();
  return t;
 }

 /** Dequeues the {@linkplain PriorityQueue#last() last} element from the queue.
```

```
 *
 * @return the dequeued element.
 * @throws NoSuchElementException if the queue is empty.
 */
 public KEY_GENERIC_TYPE DEQUEUE_LAST() {
  if (start == end) throw new NoSuchElementException();
  if (end == 0) end = length;
  final KEY_GENERIC_TYPE t = array[--end];
#if KEYS_REFERENCE
  array[end] = null; // Clean-up for the garbage collector.
#endif
  reduce();
  return t;
 }

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 private final void resize(final int size, final int newLength) {
  final KEY_GENERIC_TYPE[] newArray = KEY_GENERIC_ARRAY_CAST new KEY_TYPE[newLength];
  if (start >= end) {
   if (size != 0) {
    System.arraycopy(array, start, newArray, 0, length - start);
    System.arraycopy(array, 0, newArray, length - start, end);
   }
  }
  else System.arraycopy(array, start, newArray, 0, end - start);
  start = 0;
  end = size;
  array = newArray;
  length = newLength;
 }

 private final void expand() {
  resize(length, (int)Math.min(it.unimi.dsi.fastutil.Arrays.MAX_ARRAY_SIZE, 2L * length));
 }

 private final void reduce() {
  final int size = size();
  if (length > INITIAL_CAPACITY && size <= length / 4) resize(size, length / 2);
 }

 @Override
 public void enqueue(KEY_GENERIC_TYPE x) {
  array[end++] = x;
  if (end == length) end = 0;
  if (end == start) expand();
 }

 /** Enqueues a new element as the first element (in dequeuing order) of the queue.
```

```
 * @param x the element to enqueue.
 */
public void enqueueFirst(KEY_GENERIC_TYPE x) {
 if (start == 0) start = length;
 array[--start] = x;
 if (end == start) expand();
}

@Override
public KEY_GENERIC_TYPE FIRST() {
 if (start == end) throw new NoSuchElementException();
 return array[start];
}


@Override
public KEY_GENERIC_TYPE LAST() {
 if (start == end) throw new NoSuchElementException();
 return array[(end == 0 ? length : end) - 1];
}

@Override
public void clear() {
#if KEYS_REFERENCE
 if (start <= end) Arrays.fill(array, start, end, null);
 else {
  Arrays.fill(array, start, length, null);
  Arrays.fill(array, 0, end, null);
 }
#endif
 start = end = 0;
}

/** Trims the queue to the smallest possible size. */
SUPPRESS_WARNINGS_KEY_UNCHECKED
public void trim() {
 final int size = size();
 final KEY_GENERIC_TYPE[] newArray =
#if KEYS_PRIMITIVE
        new KEY_GENERIC_TYPE[size + 1];
#else
        (KEY_GENERIC_TYPE[])new Object[size + 1];
#endif
 if (start <= end) System.arraycopy(array, start, newArray, 0, end - start);
 else {
  System.arraycopy(array, start, newArray, 0, length - start);
  System.arraycopy(array, 0, newArray, length - start, end);
 }
```

```
 start = 0;
 length = (end = size) + 1;
 array = newArray;
}

@Override
public int size() {
 final int apparentLength = end - start;
 return apparentLength >= 0 ? apparentLength : length + apparentLength;
}

private void writeObject(java.io.ObjectOutputStream s) throws java.io.IOException {
 s.defaultWriteObject();
 int size = size();
 s.writeInt(size);
 for(int i = start; size-- != 0;) {
  s.WRITE_KEY(array[i++]);
  if (i == length) i = 0;
 }
}

SUPPRESS_WARNINGS_KEY_UNCHECKED
private void readObject(java.io.ObjectInputStream s) throws java.io.IOException, ClassNotFoundException {
 s.defaultReadObject();
 end = s.readInt();
 array = KEY_GENERIC_ARRAY_CAST new KEY_TYPE[length = HashCommon.nextPowerOfTwo(end + 1)];
 for(int i = 0; i < end; i++) array[i] = KEY_GENERIC_CAST s.READ_KEY();
}
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/ArrayFIFOQueue.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
```

* limitations under the License.
*
*
*
* For the sorting and binary search code:
*
* Copyright (C) 1999 CERN - European Organization for Nuclear Research.
*
*   Permission to use, copy, modify, distribute and sell this software and
*   its documentation for any purpose is hereby granted without fee,
*   provided that the above copyright notice appear in all copies and that
*   both that copyright notice and this permission notice appear in
*   supporting documentation. CERN makes no representations about the
*   suitability of this software for any purpose. It is provided "as is"
*   without expressed or implied warranty.
*/

package PACKAGE;

import it.unimi.dsi.fastutil.Arrays;
import it.unimi.dsi.fastutil.Hash;
import java.util.Random;
import java.util.concurrent.ForkJoinPool;
import java.util.concurrent.RecursiveAction;

#if ! KEY_CLASS_Integer
import it.unimi.dsi.fastutil.ints.IntArrays;
#endif

#if KEYS_PRIMITIVE

#if ! KEY_CLASS_Boolean
import java.util.concurrent.ExecutorCompletionService;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.LinkedBlockingQueue;
import java.util.concurrent.atomic.AtomicInteger;
#endif

/** A class providing static methods and objects that do useful things with type-specific arrays.
*
* <p>In particular, the {@code forceCapacity()}, {@code ensureCapacity()}, {@code grow()},
* {@code trim()} and {@code setLength()} methods allow to handle
* arrays much like array lists. This can be very useful when efficiency (or
* syntactic simplicity) reasons make array lists unsuitable.
*
* <p>Note that {@link it.unimi.dsi.fastutil.io.BinIO} and {@link it.unimi.dsi.fastutil.io.TextIO}
* contain several methods make it possible to load and save arrays of primitive types as sequences

```
* of elements in {@link java.io.DataInput} format (i.e., not as objects) or as sequences of lines of text.
*
* <h2>Sorting</h2>
*
* <p>There are several sorting methods available. The main theme is that of letting you choose
* the sorting algorithm you prefer (i.e., trading stability of mergesort for no memory allocation in quicksort).
* Several algorithms provide a parallel version, that will use the {@linkplain Runtime#availableProcessors()
number of cores available}.
* Some algorithms also provide an explicit <em>indirect</em> sorting facility, which makes it possible
* to sort an array using the values in another array as comparator.
*
* <p>All comparison-based algorithm have an implementation based on a type-specific comparator.
*
* <p>As a general rule, sequential radix sort is significantly faster than quicksort or mergesort, in particular
* on random-looking data. In
* the parallel case, up to a few cores parallel radix sort is still the fastest, but at some point quicksort
* exploits parallelism better.
*
* <p>If you are fine with not knowing exactly which algorithm will be run (in particular, not knowing exactly
whether a support array will be allocated),
* the dual-pivot parallel sorts in {@link java.util.Arrays}
* are about 50% faster than the classical single-pivot implementation used here.
*
* <p>In any case, if sorting time is important I suggest that you benchmark your sorting load
* with your data distribution and on your architecture.
*
* @see java.util.Arrays
*/

public final class ARRAYS {

#else

import java.util.Comparator;

/** A class providing static methods and objects that do useful things with type-specific arrays.
*
* In particular, the {@code ensureCapacity()}, {@code grow()},
* {@code trim()} and {@code setLength()} methods allow to handle
* arrays much like array lists. This can be very useful when efficiency (or
* syntactic simplicity) reasons make array lists unsuitable.
*
* <p><strong>Warning:</strong> if your array is not of type {@code Object[]},
* {@link #ensureCapacity(Object[],int,int)} and {@link #grow(Object[],int,int)}
* will use {@linkplain java.lang.reflect.Array#newInstance(Class,int) reflection}
* to preserve your array type. Reflection is <em>significantly slower</em> than using {@code new}.
* This phenomenon is particularly
* evident in the first growth phases of an array reallocated with doubling (or similar) logic.
```

```
*
* <h2>Sorting</h2>
*
* <p>There are several sorting methods available. The main theme is that of letting you choose
* the sorting algorithm you prefer (i.e., trading stability of mergesort for no memory allocation in quicksort).
* Several algorithms provide a parallel version, that will use the {@linkplain Runtime#availableProcessors()
number of cores available}.
*
* <p>All comparison-based algorithm have an implementation based on a type-specific comparator.
*
* <p>If you are fine with not knowing exactly which algorithm will be run (in particular, not knowing exactly
whether a support array will be allocated),
* the dual-pivot parallel sorts in {@link java.util.Arrays}
* are about 50% faster than the classical single-pivot implementation used here.
*
* <p>In any case, if sorting time is important I suggest that you benchmark your sorting load
* with your data distribution and on your architecture.
*
* @see java.util.Arrays
*/

public final class ARRAYS {

#endif
 private ARRAYS() {}

 /** A static, final, empty array. */
 public static final KEY_TYPE[] EMPTY_ARRAY = {};

 /** A static, final, empty array to be used as default array in allocations. An
   * object distinct from {@link #EMPTY_ARRAY} makes it possible to have different
   * behaviors depending on whether the user required an empty allocation, or we are
   * just lazily delaying allocation.
   *
   * @see java.util.ArrayList
   */
 public static final KEY_TYPE[] DEFAULT_EMPTY_ARRAY = {};


#if KEY_CLASS_Object
 /** Creates a new array using a the given one as prototype.
  *
  * <p>This method returns a new array of the given length whose element
  * are of the same class as of those of {@code prototype}. In case
  * of an empty array, it tries to return {@link #EMPTY_ARRAY}, if possible.
  *
  * @param prototype an array that will be used to type the new one.
  * @param length the length of the new array.
```

```
 * @return a new array of given type and length.
 */

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 private static <K> K[] newArray(final K[] prototype, final int length) {
  final Class<?> klass = prototype.getClass();
  if (klass == Object[].class) return (K[])(length == 0 ? EMPTY_ARRAY : new Object[length]);
  return (K[])java.lang.reflect.Array.newInstance(klass.getComponentType(), length);
 }
#endif

 /** Forces an array to contain the given number of entries, preserving just a part of the array.
  *
  * @param array an array.
  * @param length the new minimum length for this array.
  * @param preserve the number of elements of the array that must be preserved in case a new allocation is
necessary.
  * @return an array with {@code length} entries whose first {@code preserve}
  * entries are the same as those of {@code array}.
  */
 public static KEY_GENERIC KEY_GENERIC_TYPE[] forceCapacity(final KEY_GENERIC_TYPE[] array, final
int length, final int preserve) {
  final KEY_GENERIC_TYPE t[] =
#if KEY_CLASS_Object
   newArray(array, length);
#else
   new KEY_TYPE[length];
#endif
  System.arraycopy(array, 0, t, 0, preserve);
  return t;
 }

 /** Ensures that an array can contain the given number of entries.
  *
  * <p>If you cannot foresee whether this array will need again to be
  * enlarged, you should probably use {@code grow()} instead.
  *
  * @param array an array.
  * @param length the new minimum length for this array.
  * @return {@code array}, if it contains {@code length} entries or more; otherwise,
  * an array with {@code length} entries whose first {@code array.length}
  * entries are the same as those of {@code array}.
  */
 public static KEY_GENERIC KEY_GENERIC_TYPE[] ensureCapacity(final KEY_GENERIC_TYPE[] array,
final int length) {
  return ensureCapacity(array, length, array.length);
 }
```

```
 /** Ensures that an array can contain the given number of entries, preserving just a part of the array.
  *
  * @param array an array.
  * @param length the new minimum length for this array.
  * @param preserve the number of elements of the array that must be preserved in case a new allocation is
necessary.
  * @return {@code array}, if it can contain {@code length} entries or more; otherwise,
  * an array with {@code length} entries whose first {@code preserve}
  * entries are the same as those of {@code array}.
  */
 public static KEY_GENERIC KEY_GENERIC_TYPE[] ensureCapacity(final KEY_GENERIC_TYPE[] array,
final int length, final int preserve) {
  return length > array.length ? forceCapacity(array, length, preserve) : array;
 }

 /** Grows the given array to the maximum between the given length and
  * the current length increased by 50%, provided that the given
  * length is larger than the current length.
  *
  * <p>If you want complete control on the array growth, you
  * should probably use {@code ensureCapacity()} instead.
  *
  * @param array an array.
  * @param length the new minimum length for this array.
  * @return {@code array}, if it can contain {@code length}
  * entries; otherwise, an array with
  * max({@code length},{@code array.length}/&amp;phi;) entries whose first
  * {@code array.length} entries are the same as those of {@code array}.
  * */

 public static KEY_GENERIC KEY_GENERIC_TYPE[] grow(final KEY_GENERIC_TYPE[] array, final int
length) {
  return grow(array, length, array.length);
 }

 /** Grows the given array to the maximum between the given length and
  * the current length increased by 50%, provided that the given
  * length is larger than the current length, preserving just a part of the array.
  *
  * <p>If you want complete control on the array growth, you
  * should probably use {@code ensureCapacity()} instead.
  *
  * @param array an array.
  * @param length the new minimum length for this array.
  * @param preserve the number of elements of the array that must be preserved in case a new allocation is
necessary.
  * @return {@code array}, if it can contain {@code length}
  * entries; otherwise, an array with
```

```
 * max({@code length},{@code array.length}/&phi;) entries whose first
 * {@code preserve} entries are the same as those of {@code array}.
 * */

 public static KEY_GENERIC KEY_GENERIC_TYPE[] grow(final KEY_GENERIC_TYPE[] array, final int
length, final int preserve) {
  if (length > array.length) {
   final int newLength = (int)Math.max(Math.min((long)array.length + (array.length >> 1),
Arrays.MAX_ARRAY_SIZE), length);

   final KEY_GENERIC_TYPE t[] =
#if KEY_CLASS_Object
    newArray(array, newLength);
#else
    new KEY_TYPE[newLength];
#endif
   System.arraycopy(array, 0, t, 0, preserve);

   return t;
  }
  return array;

 }

 /** Trims the given array to the given length.
  *
  * @param array an array.
  * @param length the new maximum length for the array.
  * @return {@code array}, if it contains {@code length}
  * entries or less; otherwise, an array with
  * {@code length} entries whose entries are the same as
  * the first {@code length} entries of {@code array}.
  *
  */

 public static KEY_GENERIC KEY_GENERIC_TYPE[] trim(final KEY_GENERIC_TYPE[] array, final int length)
{
  if (length >= array.length) return array;
  final KEY_GENERIC_TYPE t[] =
#if KEY_CLASS_Object
   newArray(array, length);
#else
   length == 0 ? EMPTY_ARRAY : new KEY_TYPE[length];
#endif
  System.arraycopy(array, 0, t, 0, length);
  return t;
 }
```

```
 /** Sets the length of the given array.
  *
  * @param array an array.
  * @param length the new length for the array.
  * @return {@code array}, if it contains exactly {@code length}
  * entries; otherwise, if it contains <em>more</em> than
  * {@code length} entries, an array with {@code length} entries
  * whose entries are the same as the first {@code length} entries of
  * {@code array}; otherwise, an array with {@code length} entries
  * whose first {@code array.length} entries are the same as those of
  * {@code array}.
  *
  */

 public static KEY_GENERIC KEY_GENERIC_TYPE[] setLength(final KEY_GENERIC_TYPE[] array, final int
length) {
  if (length == array.length) return array;
  if (length < array.length) return trim(array, length);
  return ensureCapacity(array, length);
 }

 /** Returns a copy of a portion of an array.
  *
  * @param array an array.
  * @param offset the first element to copy.
  * @param length the number of elements to copy.
  * @return a new array containing {@code length} elements of {@code array} starting at {@code offset}.
  */

 public static KEY_GENERIC KEY_GENERIC_TYPE[] copy(final KEY_GENERIC_TYPE[] array, final int offset,
final int length) {
  ensureOffsetLength(array, offset, length);
  final KEY_GENERIC_TYPE[] a =
#if KEY_CLASS_Object
   newArray(array, length);
#else
   length == 0 ? EMPTY_ARRAY : new KEY_TYPE[length];
#endif
  System.arraycopy(array, offset, a, 0, length);
  return a;
 }

 /** Returns a copy of an array.
  *
  * @param array an array.
  * @return a copy of {@code array}.
  */
```

```
public static KEY_GENERIC KEY_GENERIC_TYPE[] copy(final KEY_GENERIC_TYPE[] array) {
 return array.clone();
}

/** Fills the given array with the given value.
 *
 * @param array an array.
 * @param value the new value for all elements of the array.
 * @deprecated Please use the corresponding {@link java.util.Arrays} method.
 */

@Deprecated
public static KEY_GENERIC void fill(final KEY_GENERIC_TYPE[] array, final KEY_GENERIC_TYPE value) {
 int i = array.length;
 while(i-- != 0) array[i] = value;
}

/** Fills a portion of the given array with the given value.
 *
 * @param array an array.
 * @param from the starting index of the portion to fill (inclusive).
 * @param to the end index of the portion to fill (exclusive).
 * @param value the new value for all elements of the specified portion of the array.
 * @deprecated Please use the corresponding {@link java.util.Arrays} method.
 */

@Deprecated
public static KEY_GENERIC void fill(final KEY_GENERIC_TYPE[] array, final int from, int to, final
KEY_GENERIC_TYPE value) {
 ensureFromTo(array, from, to);
 if (from == 0) while(to-- != 0) array[to] = value;
 else for(int i = from; i < to; i++) array[i] = value;
}



/** Returns true if the two arrays are elementwise equal.
 *
 * @param a1 an array.
 * @param a2 another array.
 * @return true if the two arrays are of the same length, and their elements are equal.
 * @deprecated Please use the corresponding {@link java.util.Arrays} method, which is intrinsified in recent JVMs.
 */

@Deprecated
public static KEY_GENERIC boolean equals(final KEY_GENERIC_TYPE[] a1, final KEY_GENERIC_TYPE
a2[]) {
 int i = a1.length;
```

```
 if (i != a2.length) return false;
 while(i-- != 0) if (! KEY_EQUALS(a1[i], a2[i])) return false;
 return true;
 }



 /** Ensures that a range given by its first (inclusive) and last (exclusive) elements fits an array.
  *
  * <p>This method may be used whenever an array range check is needed.
  *
  * @param a an array.
  * @param from a start index (inclusive).
  * @param to an end index (exclusive).
  * @throws IllegalArgumentException if {@code from} is greater than {@code to}.
  * @throws ArrayIndexOutOfBoundsException if {@code from} or {@code to} are greater than the array length or
negative.
  */
 public static KEY_GENERIC void ensureFromTo(final KEY_GENERIC_TYPE[] a, final int from, final int to) {
  Arrays.ensureFromTo(a.length, from, to);
 }

 /** Ensures that a range given by an offset and a length fits an array.
  *
  * <p>This method may be used whenever an array range check is needed.
  *
  * @param a an array.
  * @param offset a start index.
  * @param length a length (the number of elements in the range).
  * @throws IllegalArgumentException if {@code length} is negative.
  * @throws ArrayIndexOutOfBoundsException if {@code offset} is negative or {@code offset}+{@code length} is
greater than the array length.
  */
 public static KEY_GENERIC void ensureOffsetLength(final KEY_GENERIC_TYPE[] a, final int offset, final int
length) {
  Arrays.ensureOffsetLength(a.length, offset, length);
 }

 /** Ensures that two arrays are of the same length.
  *
  * @param a an array.
  * @param b another array.
  * @throws IllegalArgumentException if the two argument arrays are not of the same length.
  */
 public static KEY_GENERIC void ensureSameLength(final KEY_GENERIC_TYPE[] a, final
KEY_GENERIC_TYPE[] b) {
  if (a.length != b.length) throw new IllegalArgumentException("Array size mismatch: " + a.length + " != " +
```

```
b.length);
 }

 private static final int QUICKSORT_NO_REC = 16;
 private static final int PARALLEL_QUICKSORT_NO_FORK = 8192;
 private static final int QUICKSORT_MEDIAN_OF_9 = 128;
 private static final int MERGESORT_NO_REC = 16;

 /** Swaps two elements of an anrray.
  *
  * @param x an array.
  * @param a a position in {@code x}.
  * @param b another position in {@code x}.
  */
 public static KEY_GENERIC void swap(final KEY_GENERIC_TYPE x[], final int a, final int b) {
  final KEY_GENERIC_TYPE t = x[a];
  x[a] = x[b];
  x[b] = t;
 }

 /** Swaps two sequences of elements of an array.
  *
  * @param x an array.
  * @param a a position in {@code x}.
  * @param b another position in {@code x}.
  * @param n the number of elements to exchange starting at {@code a} and {@code b}.
  */
 public static KEY_GENERIC void swap(final KEY_GENERIC_TYPE[] x, int a, int b, final int n) {
  for(int i = 0; i < n; i++, a++, b++) swap(x, a, b);
 }

 private static KEY_GENERIC int med3(final KEY_GENERIC_TYPE x[], final int a, final int b, final int c,
KEY_COMPARATOR KEY_GENERIC comp) {
  final int ab = comp.compare(x[a], x[b]);
  final int ac = comp.compare(x[a], x[c]);
  final int bc = comp.compare(x[b], x[c]);
  return (ab < 0 ?
   (bc < 0 ? b : ac < 0 ? c : a) :
   (bc > 0 ? b : ac > 0 ? c : a));
 }

 private static KEY_GENERIC void selectionSort(final KEY_GENERIC_TYPE[] a, final int from, final int to, final
KEY_COMPARATOR KEY_GENERIC comp) {
  for(int i = from; i < to - 1; i++) {
   int m = i;
   for(int j = i + 1; j < to; j++) if (comp.compare(a[j], a[m]) < 0) m = j;
   if (m != i) {
    final KEY_GENERIC_TYPE u = a[i];
```

```
   a[i] = a[m];
   a[m] = u;
  }
 }
}

 private static KEY_GENERIC void insertionSort(final KEY_GENERIC_TYPE[] a, final int from, final int to, final
KEY_COMPARATOR KEY_GENERIC comp) {
  for (int i = from; ++i < to;) {
   KEY_GENERIC_TYPE t = a[i];
   int j = i;
   for (KEY_GENERIC_TYPE u = a[j - 1]; comp.compare(t, u) < 0; u = a[--j - 1]) {
    a[j] = u;
    if (from == j - 1) {
     --j;
     break;
    }
   }
   a[j] = t;
  }
 }
}



 /** Sorts the specified range of elements according to the order induced by the specified
  * comparator using quicksort.
  *
  * <p>The sorting algorithm is a tuned quicksort adapted from Jon L. Bentley and M. Douglas
  * McIlroy, &amp;ldquo;Engineering a Sort Function&amp;rdquo;, <i>Software: Practice and Experience</i>, 23(11), pages
  * 1249&amp;minus;1265, 1993.
  *
  * <p>Note that this implementation does not allocate any object, contrarily to the implementation
  * used to sort primitive types in {@link java.util.Arrays}, which switches to mergesort on large inputs.
  *
  * @param x the array to be sorted.
  * @param from the index of the first element (inclusive) to be sorted.
  * @param to the index of the last element (exclusive) to be sorted.
  * @param comp the comparator to determine the sorting order.
  *
  */
 public static KEY_GENERIC void quickSort(final KEY_GENERIC_TYPE[] x, final int from, final int to, final
KEY_COMPARATOR KEY_GENERIC comp) {
  final int len = to - from;
  // Selection sort on smallest arrays
  if (len < QUICKSORT_NO_REC) {
   selectionSort(x, from, to, comp);
   return;
  }
```

```
// Choose a partition element, v
int m = from + len / 2;
int l = from;
int n = to - 1;
if (len > QUICKSORT_MEDIAN_OF_9) {  // Big arrays, pseudomedian of 9
 int s = len / 8;
 l = med3(x, l, l + s, l + 2 * s, comp);
 m = med3(x, m - s, m, m + s, comp);
 n = med3(x, n - 2 * s, n - s, n, comp);
}
m = med3(x, l, m, n, comp); // Mid-size, med of 3

final KEY_GENERIC_TYPE v = x[m];

// Establish Invariant: v* (<v)* (>v)* v*
int a = from, b = a, c = to - 1, d = c;
while(true) {
 int comparison;
 while (b <= c && (comparison = comp.compare(x[b], v)) <= 0) {
  if (comparison == 0) swap(x, a++, b);
  b++;
 }
 while (c >= b && (comparison = comp.compare(x[c], v)) >=0) {
  if (comparison == 0) swap(x, c, d--);
  c--;
 }
 if (b > c) break;
 swap(x, b++, c--);
}

// Swap partition elements back to middle
int s;
s = Math.min(a - from, b - a);
swap(x, from, b - s, s);
s = Math.min(d - c, to - d - 1);
swap(x, b, to - s, s);

// Recursively sort non-partition-elements
if ((s = b - a) > 1) quickSort(x, from, from + s, comp);
if ((s = d - c) > 1) quickSort(x, to - s, to, comp);
}

/** Sorts an array according to the order induced by the specified
 * comparator using quicksort.
 *
 * <p>The sorting algorithm is a tuned quicksort adapted from Jon L. Bentley and M. Douglas
 * McIlroy, &amp;ldquo;Engineering a Sort Function&amp;rdquo;, <i>Software: Practice and Experience</i>, 23(11), pages
```

```
 * 1249&amp;minus;1265, 1993.
 *
 * <p>Note that this implementation does not allocate any object, contrarily to the implementation
 * used to sort primitive types in {@link java.util.Arrays}, which switches to mergesort on large inputs.
 *
 * @param x the array to be sorted.
 * @param comp the comparator to determine the sorting order.
 *
 */
 public static KEY_GENERIC void quickSort(final KEY_GENERIC_TYPE[] x, final KEY_COMPARATOR
KEY_GENERIC comp) {
  quickSort(x, 0, x.length, comp);
 }


 protected static class ForkJoinQuickSortComp KEY_GENERIC extends RecursiveAction {
  private static final long serialVersionUID = 1L;
  private final int from;
  private final int to;
  private final KEY_GENERIC_TYPE[] x;
  private final KEY_COMPARATOR KEY_GENERIC comp;

  public ForkJoinQuickSortComp(final KEY_GENERIC_TYPE[] x , final int from , final int to, final
KEY_COMPARATOR KEY_GENERIC comp) {
   this.from = from;
   this.to = to;
   this.x = x;
   this.comp = comp;
  }

  @Override
  protected void compute() {
   final KEY_GENERIC_TYPE[] x = this.x;
   final int len = to - from;
   if (len < PARALLEL_QUICKSORT_NO_FORK) {
    quickSort(x, from, to, comp);
    return;
   }
   // Choose a partition element, v
   int m = from + len / 2;
   int l = from;
   int n = to - 1;
   int s = len / 8;
   l = med3(x, l, l + s, l + 2 * s, comp);
   m = med3(x, m - s, m, m + s, comp);
   n = med3(x, n - 2 * s, n - s, n, comp);
   m = med3(x, l, m, n, comp);
   final KEY_GENERIC_TYPE v = x[m];
```

```
  // Establish Invariant: v* (<v)* (>v)* v*
  int a = from, b = a, c = to - 1, d = c;
  while (true) {
   int comparison;
   while (b <= c && (comparison = comp.compare(x[b], v)) <= 0) {
    if (comparison == 0) swap(x, a++, b);
    b++;
   }
   while (c >= b && (comparison = comp.compare(x[c], v)) >= 0) {
    if (comparison == 0) swap(x, c, d--);
    c--;
   }
   if (b > c) break;
   swap(x, b++, c--);
  }
  // Swap partition elements back to middle
  int t;
  s = Math.min(a - from, b - a);
  swap(x, from, b - s, s);
  s = Math.min(d - c, to - d - 1);
  swap(x, b, to - s, s);
  // Recursively sort non-partition-elements
  s = b - a;
  t = d - c;
  if (s > 1 && t > 1) invokeAll(new ForkJoinQuickSortComp KEY_GENERIC_DIAMOND(x, from, from + s,
comp), new ForkJoinQuickSortComp KEY_GENERIC_DIAMOND(x, to - t, to, comp));
  else if (s > 1) invokeAll(new ForkJoinQuickSortComp KEY_GENERIC_DIAMOND(x, from, from + s, comp));
  else invokeAll(new ForkJoinQuickSortComp KEY_GENERIC_DIAMOND(x, to - t, to, comp));
 }
}

/** Sorts the specified range of elements according to the order induced by the specified
 * comparator using a parallel quicksort.
 *
 * <p>The sorting algorithm is a tuned quicksort adapted from Jon L. Bentley and M. Douglas
 * McIlroy, &ldquo;Engineering a Sort Function&rdquo;, <i>Software: Practice and Experience</i>, 23(11), pages
 * 1249&minus;1265, 1993.
 *
 * <p>This implementation uses a {@link ForkJoinPool} executor service with
 * {@link Runtime#availableProcessors()} parallel threads.
 *
 * @param x the array to be sorted.
 * @param from the index of the first element (inclusive) to be sorted.
 * @param to the index of the last element (exclusive) to be sorted.
 * @param comp the comparator to determine the sorting order.
 */
public static KEY_GENERIC void parallelQuickSort(final KEY_GENERIC_TYPE[] x, final int from, final int to,
final KEY_COMPARATOR KEY_GENERIC comp) {
```

```
 if (to - from < PARALLEL_QUICKSORT_NO_FORK) quickSort(x, from, to, comp);
 else {
  final ForkJoinPool pool = new ForkJoinPool(Runtime.getRuntime().availableProcessors());
  pool.invoke(new ForkJoinQuickSortComp KEY_GENERIC_DIAMOND(x, from, to, comp));
  pool.shutdown();
 }
}

/** Sorts an array according to the order induced by the specified
 * comparator using a parallel quicksort.
 *
 * <p>The sorting algorithm is a tuned quicksort adapted from Jon L. Bentley and M. Douglas
 * McIlroy, &ldquo;Engineering a Sort Function&rdquo;, <i>Software: Practice and Experience</i>, 23(11), pages
 * 1249&minus;1265, 1993.
 *
 * <p>This implementation uses a {@link ForkJoinPool} executor service with
 * {@link Runtime#availableProcessors()} parallel threads.
 *
 * @param x the array to be sorted.
 * @param comp the comparator to determine the sorting order.
 */
public static KEY_GENERIC void parallelQuickSort(final KEY_GENERIC_TYPE[] x, final
KEY_COMPARATOR KEY_GENERIC comp) {
 parallelQuickSort(x, 0, x.length, comp);
}


SUPPRESS_WARNINGS_KEY_UNCHECKED
private static KEY_GENERIC int med3(final KEY_GENERIC_TYPE x[], final int a, final int b, final int c) {
 final int ab = KEY_CMP(x[a], x[b]);
 final int ac = KEY_CMP(x[a], x[c]);
 final int bc = KEY_CMP(x[b], x[c]);
 return (ab < 0 ?
  (bc < 0 ? b : ac < 0 ? c : a) :
  (bc > 0 ? b : ac > 0 ? c : a));
}

SUPPRESS_WARNINGS_KEY_UNCHECKED
private static KEY_GENERIC void selectionSort(final KEY_GENERIC_TYPE[] a, final int from, final int to) {
 for(int i = from; i < to - 1; i++) {
  int m = i;
  for(int j = i + 1; j < to; j++) if (KEY_LESS(a[j], a[m])) m = j;
  if (m != i) {
   final KEY_GENERIC_TYPE u = a[i];
   a[i] = a[m];
   a[m] = u;
  }
 }
```

```
}

SUPPRESS_WARNINGS_KEY_UNCHECKED
private static KEY_GENERIC void insertionSort(final KEY_GENERIC_TYPE[] a, final int from, final int to) {
 for (int i = from; ++i < to;) {
  KEY_GENERIC_TYPE t = a[i];
  int j = i;
  for (KEY_GENERIC_TYPE u = a[j - 1]; KEY_LESS(t, u); u = a[--j - 1]) {
   a[j] = u;
   if (from == j - 1) {
    --j;
    break;
   }
  }
  a[j] = t;
 }
}

/** Sorts the specified range of elements according to the natural ascending order using quicksort.
 *
 * <p>The sorting algorithm is a tuned quicksort adapted from Jon L. Bentley and M. Douglas
 * McIlroy, &ldquo;Engineering a Sort Function&rdquo;, <i>Software: Practice and Experience</i>, 23(11), pages
 * 1249&minus;1265, 1993.
 *
 * <p>Note that this implementation does not allocate any object, contrarily to the implementation
 * used to sort primitive types in {@link java.util.Arrays}, which switches to mergesort on large inputs.
 *
 * @param x the array to be sorted.
 * @param from the index of the first element (inclusive) to be sorted.
 * @param to the index of the last element (exclusive) to be sorted.
 */

SUPPRESS_WARNINGS_KEY_UNCHECKED
public static KEY_GENERIC void quickSort(final KEY_GENERIC_TYPE[] x, final int from, final int to) {
 final int len = to - from;
 // Selection sort on smallest arrays
 if (len < QUICKSORT_NO_REC) {
  selectionSort(x, from, to);
  return;
 }

 // Choose a partition element, v
 int m = from + len / 2;
 int l = from;
 int n = to - 1;
 if (len > QUICKSORT_MEDIAN_OF_9) {  // Big arrays, pseudomedian of 9
  int s = len / 8;
  l = med3(x, l, l + s, l + 2 * s);
```

```
 m = med3(x, m - s, m, m + s);
 n = med3(x, n - 2 * s, n - s, n);
 }
 m = med3(x, l, m, n); // Mid-size, med of 3

 final KEY_GENERIC_TYPE v = x[m];

 // Establish Invariant: v* (<v)* (>v)* v*
 int a = from, b = a, c = to - 1, d = c;
 while(true) {
  int comparison;
  while (b <= c && (comparison = KEY_CMP(x[b], v)) <= 0) {
   if (comparison == 0) swap(x, a++, b);
   b++;
  }
  while (c >= b && (comparison = KEY_CMP(x[c], v)) >=0) {
   if (comparison == 0) swap(x, c, d--);
   c--;
  }
  if (b > c) break;
  swap(x, b++, c--);
 }

 // Swap partition elements back to middle
 int s;
 s = Math.min(a - from, b - a);
 swap(x, from, b - s, s);
 s = Math.min(d - c, to - d - 1);
 swap(x, b, to - s, s);

 // Recursively sort non-partition-elements
 if ((s = b - a) > 1) quickSort(x, from, from + s);
 if ((s = d - c) > 1) quickSort(x, to - s, to);
}

/** Sorts an array according to the natural ascending order using quicksort.
 *
 * <p>The sorting algorithm is a tuned quicksort adapted from Jon L. Bentley and M. Douglas
 * McIlroy, &ldquo;Engineering a Sort Function&rdquo;, <i>Software: Practice and Experience</i>, 23(11), pages
 * 1249&minus;1265, 1993.
 *
 * <p>Note that this implementation does not allocate any object, contrarily to the implementation
 * used to sort primitive types in {@link java.util.Arrays}, which switches to mergesort on large inputs.
 *
 * @param x the array to be sorted.
 *
 */
public static KEY_GENERIC void quickSort(final KEY_GENERIC_TYPE[] x) {
```

```
 quickSort(x, 0, x.length);
}

protected static class ForkJoinQuickSort KEY_GENERIC extends RecursiveAction {
 private static final long serialVersionUID = 1L;
 private final int from;
 private final int to;
 private final KEY_GENERIC_TYPE[] x;

 public ForkJoinQuickSort(final KEY_GENERIC_TYPE[] x , final int from , final int to) {
  this.from = from;
  this.to = to;
  this.x = x;
 }

 @Override
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 protected void compute() {
  final KEY_GENERIC_TYPE[] x = this.x;
  final int len = to - from;
  if (len < PARALLEL_QUICKSORT_NO_FORK) {
   quickSort(x, from, to);
   return;
  }
  // Choose a partition element, v
  int m = from + len / 2;
  int l = from;
  int n = to - 1;
  int s = len / 8;
  l = med3(x, l, l + s, l + 2 * s);
  m = med3(x, m - s, m, m + s);
  n = med3(x, n - 2 * s, n - s, n);
  m = med3(x, l, m, n);
  final KEY_GENERIC_TYPE v = x[m];
  // Establish Invariant: v* (<v)* (>v)* v*
  int a = from, b = a, c = to - 1, d = c;
  while (true) {
   int comparison;
   while (b <= c && (comparison = KEY_CMP(x[b], v)) <= 0) {
    if (comparison == 0) swap(x, a++, b);
    b++;
   }
   while (c >= b && (comparison = KEY_CMP(x[c], v)) >= 0) {
    if (comparison == 0) swap(x, c, d--);
    c--;
   }
   if (b > c) break;
   swap(x, b++, c--);
```

```
  }
  // Swap partition elements back to middle
  int t;
  s = Math.min(a - from, b - a);
  swap(x, from, b - s, s);
  s = Math.min(d - c, to - d - 1);
  swap(x, b, to - s, s);
  // Recursively sort non-partition-elements
  s = b - a;
  t = d - c;
  if (s > 1 && t > 1) invokeAll(new ForkJoinQuickSort KEY_GENERIC_DIAMOND(x, from, from + s), new
ForkJoinQuickSort KEY_GENERIC_DIAMOND(x, to - t, to));
  else if (s > 1) invokeAll(new ForkJoinQuickSort KEY_GENERIC_DIAMOND(x, from, from + s));
  else invokeAll(new ForkJoinQuickSort KEY_GENERIC_DIAMOND(x, to - t, to));
 }
}

/**  Sorts the specified range of elements according to the natural ascending order using a parallel quicksort.
 *
 * <p>The sorting algorithm is a tuned quicksort adapted from Jon L. Bentley and M. Douglas
 * McIlroy, &ldquo;Engineering a Sort Function&rdquo;, <i>Software: Practice and Experience</i>, 23(11), pages
 * 1249&minus;1265, 1993.
 *
 * <p>This implementation uses a {@link ForkJoinPool} executor service with
 * {@link Runtime#availableProcessors()} parallel threads.
 *
 * @param x the array to be sorted.
 * @param from the index of the first element (inclusive) to be sorted.
 * @param to the index of the last element (exclusive) to be sorted.
 */
public static KEY_GENERIC void parallelQuickSort(final KEY_GENERIC_TYPE[] x, final int from, final int to) {
 if (to - from < PARALLEL_QUICKSORT_NO_FORK) quickSort(x, from, to);
 else {
  final ForkJoinPool pool = new ForkJoinPool(Runtime.getRuntime().availableProcessors());
  pool.invoke(new ForkJoinQuickSort KEY_GENERIC_DIAMOND(x, from, to));
  pool.shutdown();
 }
}

/** Sorts an array according to the natural ascending order using a parallel quicksort.
 *
 * <p>The sorting algorithm is a tuned quicksort adapted from Jon L. Bentley and M. Douglas
 * McIlroy, &ldquo;Engineering a Sort Function&rdquo;, <i>Software: Practice and Experience</i>, 23(11), pages
 * 1249&minus;1265, 1993.
 *
 * <p>This implementation uses a {@link ForkJoinPool} executor service with
 * {@link Runtime#availableProcessors()} parallel threads.
 *
```

```
 * @param x the array to be sorted.
 *
 */
public static KEY_GENERIC void parallelQuickSort(final KEY_GENERIC_TYPE[] x) {
 parallelQuickSort(x, 0, x.length);
}


SUPPRESS_WARNINGS_KEY_UNCHECKED
private static KEY_GENERIC int med3Indirect(final int perm[], final KEY_GENERIC_TYPE x[], final int a, final
int b, final int c) {
 final KEY_GENERIC_TYPE aa = x[perm[a]];
 final KEY_GENERIC_TYPE bb = x[perm[b]];
 final KEY_GENERIC_TYPE cc = x[perm[c]];
 final int ab = KEY_CMP(aa, bb);
 final int ac = KEY_CMP(aa, cc);
 final int bc = KEY_CMP(bb, cc);
 return (ab < 0 ?
  (bc < 0 ? b : ac < 0 ? c : a) :
  (bc > 0 ? b : ac > 0 ? c : a));
}

SUPPRESS_WARNINGS_KEY_UNCHECKED
private static KEY_GENERIC void insertionSortIndirect(final int[] perm, final KEY_GENERIC_TYPE[] a, final int
from, final int to) {
 for (int i = from; ++i < to;) {
  int t = perm[i];
  int j = i;
  for (int u = perm[j - 1]; KEY_LESS(a[t], a[u]); u = perm[--j - 1]) {
   perm[j] = u;
   if (from == j - 1) {
    --j;
    break;
   }
  }
  perm[j] = t;
 }
}

/** Sorts the specified range of elements according to the natural ascending order using indirect quicksort.
 *
 * <p>The sorting algorithm is a tuned quicksort adapted from Jon L. Bentley and M. Douglas
 * McIlroy, &ldquo;Engineering a Sort Function&rdquo;, <i>Software: Practice and Experience</i>, 23(11), pages
 * 1249&minus;1265, 1993.
 *
 * <p>This method implement an <em>indirect</em> sort. The elements of {@code perm} (which must
 * be exactly the numbers in the interval {@code [0..perm.length)}) will be permuted so that
 * {@code x[perm[i]] &le; x[perm[i + 1]]}.
```

```
 *
 * <p>Note that this implementation does not allocate any object, contrarily to the implementation
 * used to sort primitive types in {@link java.util.Arrays}, which switches to mergesort on large inputs.
 *
 * @param perm a permutation array indexing {@code x}.
 * @param x the array to be sorted.
 * @param from the index of the first element (inclusive) to be sorted.
 * @param to the index of the last element (exclusive) to be sorted.
 */

SUPPRESS_WARNINGS_KEY_UNCHECKED
public static KEY_GENERIC void quickSortIndirect(final int[] perm, final KEY_GENERIC_TYPE[] x, final int
from, final int to) {
 final int len = to - from;
 // Selection sort on smallest arrays
 if (len < QUICKSORT_NO_REC) {
  insertionSortIndirect(perm, x, from, to);
  return;
 }

 // Choose a partition element, v
 int m = from + len / 2;
 int l = from;
 int n = to - 1;
 if (len > QUICKSORT_MEDIAN_OF_9) {  // Big arrays, pseudomedian of 9
  int s = len / 8;
  l = med3Indirect(perm, x, l, l + s, l + 2 * s);
  m = med3Indirect(perm, x, m - s, m, m + s);
  n = med3Indirect(perm, x, n - 2 * s, n - s, n);
 }
 m = med3Indirect(perm, x, l, m, n); // Mid-size, med of 3

 final KEY_GENERIC_TYPE v = x[perm[m]];

 // Establish Invariant: v* (<v)* (>v)* v*
 int a = from, b = a, c = to - 1, d = c;
 while(true) {
  int comparison;
  while (b <= c && (comparison = KEY_CMP(x[perm[b]], v)) <= 0) {
   if (comparison == 0) IntArrays.swap(perm, a++, b);
   b++;
  }
  while (c >= b && (comparison = KEY_CMP(x[perm[c]], v)) >=0) {
   if (comparison == 0) IntArrays.swap(perm, c, d--);
   c--;
  }
  if (b > c) break;
  IntArrays.swap(perm, b++, c--);
```

```
 }

 // Swap partition elements back to middle
 int s;
 s = Math.min(a - from, b - a);
 IntArrays.swap(perm, from, b - s, s);
 s = Math.min(d - c, to - d - 1);
 IntArrays.swap(perm, b, to - s, s);

 // Recursively sort non-partition-elements
 if ((s = b - a) > 1) quickSortIndirect(perm, x, from, from + s);
 if ((s = d - c) > 1) quickSortIndirect(perm, x, to - s, to);
}

/** Sorts an array according to the natural ascending order using indirect quicksort.
 *
 * <p>The sorting algorithm is a tuned quicksort adapted from Jon L. Bentley and M. Douglas
 * McIlroy, &ldquo;Engineering a Sort Function&rdquo;, <i>Software: Practice and Experience</i>, 23(11), pages
 * 1249&minus;1265, 1993.
 *
 * <p>This method implement an <em>indirect</em> sort. The elements of {@code perm} (which must
 * be exactly the numbers in the interval {@code [0..perm.length)}) will be permuted so that
 * {@code x[perm[i]] &le; x[perm[i + 1]]}.
 *
 * <p>Note that this implementation does not allocate any object, contrarily to the implementation
 * used to sort primitive types in {@link java.util.Arrays}, which switches to mergesort on large inputs.
 *
 * @param perm a permutation array indexing {@code x}.
 * @param x the array to be sorted.
 */
public static KEY_GENERIC void quickSortIndirect(final int perm[], final KEY_GENERIC_TYPE[] x) {
 quickSortIndirect(perm, x, 0, x.length);
}

protected static class ForkJoinQuickSortIndirect KEY_GENERIC extends RecursiveAction {
 private static final long serialVersionUID = 1L;
 private final int from;
 private final int to;
 private final int[] perm;
 private final KEY_GENERIC_TYPE[] x;

 public ForkJoinQuickSortIndirect(final int perm[], final KEY_GENERIC_TYPE[] x , final int from , final int to) {
  this.from = from;
  this.to = to;
  this.x = x;
  this.perm = perm;
 }
```

```
@Override
SUPPRESS_WARNINGS_KEY_UNCHECKED
protected void compute() {
 final KEY_GENERIC_TYPE[] x = this.x;
 final int len = to - from;
 if (len < PARALLEL_QUICKSORT_NO_FORK) {
  quickSortIndirect(perm, x, from, to);
  return;
 }
 // Choose a partition element, v
 int m = from + len / 2;
 int l = from;
 int n = to - 1;
 int s = len / 8;
 l = med3Indirect(perm, x, l, l + s, l + 2 * s);
 m = med3Indirect(perm, x, m - s, m, m + s);
 n = med3Indirect(perm, x, n - 2 * s, n - s, n);
 m = med3Indirect(perm, x, l, m, n);
 final KEY_GENERIC_TYPE v = x[perm[m]];
 // Establish Invariant: v* (<v)* (>v)* v*
 int a = from, b = a, c = to - 1, d = c;
 while (true) {
  int comparison;
  while (b <= c && (comparison = KEY_CMP(x[perm[b]], v)) <= 0) {
   if (comparison == 0) IntArrays.swap(perm, a++, b);
   b++;
  }
  while (c >= b && (comparison = KEY_CMP(x[perm[c]], v)) >= 0) {
   if (comparison == 0) IntArrays.swap(perm, c, d--);
   c--;
  }
  if (b > c) break;
  IntArrays.swap(perm, b++, c--);
 }
 // Swap partition elements back to middle
 int t;
 s = Math.min(a - from, b - a);
 IntArrays.swap(perm, from, b - s, s);
 s = Math.min(d - c, to - d - 1);
 IntArrays.swap(perm, b, to - s, s);
 // Recursively sort non-partition-elements
 s = b - a;
 t = d - c;
 if (s > 1 && t > 1) invokeAll(new ForkJoinQuickSortIndirect KEY_GENERIC_DIAMOND(perm, x, from, from +
s), new ForkJoinQuickSortIndirect KEY_GENERIC_DIAMOND(perm, x, to - t, to));
 else if (s > 1) invokeAll(new ForkJoinQuickSortIndirect KEY_GENERIC_DIAMOND(perm, x, from, from + s));
 else invokeAll(new ForkJoinQuickSortIndirect KEY_GENERIC_DIAMOND(perm, x, to - t, to));
}
```

```
}

/**  Sorts the specified range of elements according to the natural ascending order using a parallel indirect quicksort.
 *
 * <p>The sorting algorithm is a tuned quicksort adapted from Jon L. Bentley and M. Douglas
 * McIlroy, &ldquo;Engineering a Sort Function&rdquo;, <i>Software: Practice and Experience</i>, 23(11), pages
 * 1249&minus;1265, 1993.
 *
 * <p>This method implement an <em>indirect</em> sort. The elements of {@code perm} (which must
 * be exactly the numbers in the interval {@code [0..perm.length)}) will be permuted so that
 * {@code x[perm[i]] &le; x[perm[i + 1]]}.
 *
 * <p>This implementation uses a {@link ForkJoinPool} executor service with
 * {@link Runtime#availableProcessors()} parallel threads.
 *
 * @param perm a permutation array indexing {@code x}.
 * @param x the array to be sorted.
 * @param from the index of the first element (inclusive) to be sorted.
 * @param to the index of the last element (exclusive) to be sorted.
 */
public static KEY_GENERIC void parallelQuickSortIndirect(final int[] perm, final KEY_GENERIC_TYPE[] x,
final int from, final int to) {
 if (to - from < PARALLEL_QUICKSORT_NO_FORK) quickSortIndirect(perm, x, from, to);
 else {
  final ForkJoinPool pool = new ForkJoinPool(Runtime.getRuntime().availableProcessors());
  pool.invoke(new ForkJoinQuickSortIndirect KEY_GENERIC_DIAMOND(perm, x, from, to));
  pool.shutdown();
 }
}

/** Sorts an array according to the natural ascending order using a parallel indirect quicksort.
 *
 * <p>The sorting algorithm is a tuned quicksort adapted from Jon L. Bentley and M. Douglas
 * McIlroy, &ldquo;Engineering a Sort Function&rdquo;, <i>Software: Practice and Experience</i>, 23(11), pages
 * 1249&minus;1265, 1993.
 *
 * <p>This method implement an <em>indirect</em> sort. The elements of {@code perm} (which must
 * be exactly the numbers in the interval {@code [0..perm.length)}) will be permuted so that
 * {@code x[perm[i]] &le; x[perm[i + 1]]}.
 *
 * <p>This implementation uses a {@link ForkJoinPool} executor service with
 * {@link Runtime#availableProcessors()} parallel threads.
 *
 * @param perm a permutation array indexing {@code x}.
 * @param x the array to be sorted.
 *
 */
public static KEY_GENERIC void parallelQuickSortIndirect(final int perm[], final KEY_GENERIC_TYPE[] x) {
```

```
 parallelQuickSortIndirect(perm, x, 0, x.length);
 }

 /** Stabilizes a permutation.
  *
  * <p>This method can be used to stabilize the permutation generated by an indirect sorting, assuming that
  * initially the permutation array was in ascending order (e.g., the identity, as usually happens). This method
  * scans the permutation, and for each non-singleton block of elements with the same associated values in {@code
x},
  * permutes them in ascending order. The resulting permutation corresponds to a stable sort.
  *
  * <p>Usually combining an unstable indirect sort and this method is more efficient than using a stable sort,
  * as most stable sort algorithms require a support array.
  *
  * <p>More precisely, assuming that {@code x[perm[i]] &le; x[perm[i + 1]]}, after
  * stabilization we will also have that {@code x[perm[i]] = x[perm[i + 1]]} implies
  * {@code perm[i] &le; perm[i + 1]}.
  *
  * @param perm a permutation array indexing {@code x} so that it is sorted.
  * @param x the sorted array to be stabilized.
  * @param from the index of the first element (inclusive) to be stabilized.
  * @param to the index of the last element (exclusive) to be stabilized.
  */
 public static KEY_GENERIC void stabilize(final int perm[], final KEY_GENERIC_TYPE[] x, final int from, final
int to) {
  int curr = from;
  for(int i = from + 1; i < to; i++) {
   if (x[perm[i]] != x[perm[curr]]) {
    if (i - curr > 1) IntArrays.parallelQuickSort(perm, curr, i);
    curr = i;
   }
  }
  if (to - curr > 1) IntArrays.parallelQuickSort(perm, curr, to);
 }

 /** Stabilizes a permutation.
  *
  * <p>This method can be used to stabilize the permutation generated by an indirect sorting, assuming that
  * initially the permutation array was in ascending order (e.g., the identity, as usually happens). This method
  * scans the permutation, and for each non-singleton block of elements with the same associated values in {@code
x},
  * permutes them in ascending order. The resulting permutation corresponds to a stable sort.
  *
  * <p>Usually combining an unstable indirect sort and this method is more efficient than using a stable sort,
  * as most stable sort algorithms require a support array.
  *
  * <p>More precisely, assuming that {@code x[perm[i]] &le; x[perm[i + 1]]}, after
  * stabilization we will also have that {@code x[perm[i]] = x[perm[i + 1]]} implies
```

```
 * {@code perm[i] &le; perm[i + 1]}.
 *
 * @param perm a permutation array indexing {@code x} so that it is sorted.
 * @param x the sorted array to be stabilized.
 */
public static KEY_GENERIC void stabilize(final int perm[], final KEY_GENERIC_TYPE[] x) {
 stabilize(perm, x, 0, perm.length);
}

SUPPRESS_WARNINGS_KEY_UNCHECKED
private static KEY_GENERIC int med3(final KEY_GENERIC_TYPE x[], final KEY_GENERIC_TYPE[] y, final int a, final int b, final int c) {
 int t;
 final int ab = (t = KEY_CMP(x[a], x[b])) == 0 ? KEY_CMP(y[a], y[b]) : t;
 final int ac = (t = KEY_CMP(x[a], x[c])) == 0 ? KEY_CMP(y[a], y[c]) : t;
 final int bc = (t = KEY_CMP(x[b], x[c])) == 0 ? KEY_CMP(y[b], y[c]) : t;
 return (ab < 0 ?
   (bc < 0 ? b : ac < 0 ? c : a) :
   (bc > 0 ? b : ac > 0 ? c : a));
}

private static KEY_GENERIC void swap(final KEY_GENERIC_TYPE x[], final KEY_GENERIC_TYPE[] y, final int a, final int b) {
 final KEY_GENERIC_TYPE t = x[a];
 final KEY_GENERIC_TYPE u = y[a];
 x[a] = x[b];
 y[a] = y[b];
 x[b] = t;
 y[b] = u;
}

private static KEY_GENERIC void swap(final KEY_GENERIC_TYPE[] x, final KEY_GENERIC_TYPE[] y, int a, int b, final int n) {
 for (int i = 0; i < n; i++, a++, b++) swap(x, y, a, b);
}

SUPPRESS_WARNINGS_KEY_UNCHECKED
private static KEY_GENERIC void selectionSort(final KEY_GENERIC_TYPE[] a, final KEY_GENERIC_TYPE[] b, final int from, final int to) {
 for(int i = from; i < to - 1; i++) {
  int m = i, u;
  for(int j = i + 1; j < to; j++)
   if ((u = KEY_CMP(a[j], a[m])) < 0 || u == 0 && KEY_LESS(b[j], b[m])) m = j;

  if (m != i) {
   KEY_GENERIC_TYPE t = a[i];
   a[i] = a[m];
   a[m] = t;
```

```
  t = b[i];
  b[i] = b[m];
  b[m] = t;
 }
 }
}

/** Sorts the specified range of elements of two arrays according to the natural lexicographical
 *  ascending order using quicksort.
 *
 * <p>The sorting algorithm is a tuned quicksort adapted from Jon L. Bentley and M. Douglas
 * McIlroy, &ldquo;Engineering a Sort Function&rdquo;, <i>Software: Practice and Experience</i>, 23(11), pages
 * 1249&minus;1265, 1993.
 *
 * <p>This method implements a <em>lexicographical</em> sorting of the arguments. Pairs of
 * elements in the same position in the two provided arrays will be considered a single key, and
 * permuted accordingly. In the end, either {@code x[i] &lt; x[i + 1]} or <code>x[i]
 * == x[i + 1]</code> and {@code y[i] &le; y[i + 1]}.
 *
 * @param x the first array to be sorted.
 * @param y the second array to be sorted.
 * @param from the index of the first element (inclusive) to be sorted.
 * @param to the index of the last element (exclusive) to be sorted.
 */

SUPPRESS_WARNINGS_KEY_UNCHECKED
public static KEY_GENERIC void quickSort(final KEY_GENERIC_TYPE[] x, final KEY_GENERIC_TYPE[] y,
final int from, final int to) {
 final int len = to - from;
 if (len < QUICKSORT_NO_REC) {
  selectionSort(x, y, from, to);
  return;
 }
 // Choose a partition element, v
 int m = from + len / 2;
 int l = from;
 int n = to - 1;
 if (len > QUICKSORT_MEDIAN_OF_9) { // Big arrays, pseudomedian of 9
  int s = len / 8;
  l = med3(x, y, l, l + s, l + 2 * s);
  m = med3(x, y, m - s, m, m + s);
  n = med3(x, y, n - 2 * s, n - s, n);
 }
 m = med3(x, y, l, m, n); // Mid-size, med of 3
 final KEY_GENERIC_TYPE v = x[m], w = y[m];
 // Establish Invariant: v* (<v)* (>v)* v*
 int a = from, b = a, c = to - 1, d = c;
 while (true) {
```

```
 int comparison, t;
 while (b <= c && (comparison = (t = KEY_CMP(x[b], v)) == 0 ? KEY_CMP(y[b], w) : t) <= 0) {
  if (comparison == 0) swap(x, y, a++, b);
  b++;
 }
 while (c >= b && (comparison = (t = KEY_CMP(x[c], v)) == 0 ? KEY_CMP(y[c], w) : t) >= 0) {
  if (comparison == 0) swap(x, y, c, d--);
  c--;
 }
 if (b > c) break;
 swap(x, y, b++, c--);
 }
 // Swap partition elements back to middle
 int s;
 s = Math.min(a - from, b - a);
 swap(x, y, from, b - s, s);
 s = Math.min(d - c, to - d - 1);
 swap(x, y, b, to - s, s);
 // Recursively sort non-partition-elements
 if ((s = b - a) > 1) quickSort(x, y, from, from + s);
 if ((s = d - c) > 1) quickSort(x, y, to - s, to);
}

/** Sorts two arrays according to the natural lexicographical ascending order using quicksort.
 *
 * <p>The sorting algorithm is a tuned quicksort adapted from Jon L. Bentley and M. Douglas
 * McIlroy, &ldquo;Engineering a Sort Function&rdquo;, <i>Software: Practice and Experience</i>, 23(11), pages
 * 1249&minus;1265, 1993.
 *
 * <p>This method implements a <em>lexicographical</em> sorting of the arguments. Pairs of
 * elements in the same position in the two provided arrays will be considered a single key, and
 * permuted accordingly. In the end, either {@code x[i] &lt; x[i + 1]} or <code>x[i]
 * == x[i + 1]</code> and {@code y[i] &le; y[i + 1]}.
 *
 * @param x the first array to be sorted.
 * @param y the second array to be sorted.
 */
public static KEY_GENERIC void quickSort(final KEY_GENERIC_TYPE[] x, final KEY_GENERIC_TYPE[] y)
{
 ensureSameLength(x, y);
 quickSort(x, y, 0, x.length);
}

protected static class ForkJoinQuickSort2 KEY_GENERIC extends RecursiveAction {
 private static final long serialVersionUID = 1L;
 private final int from;
 private final int to;
 private final KEY_GENERIC_TYPE[] x, y;
```

```
  public ForkJoinQuickSort2(final KEY_GENERIC_TYPE[] x, final KEY_GENERIC_TYPE[] y, final int from ,
final int to) {
  this.from = from;
  this.to = to;
  this.x = x;
  this.y = y;
  }

  @Override
  SUPPRESS_WARNINGS_KEY_UNCHECKED
  protected void compute() {
  final KEY_GENERIC_TYPE[] x = this.x;
  final KEY_GENERIC_TYPE[] y = this.y;
  final int len = to - from;
  if (len < PARALLEL_QUICKSORT_NO_FORK) {
   quickSort(x, y, from, to);
   return;
  }
  // Choose a partition element, v
  int m = from + len / 2;
  int l = from;
  int n = to - 1;
  int s = len / 8;
  l = med3(x, y, l, l + s, l + 2 * s);
  m = med3(x, y, m - s, m, m + s);
  n = med3(x, y, n - 2 * s, n - s, n);
  m = med3(x, y, l, m, n);
  final KEY_GENERIC_TYPE v = x[m], w = y[m];
  // Establish Invariant: v* (<v)* (>v)* v*
  int a = from, b = a, c = to - 1, d = c;
  while (true) {
   int comparison, t;
   while (b <= c && (comparison = (t = KEY_CMP(x[b], v)) == 0 ? KEY_CMP(y[b], w) : t) <= 0) {
    if (comparison == 0) swap(x, y, a++, b);
    b++;
   }
   while (c >= b && (comparison = (t = KEY_CMP(x[c], v)) == 0 ? KEY_CMP(y[c], w) : t) >= 0) {
    if (comparison == 0) swap(x, y, c, d--);
    c--;
   }
   if (b > c) break;
   swap(x, y, b++, c--);
  }
  // Swap partition elements back to middle
  int t;
  s = Math.min(a - from, b - a);
  swap(x, y, from, b - s, s);
```

```
  s = Math.min(d - c, to - d - 1);
  swap(x, y, b, to - s, s);
  s = b - a;
  t = d - c;
  // Recursively sort non-partition-elements
  if (s > 1 && t > 1) invokeAll(new ForkJoinQuickSort2 KEY_GENERIC_DIAMOND(x, y, from, from + s), new
ForkJoinQuickSort2 KEY_GENERIC_DIAMOND(x, y, to - t, to));
  else if (s > 1) invokeAll(new ForkJoinQuickSort2 KEY_GENERIC_DIAMOND(x, y, from, from + s));
  else invokeAll(new ForkJoinQuickSort2 KEY_GENERIC_DIAMOND(x, y, to - t, to));
 }
}

/** Sorts the specified range of elements of two arrays according to the natural lexicographical
 *  ascending order using a parallel quicksort.
 *
 * <p>The sorting algorithm is a tuned quicksort adapted from Jon L. Bentley and M. Douglas
 * McIlroy, &ldquo;Engineering a Sort Function&rdquo;, <i>Software: Practice and Experience</i>, 23(11), pages
 * 1249&minus;1265, 1993.
 *
 * <p>This method implements a <em>lexicographical</em> sorting of the arguments. Pairs of
 * elements in the same position in the two provided arrays will be considered a single key, and
 * permuted accordingly. In the end, either {@code x[i] &lt; x[i + 1]} or <code>x[i]
 * == x[i + 1]</code> and {@code y[i] &le; y[i + 1]}.
 *
 * <p>This implementation uses a {@link ForkJoinPool} executor service with
 * {@link Runtime#availableProcessors()} parallel threads.
 *
 * @param x the first array to be sorted.
 * @param y the second array to be sorted.
 * @param from the index of the first element (inclusive) to be sorted.
 * @param to the index of the last element (exclusive) to be sorted.
 */
public static KEY_GENERIC void parallelQuickSort(final KEY_GENERIC_TYPE[] x, final
KEY_GENERIC_TYPE[] y, final int from, final int to) {
 if (to - from < PARALLEL_QUICKSORT_NO_FORK) quickSort(x, y, from, to);
 final ForkJoinPool pool = new ForkJoinPool(Runtime.getRuntime().availableProcessors());
 pool.invoke(new ForkJoinQuickSort2 KEY_GENERIC_DIAMOND(x, y, from, to));
 pool.shutdown();
}

/** Sorts two arrays according to the natural lexicographical
 *  ascending order using a parallel quicksort.
 *
 * <p>The sorting algorithm is a tuned quicksort adapted from Jon L. Bentley and M. Douglas
 * McIlroy, &ldquo;Engineering a Sort Function&rdquo;, <i>Software: Practice and Experience</i>, 23(11), pages
 * 1249&minus;1265, 1993.
 *
 * <p>This method implements a <em>lexicographical</em> sorting of the arguments. Pairs of
```

```
 * elements in the same position in the two provided arrays will be considered a single key, and
 * permuted accordingly. In the end, either {@code x[i] &lt; x[i + 1]} or <code>x[i]
 * == x[i + 1]</code> and {@code y[i] &le; y[i + 1]}.
 *
 * <p>This implementation uses a {@link ForkJoinPool} executor service with
 * {@link Runtime#availableProcessors()} parallel threads.
 *
 * @param x the first array to be sorted.
 * @param y the second array to be sorted.
 */
 public static KEY_GENERIC void parallelQuickSort(final KEY_GENERIC_TYPE[] x, final
KEY_GENERIC_TYPE[] y) {
  ensureSameLength(x, y);
  parallelQuickSort(x, y, 0, x.length);
 }



 /** Sorts the specified range of elements according to the natural ascending order using mergesort, using a given
pre-filled support array.
 *
 * <p>This sort is guaranteed to be <i>stable</i>: equal elements will not be reordered as a result
 * of the sort. Moreover, no support arrays will be allocated.

 * @param a the array to be sorted.
 * @param from the index of the first element (inclusive) to be sorted.
 * @param to the index of the last element (exclusive) to be sorted.
 * @param supp a support array containing at least {@code to} elements, and whose entries are identical to those
 * of {@code a} in the specified range.
 */

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public static KEY_GENERIC void mergeSort(final KEY_GENERIC_TYPE a[], final int from, final int to, final
KEY_GENERIC_TYPE supp[]) {
  int len = to - from;

  // Insertion sort on smallest arrays
  if (len < MERGESORT_NO_REC) {
   insertionSort(a, from, to);
   return;
  }

  // Recursively sort halves of a into supp
  final int mid = (from + to) >>> 1;
  mergeSort(supp, from, mid, a);
  mergeSort(supp, mid, to, a);

  // If list is already sorted, just copy from supp to a.  This is an
```

```
 // optimization that results in faster sorts for nearly ordered lists.
 if (KEY_LESSEQ(supp[mid - 1], supp[mid])) {
  System.arraycopy(supp, from, a, from, len);
  return;
 }

 // Merge sorted halves (now in supp) into a
 for(int i = from, p = from, q = mid; i < to; i++) {
  if (q >= to || p < mid && KEY_LESSEQ(supp[p], supp[q])) a[i] = supp[p++];
  else a[i] = supp[q++];
 }
}

/** Sorts the specified range of elements according to the natural ascending order using mergesort.
 *
 * <p>This sort is guaranteed to be <i>stable</i>: equal elements will not be reordered as a result
 * of the sort. An array as large as {@code a} will be allocated by this method.

 * @param a the array to be sorted.
 * @param from the index of the first element (inclusive) to be sorted.
 * @param to the index of the last element (exclusive) to be sorted.
 */
public static KEY_GENERIC void mergeSort(final KEY_GENERIC_TYPE a[], final int from, final int to) {
 mergeSort(a, from, to, a.clone());
}

/** Sorts an array according to the natural ascending order using mergesort.
 *
 * <p>This sort is guaranteed to be <i>stable</i>: equal elements will not be reordered as a result
 * of the sort. An array as large as {@code a} will be allocated by this method.

 * @param a the array to be sorted.
 */
public static KEY_GENERIC void mergeSort(final KEY_GENERIC_TYPE a[]) {
 mergeSort(a, 0, a.length);
}

/** Sorts the specified range of elements according to the order induced by the specified
 * comparator using mergesort, using a given pre-filled support array.
 *
 * <p>This sort is guaranteed to be <i>stable</i>: equal elements will not be reordered as a result
 * of the sort. Moreover, no support arrays will be allocated.

 * @param a the array to be sorted.
 * @param from the index of the first element (inclusive) to be sorted.
 * @param to the index of the last element (exclusive) to be sorted.
 * @param comp the comparator to determine the sorting order.
 * @param supp a support array containing at least {@code to} elements, and whose entries are identical to those
```

```
 * of {@code a} in the specified range.
 */
 public static KEY_GENERIC void mergeSort(final KEY_GENERIC_TYPE a[], final int from, final int to,
KEY_COMPARATOR KEY_GENERIC comp, final KEY_GENERIC_TYPE supp[]) {
 int len = to - from;

 // Insertion sort on smallest arrays
 if (len < MERGESORT_NO_REC) {
  insertionSort(a, from, to, comp);
  return;
 }

 // Recursively sort halves of a into supp
 final int mid = (from + to) >>> 1;
 mergeSort(supp, from, mid, comp, a);
 mergeSort(supp, mid, to, comp, a);

 // If list is already sorted, just copy from supp to a.  This is an
 // optimization that results in faster sorts for nearly ordered lists.
 if (comp.compare(supp[mid - 1], supp[mid]) <= 0) {
  System.arraycopy(supp, from, a, from, len);
  return;
 }

 // Merge sorted halves (now in supp) into a
 for(int i = from, p = from, q = mid; i < to; i++) {
  if (q >= to || p < mid && comp.compare(supp[p], supp[q]) <= 0) a[i] = supp[p++];
  else a[i] = supp[q++];
 }
}

/** Sorts the specified range of elements according to the order induced by the specified
 * comparator using mergesort.
 *
 * <p>This sort is guaranteed to be <i>stable</i>: equal elements will not be reordered as a result
 * of the sort. An array as large as {@code a} will be allocated by this method.
 *
 * @param a the array to be sorted.
 * @param from the index of the first element (inclusive) to be sorted.
 * @param to the index of the last element (exclusive) to be sorted.
 * @param comp the comparator to determine the sorting order.
 */
 public static KEY_GENERIC void mergeSort(final KEY_GENERIC_TYPE a[], final int from, final int to,
KEY_COMPARATOR KEY_GENERIC comp) {
 mergeSort(a, from, to, comp, a.clone());
 }

 /** Sorts an array according to the order induced by the specified
```

```
 * comparator using mergesort.
 *
 * <p>This sort is guaranteed to be <i>stable</i>: equal elements will not be reordered as a result
 * of the sort.  An array as large as {@code a} will be allocated by this method.

 * @param a the array to be sorted.
 * @param comp the comparator to determine the sorting order.
 */
 public static KEY_GENERIC void mergeSort(final KEY_GENERIC_TYPE a[], KEY_COMPARATOR
KEY_GENERIC comp) {
  mergeSort(a, 0, a.length, comp);
 }

#if ! KEY_CLASS_Boolean

 /**
  * Searches a range of the specified array for the specified value using
  * the binary search algorithm. The range must be sorted prior to making this call.
  * If it is not sorted, the results are undefined. If the range contains multiple elements with
  * the specified value, there is no guarantee which one will be found.
  *
  * @param a the array to be searched.
  * @param from  the index of the first element (inclusive) to be searched.
  * @param to  the index of the last element (exclusive) to be searched.
  * @param key the value to be searched for.
  * @return index of the search key, if it is contained in the array;
  *         otherwise, {@code (-(<i>insertion point</i>) - 1)}.  The <i>insertion
  *         point</i> is defined as the the point at which the value would
  *         be inserted into the array: the index of the first
  *         element greater than the key, or the length of the array, if all
  *         elements in the array are less than the specified key.  Note
  *         that this guarantees that the return value will be &ge; 0 if
  *         and only if the key is found.
  * @see java.util.Arrays
  */
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public static KEY_GENERIC int binarySearch(final KEY_GENERIC_TYPE[] a, int from, int to, final
KEY_GENERIC_TYPE key) {
  KEY_GENERIC_TYPE midVal;
  to--;
  while (from <= to) {
   final int mid = (from + to) >>> 1;
   midVal = a[mid];
#if KEYS_PRIMITIVE
   if (midVal < key) from = mid + 1;
   else if (midVal > key) to = mid - 1;
   else return mid;
#else
```

```
  final int cmp = ((Comparable KEY_SUPER_GENERIC)midVal).compareTo(key);
  if (cmp < 0) from = mid + 1;
  else if (cmp > 0) to = mid - 1;
  else return mid;
#endif
  }
  return -(from + 1);
 }

 /**
  * Searches an array for the specified value using
  * the binary search algorithm. The range must be sorted prior to making this call.
  * If it is not sorted, the results are undefined. If the range contains multiple elements with
  * the specified value, there is no guarantee which one will be found.
  *
  * @param a the array to be searched.
  * @param key the value to be searched for.
  * @return index of the search key, if it is contained in the array;
  *         otherwise, {@code (-(<i>insertion point</i>) - 1)}.  The <i>insertion
  *         point</i> is defined as the the point at which the value would
  *         be inserted into the array: the index of the first
  *         element greater than the key, or the length of the array, if all
  *         elements in the array are less than the specified key.  Note
  *         that this guarantees that the return value will be &ge; 0 if
  *         and only if the key is found.
  * @see java.util.Arrays
  */
 public static KEY_GENERIC int binarySearch(final KEY_GENERIC_TYPE[] a, final KEY_GENERIC_TYPE
key) {
  return binarySearch(a, 0, a.length, key);
 }

 /**
  * Searches a range of the specified array for the specified value using
  * the binary search algorithm and a specified comparator. The range must be sorted following the comparator prior
to making this call.
  * If it is not sorted, the results are undefined. If the range contains multiple elements with
  * the specified value, there is no guarantee which one will be found.
  *
  * @param a the array to be searched.
  * @param from  the index of the first element (inclusive) to be searched.
  * @param to  the index of the last element (exclusive) to be searched.
  * @param key the value to be searched for.
  * @param c a comparator.
  * @return index of the search key, if it is contained in the array;
  *         otherwise, {@code (-(<i>insertion point</i>) - 1)}.  The <i>insertion
  *         point</i> is defined as the the point at which the value would
  *         be inserted into the array: the index of the first
```

```
 *            element greater than the key, or the length of the array, if all
 *            elements in the array are less than the specified key.  Note
 *            that this guarantees that the return value will be &ge; 0 if
 *            and only if the key is found.
 * @see java.util.Arrays
 */
 public static KEY_GENERIC int binarySearch(final KEY_GENERIC_TYPE[] a, int from, int to, final
KEY_GENERIC_TYPE key, final KEY_COMPARATOR KEY_GENERIC c) {
  KEY_GENERIC_TYPE midVal;
  to--;
  while (from <= to) {
   final int mid = (from + to) >>> 1;
   midVal = a[mid];
   final int cmp = c.compare(midVal, key);
   if (cmp < 0) from = mid + 1;
   else if (cmp > 0) to = mid - 1;
   else return mid; // key found
  }
  return -(from + 1);
 }

 /**
  * Searches an array for the specified value using
  * the binary search algorithm and a specified comparator. The range must be sorted following the comparator prior
to making this call.
  * If it is not sorted, the results are undefined. If the range contains multiple elements with
  * the specified value, there is no guarantee which one will be found.
  *
  * @param a the array to be searched.
  * @param key the value to be searched for.
  * @param c a comparator.
  * @return index of the search key, if it is contained in the array;
  *            otherwise, {@code (-(<i>insertion point</i>) - 1)}.  The <i>insertion
  *            point</i> is defined as the the point at which the value would
  *            be inserted into the array: the index of the first
  *            element greater than the key, or the length of the array, if all
  *            elements in the array are less than the specified key.  Note
  *            that this guarantees that the return value will be &ge; 0 if
  *            and only if the key is found.
  * @see java.util.Arrays
  */
 public static KEY_GENERIC int binarySearch(final KEY_GENERIC_TYPE[] a, final KEY_GENERIC_TYPE
key, final KEY_COMPARATOR KEY_GENERIC c) {
  return binarySearch(a, 0, a.length, key, c);
 }


#if KEYS_PRIMITIVE
```

```
 /** The size of a digit used during radix sort (must be a power of 2). */
 private static final int DIGIT_BITS = 8;
 /** The mask to extract a digit of {@link #DIGIT_BITS} bits. */
 private static final int DIGIT_MASK = (1 << DIGIT_BITS) - 1;
 /** The number of digits per element. */
 private static final int DIGITS_PER_ELEMENT = KEY_CLASS.SIZE / DIGIT_BITS;
 private static final int RADIXSORT_NO_REC = 1024;
 private static final int PARALLEL_RADIXSORT_NO_FORK = 1024;

 /** This method fixes negative numbers so that the combination exponent/significand is lexicographically sorted. */
#if KEY_CLASS_Double
 private static final long fixDouble(final double d) {
  final long l = Double.doubleToLongBits(d);
  return l >= 0 ? l : l ^ 0x7FFFFFFFFFFFFFFFL;
 }
#elif KEY_CLASS_Float
 private static final int fixFloat(final float f) {
  final int i = Float.floatToIntBits(f);
  return i >= 0 ? i : i ^ 0x7FFFFFFF;
 }
#endif


 /** Sorts the specified array using radix sort.
  *
  * <p>The sorting algorithm is a tuned radix sort adapted from Peter M. McIlroy, Keith Bostic and M. Douglas
  * McIlroy, &ldquo;Engineering radix sort&rdquo;, <i>Computing Systems</i>, 6(1), pages 5&minus;27 (1993).
  *
  * <p>This implementation is significantly faster than quicksort
  * already at small sizes (say, more than 10000 elements), but it can only
  * sort in ascending order.
  *
  * @param a the array to be sorted.
  */
 public static void radixSort(final KEY_TYPE[] a) {
  radixSort(a, 0, a.length);
 }

 /** Sorts the specified range of an array using radix sort.
  *
  * <p>The sorting algorithm is a tuned radix sort adapted from Peter M. McIlroy, Keith Bostic and M. Douglas
  * McIlroy, &ldquo;Engineering radix sort&rdquo;, <i>Computing Systems</i>, 6(1), pages 5&minus;27 (1993).
  *
  * <p>This implementation is significantly faster than quicksort
  * already at small sizes (say, more than 10000 elements), but it can only
  * sort in ascending order.
  *
  * @param a the array to be sorted.
```

```
 * @param from the index of the first element (inclusive) to be sorted.
 * @param to the index of the last element (exclusive) to be sorted.
 */
public static void radixSort(final KEY_TYPE[] a, final int from, final int to) {
 if (to - from < RADIXSORT_NO_REC) {
  quickSort(a, from, to);
  return;
 }

 final int maxLevel = DIGITS_PER_ELEMENT - 1;

 final int stackSize = ((1 << DIGIT_BITS) - 1) * (DIGITS_PER_ELEMENT - 1) + 1;
 int stackPos = 0;
 final int[] offsetStack = new int[stackSize];
 final int[] lengthStack = new int[stackSize];
 final int[] levelStack = new int[stackSize];

 offsetStack[stackPos] = from;
 lengthStack[stackPos] = to - from;
 levelStack[stackPos++] = 0;

 final int[] count = new int[1 << DIGIT_BITS];
 final int[] pos = new int[1 << DIGIT_BITS];

 while(stackPos > 0) {
  final int first = offsetStack[--stackPos];
  final int length = lengthStack[stackPos];
  final int level = levelStack[stackPos];
#if KEY_CLASS_Character
  final int signMask = 0;
#else
  final int signMask = level % DIGITS_PER_ELEMENT == 0 ? 1 << DIGIT_BITS - 1 : 0;
#endif
  final int shift = (DIGITS_PER_ELEMENT - 1 - level % DIGITS_PER_ELEMENT) * DIGIT_BITS; // This is the
shift that extract the right byte from a key

  // Count keys.
  for(int i = first + length; i-- != first;) count[INT(KEY2LEXINT(a[i]) >>> shift & DIGIT_MASK ^ signMask)]++;
  // Compute cumulative distribution
  int lastUsed = -1;
  for (int i = 0, p = first; i < 1 << DIGIT_BITS; i++) {
   if (count[i] != 0) lastUsed = i;
   pos[i] = (p += count[i]);
  }

  final int end = first + length - count[lastUsed];
  // i moves through the start of each block
  for(int i = first, c = -1, d; i <= end; i += count[c], count[c] = 0) {
```

```
  KEY_TYPE t = a[i];
  c = INT(KEY2LEXINT(t) >>> shift & DIGIT_MASK ^ signMask);

  if (i < end) { // When all slots are OK, the last slot is necessarily OK.
   while ((d = --pos[c]) > i) {
    final KEY_TYPE z = t;
    t = a[d];
    a[d] = z;
    c = INT(KEY2LEXINT(t) >>> shift & DIGIT_MASK ^ signMask);
   }
   a[i] = t;
  }

  if (level < maxLevel && count[c] > 1) {
   if (count[c] < RADIXSORT_NO_REC) quickSort(a, i, i + count[c]);
   else {
    offsetStack[stackPos] = i;
    lengthStack[stackPos] = count[c];
    levelStack[stackPos++] = level + 1;
   }
  }
 }
}

protected static final class Segment {
 protected final int offset, length, level;
 protected Segment(final int offset, final int length, final int level) {
  this.offset = offset;
  this.length = length;
  this.level = level;
 }

 @Override
 public String toString() { return "Segment [offset=" + offset + ", length=" + length + ", level=" + level + "]"; }
}

protected static final Segment POISON_PILL = new Segment(-1, -1, -1);


/** Sorts the specified range of an array using parallel radix sort.
 *
 * <p>The sorting algorithm is a tuned radix sort adapted from Peter M. McIlroy, Keith Bostic and M. Douglas
 * McIlroy, &ldquo;Engineering radix sort&rdquo;, <i>Computing Systems</i>, 6(1), pages 5&minus;27 (1993).
 *
 * <p>This implementation uses a pool of {@link Runtime#availableProcessors()} threads.
 *
 * @param a the array to be sorted.
```

```
 * @param from the index of the first element (inclusive) to be sorted.
 * @param to the index of the last element (exclusive) to be sorted.
 */
public static void parallelRadixSort(final KEY_TYPE[] a, final int from, final int to) {
 if (to - from < PARALLEL_RADIXSORT_NO_FORK) {
  quickSort(a, from, to);
  return;
 }
 final int maxLevel = DIGITS_PER_ELEMENT - 1;
 final LinkedBlockingQueue<Segment> queue = new LinkedBlockingQueue<>();
 queue.add(new Segment(from, to - from, 0));
 final AtomicInteger queueSize = new AtomicInteger(1);
 final int numberOfThreads = Runtime.getRuntime().availableProcessors();
 final ExecutorService executorService = Executors.newFixedThreadPool(numberOfThreads,
Executors.defaultThreadFactory());
 final ExecutorCompletionService<Void> executorCompletionService = new
ExecutorCompletionService<>(executorService);

 for(int j = numberOfThreads; j-- != 0;) executorCompletionService.submit(() -> {
  final int[] count = new int[1 << DIGIT_BITS];
  final int[] pos = new int[1 << DIGIT_BITS];

  for(;;) {
   if (queueSize.get() == 0) for(int i = numberOfThreads; i-- != 0;) queue.add(POISON_PILL);
   final Segment segment = queue.take();
   if (segment == POISON_PILL) return null;

   final int first = segment.offset;
   final int length = segment.length;
   final int level = segment.level;

#if KEY_CLASS_Character
   final int signMask = 0;
#else
   final int signMask = level % DIGITS_PER_ELEMENT == 0 ? 1 << DIGIT_BITS - 1 : 0;
#endif
   final int shift = (DIGITS_PER_ELEMENT - 1 - level % DIGITS_PER_ELEMENT) * DIGIT_BITS; // This is the
shift that extract the right byte from a key

   // Count keys.
   for(int i = first + length; i-- != first;) count[INT(KEY2LEXINT(a[i]) >>> shift & DIGIT_MASK ^ signMask)]++;
   // Compute cumulative distribution
   int lastUsed = -1;
   for(int i = 0, p = first; i < 1 << DIGIT_BITS; i++) {
    if (count[i] != 0) lastUsed = i;
    pos[i] = (p += count[i]);
   }
```

```
   final int end = first + length - count[lastUsed];
   // i moves through the start of each block
   for(int i = first, c = -1, d; i <= end; i += count[c], count[c] = 0) {
    KEY_TYPE t = a[i];
    c = INT(KEY2LEXINT(t) >>> shift & DIGIT_MASK ^ signMask);
    if (i < end) {
     while((d = --pos[c]) > i) {
      final KEY_TYPE z = t;
      t = a[d];
      a[d] = z;
      c = INT(KEY2LEXINT(t) >>> shift & DIGIT_MASK ^ signMask);
     }
     a[i] = t;
    }

    if (level < maxLevel && count[c] > 1) {
     if (count[c] < PARALLEL_RADIXSORT_NO_FORK) quickSort(a, i, i + count[c]);
     else {
      queueSize.incrementAndGet();
      queue.add(new Segment(i, count[c], level + 1));
     }
    }
   }
   queueSize.decrementAndGet();
  }
 });

 Throwable problem = null;
 for(int i = numberOfThreads; i-- != 0;)
  try {
   executorCompletionService.take().get();
  }
  catch(Exception e) {
   problem = e.getCause(); // We keep only the last one. They will be logged anyway.
  }

 executorService.shutdown();
 if (problem != null) throw (problem instanceof RuntimeException) ? (RuntimeException)problem : new
RuntimeException(problem);
 }

 /** Sorts the specified array using parallel radix sort.
  *
  * <p>The sorting algorithm is a tuned radix sort adapted from Peter M. McIlroy, Keith Bostic and M. Douglas
  * McIlroy, &ldquo;Engineering radix sort&rdquo;, <i>Computing Systems</i>, 6(1), pages 5&minus;27 (1993).
  *
  * <p>This implementation uses a pool of {@link Runtime#availableProcessors()} threads.
  *
```

```
 * @param a the array to be sorted.
 */
public static void parallelRadixSort(final KEY_TYPE[] a) {
 parallelRadixSort(a, 0, a.length);
}

/** Sorts the specified array using indirect radix sort.
 *
 * <p>The sorting algorithm is a tuned radix sort adapted from Peter M. McIlroy, Keith Bostic and M. Douglas
 * McIlroy, &ldquo;Engineering radix sort&rdquo;, <i>Computing Systems</i>, 6(1), pages 5&minus;27 (1993).
 *
 * <p>This method implement an <em>indirect</em> sort. The elements of {@code perm} (which must
 * be exactly the numbers in the interval {@code [0..perm.length)}) will be permuted so that
 * {@code a[perm[i]] &le; a[perm[i + 1]]}.
 *
 * <p>This implementation will allocate, in the stable case, a support array as large as {@code perm} (note that the stable
 * version is slightly faster).
 *
 * @param perm a permutation array indexing {@code a}.
 * @param a the array to be sorted.
 * @param stable whether the sorting algorithm should be stable.
 */
public static void radixSortIndirect(final int[] perm, final KEY_TYPE[] a, final boolean stable) {
 radixSortIndirect(perm, a, 0, perm.length, stable);
}

/** Sorts the specified array using indirect radix sort.
 *
 * <p>The sorting algorithm is a tuned radix sort adapted from Peter M. McIlroy, Keith Bostic and M. Douglas
 * McIlroy, &ldquo;Engineering radix sort&rdquo;, <i>Computing Systems</i>, 6(1), pages 5&minus;27 (1993).
 *
 * <p>This method implement an <em>indirect</em> sort. The elements of {@code perm} (which must
 * be exactly the numbers in the interval {@code [0..perm.length)}) will be permuted so that
 * {@code a[perm[i]] &le; a[perm[i + 1]]}.
 *
 * <p>This implementation will allocate, in the stable case, a support array as large as {@code perm} (note that the stable
 * version is slightly faster).
 *
 * @param perm a permutation array indexing {@code a}.
 * @param a the array to be sorted.
 * @param from the index of the first element of {@code perm} (inclusive) to be permuted.
 * @param to the index of the last element of {@code perm} (exclusive) to be permuted.
 * @param stable whether the sorting algorithm should be stable.
 */
public static void radixSortIndirect(final int[] perm, final KEY_TYPE[] a, final int from, final int to, final boolean stable) {
```

```
 if (to - from < RADIXSORT_NO_REC) {
  insertionSortIndirect(perm, a, from, to);
  return;
 }

 final int maxLevel = DIGITS_PER_ELEMENT - 1;

 final int stackSize = ((1 << DIGIT_BITS) - 1) * (DIGITS_PER_ELEMENT - 1) + 1;
 int stackPos = 0;
 final int[] offsetStack = new int[stackSize];
 final int[] lengthStack = new int[stackSize];
 final int[] levelStack = new int[stackSize];

 offsetStack[stackPos] = from;
 lengthStack[stackPos] = to - from;
 levelStack[stackPos++] = 0;

 final int[] count = new int[1 << DIGIT_BITS];
 final int[] pos = new int[1 << DIGIT_BITS];
 final int[] support = stable ? new int[perm.length] : null;

 while(stackPos > 0) {
  final int first = offsetStack[--stackPos];
  final int length = lengthStack[stackPos];
  final int level = levelStack[stackPos];
#if KEY_CLASS_Character
  final int signMask = 0;
#else
  final int signMask = level % DIGITS_PER_ELEMENT == 0 ? 1 << DIGIT_BITS - 1 : 0;
#endif

  final int shift = (DIGITS_PER_ELEMENT - 1 - level % DIGITS_PER_ELEMENT) * DIGIT_BITS; // This is the
shift that extract the right byte from a key

  // Count keys.
  for(int i = first + length; i-- != first;) count[INT(KEY2LEXINT(a[perm[i]]) >>> shift & DIGIT_MASK ^
signMask)]++;
  // Compute cumulative distribution
  int lastUsed = -1;
  for (int i = 0, p = stable ? 0 : first; i < 1 << DIGIT_BITS; i++) {
   if (count[i] != 0) lastUsed = i;
   pos[i] = (p += count[i]);
  }

  if (stable) {
   for(int i = first + length; i-- != first;) support[--pos[INT(KEY2LEXINT(a[perm[i]]) >>> shift & DIGIT_MASK ^
signMask)]] = perm[i];
   System.arraycopy(support, 0, perm, first, length);
```

```
  for(int i = 0, p = first; i <= lastUsed; i++) {
   if (level < maxLevel && count[i] > 1) {
    if (count[i] < RADIXSORT_NO_REC) insertionSortIndirect(perm, a, p, p + count[i]);
    else {
     offsetStack[stackPos] = p;
     lengthStack[stackPos] = count[i];
     levelStack[stackPos++] = level + 1;
    }
   }
   p += count[i];
  }
  java.util.Arrays.fill(count, 0);
 }
 else {
  final int end = first + length - count[lastUsed];
  // i moves through the start of each block
  for(int i = first, c = -1, d; i <= end; i += count[c], count[c] = 0) {
   int t = perm[i];
   c = INT(KEY2LEXINT(a[t]) >>> shift & DIGIT_MASK ^ signMask);

   if (i < end) { // When all slots are OK, the last slot is necessarily OK.
    while((d = --pos[c]) > i) {
     final int z = t;
     t = perm[d];
     perm[d] = z;
     c = INT(KEY2LEXINT(a[t]) >>> shift & DIGIT_MASK ^ signMask);
    }
    perm[i] = t;
   }

   if (level < maxLevel && count[c] > 1) {
    if (count[c] < RADIXSORT_NO_REC) insertionSortIndirect(perm, a, i, i + count[c]);
    else {
     offsetStack[stackPos] = i;
     lengthStack[stackPos] = count[c];
     levelStack[stackPos++] = level + 1;
    }
   }
  }
 }
}

/** Sorts the specified range of an array using parallel indirect radix sort.
 *
 * <p>The sorting algorithm is a tuned radix sort adapted from Peter M. McIlroy, Keith Bostic and M. Douglas
 * McIlroy, &ldquo;Engineering radix sort&rdquo;, <i>Computing Systems</i>, 6(1), pages 5&minus;27 (1993).
 *
```

```
 * <p>This method implement an <em>indirect</em> sort. The elements of {@code perm} (which must
 * be exactly the numbers in the interval {@code [0..perm.length)}) will be permuted so that
 * {@code a[perm[i]] &le; a[perm[i + 1]]}.
 *
 * <p>This implementation uses a pool of {@link Runtime#availableProcessors()} threads.
 *
 * @param perm a permutation array indexing {@code a}.
 * @param a the array to be sorted.
 * @param from the index of the first element (inclusive) to be sorted.
 * @param to the index of the last element (exclusive) to be sorted.
 * @param stable whether the sorting algorithm should be stable.
 */
 public static void parallelRadixSortIndirect(final int perm[], final KEY_TYPE[] a, final int from, final int to, final
boolean stable) {
  if (to - from < PARALLEL_RADIXSORT_NO_FORK) {
   radixSortIndirect(perm, a, from, to, stable);
   return;
  }
  final int maxLevel = DIGITS_PER_ELEMENT - 1;
  final LinkedBlockingQueue<Segment> queue = new LinkedBlockingQueue<>();
  queue.add(new Segment(from, to - from, 0));
  final AtomicInteger queueSize = new AtomicInteger(1);
  final int numberOfThreads = Runtime.getRuntime().availableProcessors();
  final ExecutorService executorService = Executors.newFixedThreadPool(numberOfThreads,
Executors.defaultThreadFactory());
  final ExecutorCompletionService<Void> executorCompletionService = new
ExecutorCompletionService<>(executorService);
  final int[] support = stable ? new int[perm.length] : null;

  for(int j = numberOfThreads; j-- != 0;) executorCompletionService.submit(() -> {
   final int[] count = new int[1 << DIGIT_BITS];
   final int[] pos = new int[1 << DIGIT_BITS];

   for(;;) {
    if (queueSize.get() == 0) for(int i = numberOfThreads; i-- != 0;) queue.add(POISON_PILL);
    final Segment segment = queue.take();
    if (segment == POISON_PILL) return null;

    final int first = segment.offset;
    final int length = segment.length;
    final int level = segment.level;

#if KEY_CLASS_Character
    final int signMask = 0;
#else
    final int signMask = level % DIGITS_PER_ELEMENT == 0 ? 1 << DIGIT_BITS - 1 : 0;
#endif
    final int shift = (DIGITS_PER_ELEMENT - 1 - level % DIGITS_PER_ELEMENT) * DIGIT_BITS; // This is the
```

shift that extract the right byte from a key

```
   // Count keys.
   for(int i = first + length; i-- != first;) count[INT(KEY2LEXINT(a[perm[i]]) >>> shift & DIGIT_MASK ^
signMask)]++;
   // Compute cumulative distribution
   int lastUsed = -1;
   for (int i = 0, p = first; i < 1 << DIGIT_BITS; i++) {
    if (count[i] != 0) lastUsed = i;
    pos[i] = (p += count[i]);
   }

   if (stable) {
    for(int i = first + length; i-- != first;) support[--pos[INT(KEY2LEXINT(a[perm[i]]) >>> shift & DIGIT_MASK ^
signMask)]] = perm[i];
    System.arraycopy(support, first, perm, first, length);
    for(int i = 0, p = first; i <= lastUsed; i++) {
     if (level < maxLevel && count[i] > 1) {
      if (count[i] < PARALLEL_RADIXSORT_NO_FORK) radixSortIndirect(perm, a, p, p + count[i], stable);
      else {
       queueSize.incrementAndGet();
       queue.add(new Segment(p, count[i], level + 1));
      }
     }
     p += count[i];
    }
    java.util.Arrays.fill(count, 0);
   }
   else {
    final int end = first + length - count[lastUsed];
    // i moves through the start of each block
    for(int i = first, c = -1, d; i <= end; i += count[c], count[c] = 0) {
     int t = perm[i];
     c = INT(KEY2LEXINT(a[t]) >>> shift & DIGIT_MASK ^ signMask);

     if (i < end) { // When all slots are OK, the last slot is necessarily OK.
      while((d = --pos[c]) > i) {
       final int z = t;
       t = perm[d];
       perm[d] = z;
       c = INT(KEY2LEXINT(a[t]) >>> shift & DIGIT_MASK ^ signMask);
      }
      perm[i] = t;
     }

     if (level < maxLevel && count[c] > 1) {
      if (count[c] < PARALLEL_RADIXSORT_NO_FORK) radixSortIndirect(perm, a, i, i + count[c], stable);
```

```
      else {
       queueSize.incrementAndGet();
       queue.add(new Segment(i, count[c], level + 1));
      }
     }
    }
   }
   queueSize.decrementAndGet();
  }
 });

 Throwable problem = null;
 for(int i = numberOfThreads; i-- != 0;)
  try {
   executorCompletionService.take().get();
  }
  catch(Exception e) {
   problem = e.getCause(); // We keep only the last one. They will be logged anyway.
  }

 executorService.shutdown();
 if (problem != null) throw (problem instanceof RuntimeException) ? (RuntimeException)problem : new
RuntimeException(problem);
 }

 /** Sorts the specified array using parallel indirect radix sort.
  *
  * <p>The sorting algorithm is a tuned radix sort adapted from Peter M. McIlroy, Keith Bostic and M. Douglas
  * McIlroy, &ldquo;Engineering radix sort&rdquo;, <i>Computing Systems</i>, 6(1), pages 5&minus;27 (1993).
  *
  * <p>This method implement an <em>indirect</em> sort. The elements of {@code perm} (which must
  * be exactly the numbers in the interval {@code [0..perm.length)}) will be permuted so that
  * {@code a[perm[i]] &le; a[perm[i + 1]]}.
  *
  * <p>This implementation uses a pool of {@link Runtime#availableProcessors()} threads.
  *
  * @param perm a permutation array indexing {@code a}.
  * @param a the array to be sorted.
  * @param stable whether the sorting algorithm should be stable.
  */
 public static void parallelRadixSortIndirect(final int perm[], final KEY_TYPE[] a, final boolean stable) {
  parallelRadixSortIndirect(perm, a, 0, a.length, stable);
 }

 /** Sorts the specified pair of arrays lexicographically using radix sort.
  * <p>The sorting algorithm is a tuned radix sort adapted from Peter M. McIlroy, Keith Bostic and M. Douglas
  * McIlroy, &ldquo;Engineering radix sort&rdquo;, <i>Computing Systems</i>, 6(1), pages 5&minus;27 (1993).
  *
```

```
 * <p>This method implements a <em>lexicographical</em> sorting of the arguments. Pairs of elements
 * in the same position in the two provided arrays will be considered a single key, and permuted
 * accordingly. In the end, either {@code a[i] &lt; a[i + 1]} or {@code a[i] == a[i + 1]} and {@code b[i] &le; b[i +
1]}.
 *
 * @param a the first array to be sorted.
 * @param b the second array to be sorted.
 */

public static void radixSort(final KEY_TYPE[] a, final KEY_TYPE[] b) {
 ensureSameLength(a, b);
 radixSort(a, b, 0, a.length);
}

/** Sorts the specified range of elements of two arrays using radix sort.
 *
 * <p>The sorting algorithm is a tuned radix sort adapted from Peter M. McIlroy, Keith Bostic and M. Douglas
 * McIlroy, &ldquo;Engineering radix sort&rdquo;, <i>Computing Systems</i>, 6(1), pages 5&minus;27 (1993).
 *
 * <p>This method implements a <em>lexicographical</em> sorting of the arguments. Pairs of elements
 * in the same position in the two provided arrays will be considered a single key, and permuted
 * accordingly. In the end, either {@code a[i] &lt; a[i + 1]} or {@code a[i] == a[i + 1]} and {@code b[i] &le; b[i +
1]}.
 *
 * @param a the first array to be sorted.
 * @param b the second array to be sorted.
 * @param from the index of the first element (inclusive) to be sorted.
 * @param to the index of the last element (exclusive) to be sorted.
 */
public static void radixSort(final KEY_TYPE[] a, final KEY_TYPE[] b, final int from, final int to) {
 if (to - from < RADIXSORT_NO_REC) {
  selectionSort(a, b, from, to);
  return;
 }

 final int layers = 2;
 final int maxLevel = DIGITS_PER_ELEMENT * layers - 1;

 final int stackSize = ((1 << DIGIT_BITS) - 1) * (layers * DIGITS_PER_ELEMENT - 1) + 1;
 int stackPos = 0;
 final int[] offsetStack = new int[stackSize];
 final int[] lengthStack = new int[stackSize];
 final int[] levelStack = new int[stackSize];

 offsetStack[stackPos] = from;
 lengthStack[stackPos] = to - from;
 levelStack[stackPos++] = 0;
```

```
  final int[] count = new int[1 << DIGIT_BITS];
  final int[] pos = new int[1 << DIGIT_BITS];

  while(stackPos > 0) {
   final int first = offsetStack[--stackPos];
   final int length = lengthStack[stackPos];
   final int level = levelStack[stackPos];
#if KEY_CLASS_Character
   final int signMask = 0;
#else
   final int signMask = level % DIGITS_PER_ELEMENT == 0 ? 1 << DIGIT_BITS - 1 : 0;
#endif

   final KEY_TYPE[] k = level < DIGITS_PER_ELEMENT ? a : b; // This is the key array
   final int shift = (DIGITS_PER_ELEMENT - 1 - level % DIGITS_PER_ELEMENT) * DIGIT_BITS; // This is the
shift that extract the right byte from a key

   // Count keys.
   for(int i = first + length; i-- != first;) count[INT(KEY2LEXINT(k[i]) >>> shift & DIGIT_MASK ^ signMask)]++;

   // Compute cumulative distribution
   int lastUsed = -1;
   for (int i = 0, p = first; i < 1 << DIGIT_BITS; i++) {
    if (count[i] != 0) lastUsed = i;
    pos[i] = (p += count[i]);
   }


   final int end = first + length - count[lastUsed];
   // i moves through the start of each block
   for(int i = first, c = -1, d; i <= end; i += count[c], count[c] = 0) {
    KEY_TYPE t = a[i];
    KEY_TYPE u = b[i];
    c = INT(KEY2LEXINT(k[i]) >>> shift & DIGIT_MASK ^ signMask);

    if (i < end) { // When all slots are OK, the last slot is necessarily OK.
     while((d = --pos[c]) > i) {
      c = INT(KEY2LEXINT(k[d]) >>> shift & DIGIT_MASK ^ signMask);
      KEY_TYPE z = t;
      t = a[d];
      a[d] = z;
      z = u;
      u = b[d];
      b[d] = z;
     }
     a[i] = t;
     b[i] = u;
```

```
   }

   if (level < maxLevel && count[c] > 1) {
    if (count[c] < RADIXSORT_NO_REC) selectionSort(a, b, i, i + count[c]);
    else {
     offsetStack[stackPos] = i;
     lengthStack[stackPos] = count[c];
     levelStack[stackPos++] = level + 1;
    }
   }
  }
 }
}

 /** Sorts the specified range of elements of two arrays using a parallel radix sort.
  *
  * <p>The sorting algorithm is a tuned radix sort adapted from Peter M. McIlroy, Keith Bostic and M. Douglas
  * McIlroy, &ldquo;Engineering radix sort&rdquo;, <i>Computing Systems</i>, 6(1), pages 5&minus;27 (1993).
  *
  * <p>This method implements a <em>lexicographical</em> sorting of the arguments. Pairs of elements
  * in the same position in the two provided arrays will be considered a single key, and permuted
  * accordingly. In the end, either {@code a[i] &lt; a[i + 1]} or {@code a[i] == a[i + 1]} and {@code b[i] &le; b[i +
1]}.
  *
  * <p>This implementation uses a pool of {@link Runtime#availableProcessors()} threads.
  *
  * @param a the first array to be sorted.
  * @param b the second array to be sorted.
  * @param from the index of the first element (inclusive) to be sorted.
  * @param to the index of the last element (exclusive) to be sorted.
  */
 public static void parallelRadixSort(final KEY_TYPE[] a, final KEY_TYPE[] b, final int from, final int to) {
  if (to - from < PARALLEL_RADIXSORT_NO_FORK) {
   quickSort(a, b, from, to);
   return;
  }
  final int layers = 2;
  if (a.length != b.length) throw new IllegalArgumentException("Array size mismatch.");
  final int maxLevel = DIGITS_PER_ELEMENT * layers - 1;
  final LinkedBlockingQueue<Segment> queue = new LinkedBlockingQueue<>();
  queue.add(new Segment(from, to - from, 0));
  final AtomicInteger queueSize = new AtomicInteger(1);
  final int numberOfThreads = Runtime.getRuntime().availableProcessors();
  final ExecutorService executorService = Executors.newFixedThreadPool(numberOfThreads,
Executors.defaultThreadFactory());
  final ExecutorCompletionService<Void> executorCompletionService = new
ExecutorCompletionService<>(executorService);
  for (int j = numberOfThreads; j-- != 0;) executorCompletionService.submit(() -> {
```

```
final int[] count = new int[1 << DIGIT_BITS];
final int[] pos = new int[1 << DIGIT_BITS];
for (;;) {
 if (queueSize.get() == 0) for (int i = numberOfThreads; i-- != 0;)
  queue.add(POISON_PILL);
 final Segment segment = queue.take();
 if (segment == POISON_PILL) return null;
 final int first = segment.offset;
 final int length = segment.length;
 final int level = segment.level;
 final int signMask = level % DIGITS_PER_ELEMENT == 0 ? 1 << DIGIT_BITS - 1 : 0;
 final KEY_TYPE[] k = level < DIGITS_PER_ELEMENT ? a : b; // This is the key array
 final int shift = (DIGITS_PER_ELEMENT - 1 - level % DIGITS_PER_ELEMENT) * DIGIT_BITS;
 // Count keys.
 for (int i = first + length; i-- != first;)
  count[INT(KEY2LEXINT(k[i]) >>> shift & DIGIT_MASK ^ signMask)]++;
 // Compute cumulative distribution
 int lastUsed = -1;
 for (int i = 0, p = first; i < 1 << DIGIT_BITS; i++) {
  if (count[i] != 0) lastUsed = i;
  pos[i] = (p += count[i]);
 }
 final int end = first + length - count[lastUsed];
 for (int i = first, c = -1, d; i <= end; i += count[c], count[c] = 0) {
  KEY_TYPE t = a[i];
  KEY_TYPE u = b[i];
  c = INT(KEY2LEXINT(k[i]) >>> shift & DIGIT_MASK ^ signMask);
  if (i < end) { // When all slots are OK, the last slot is necessarily OK.
   while ((d = --pos[c]) > i) {
    c = INT(KEY2LEXINT(k[d]) >>> shift & DIGIT_MASK ^ signMask);
    final KEY_TYPE z = t;
    final KEY_TYPE w = u;
    t = a[d];
    u = b[d];
    a[d] = z;
    b[d] = w;
   }
   a[i] = t;
   b[i] = u;
  }
  if (level < maxLevel && count[c] > 1) {
   if (count[c] < PARALLEL_RADIXSORT_NO_FORK) quickSort(a, b, i, i + count[c]);
   else {
    queueSize.incrementAndGet();
    queue.add(new Segment(i, count[c], level + 1));
   }
  }
 }
```

```
    queueSize.decrementAndGet();
   }
  });
 Throwable problem = null;
 for (int i = numberOfThreads; i-- != 0;)
  try {
   executorCompletionService.take().get();
  }
  catch (Exception e) {
   problem = e.getCause(); // We keep only the last one. They will be logged anyway.
  }
 executorService.shutdown();
 if (problem != null) throw (problem instanceof RuntimeException) ? (RuntimeException)problem : new
RuntimeException(problem);
 }


 /** Sorts two arrays using a parallel radix sort.
  *
  * <p>The sorting algorithm is a tuned radix sort adapted from Peter M. McIlroy, Keith Bostic and M. Douglas
  * McIlroy, &ldquo;Engineering radix sort&rdquo;, <i>Computing Systems</i>, 6(1), pages 5&minus;27 (1993).
  *
  * <p>This method implements a <em>lexicographical</em> sorting of the arguments. Pairs of elements
  * in the same position in the two provided arrays will be considered a single key, and permuted
  * accordingly. In the end, either {@code a[i] &lt; a[i + 1]} or {@code a[i] == a[i + 1]} and {@code b[i] &le; b[i +
1]}.
  *
  * <p>This implementation uses a pool of {@link Runtime#availableProcessors()} threads.
  *
  * @param a the first array to be sorted.
  * @param b the second array to be sorted.
  */
 public static void parallelRadixSort(final KEY_TYPE[] a, final KEY_TYPE[] b) {
  ensureSameLength(a, b);
  parallelRadixSort(a, b, 0, a.length);
 }


 private static KEY_GENERIC void insertionSortIndirect(final int[] perm, final KEY_TYPE[] a, final KEY_TYPE[]
b, final int from, final int to) {
  for (int i = from; ++i < to;) {
   int t = perm[i];
   int j = i;
   for (int u = perm[j - 1]; KEY_LESS(a[t], a[u]) || KEY_CMP_EQ(a[t], a[u]) && KEY_LESS(b[t], b[u]); u = perm[--j
- 1]) {
    perm[j] = u;
    if (from == j - 1) {
     --j;
```

```
    break;
   }
  }
  perm[j] = t;
 }
}

 /** Sorts the specified pair of arrays lexicographically using indirect radix sort.
  *
  * <p>The sorting algorithm is a tuned radix sort adapted from Peter M. McIlroy, Keith Bostic and M. Douglas
  * McIlroy, &amp;ldquo;Engineering radix sort&amp;rdquo;, <i>Computing Systems</i>, 6(1), pages 5&amp;minus;27 (1993).
  *
  * <p>This method implement an <em>indirect</em> sort. The elements of {@code perm} (which must
  * be exactly the numbers in the interval {@code [0..perm.length)}) will be permuted so that
  * {@code a[perm[i]] &amp;le; a[perm[i + 1]]} or {@code a[perm[i]] == a[perm[i + 1]]} and {@code b[perm[i]] &amp;le;
b[perm[i + 1]]}.
  *
  * <p>This implementation will allocate, in the stable case, a further support array as large as {@code perm} (note
that the stable
  * version is slightly faster).
  *
  * @param perm a permutation array indexing {@code a}.
  * @param a the array to be sorted.
  * @param b the second array to be sorted.
  * @param stable whether the sorting algorithm should be stable.
  */
 public static void radixSortIndirect(final int[] perm, final KEY_TYPE[] a, final KEY_TYPE[] b, final boolean
stable) {
  ensureSameLength(a, b);
  radixSortIndirect(perm, a, b, 0, a.length, stable);
 }

 /** Sorts the specified pair of arrays lexicographically using indirect radix sort.
  *
  * <p>The sorting algorithm is a tuned radix sort adapted from Peter M. McIlroy, Keith Bostic and M. Douglas
  * McIlroy, &amp;ldquo;Engineering radix sort&amp;rdquo;, <i>Computing Systems</i>, 6(1), pages 5&amp;minus;27 (1993).
  *
  * <p>This method implement an <em>indirect</em> sort. The elements of {@code perm} (which must
  * be exactly the numbers in the interval {@code [0..perm.length)}) will be permuted so that
  * {@code a[perm[i]] &amp;le; a[perm[i + 1]]} or {@code a[perm[i]] == a[perm[i + 1]]} and {@code b[perm[i]] &amp;le;
b[perm[i + 1]]}.
  *
  * <p>This implementation will allocate, in the stable case, a further support array as large as {@code perm} (note
that the stable
  * version is slightly faster).
  *
  * @param perm a permutation array indexing {@code a}.
  * @param a the array to be sorted.
```

```
  * @param b the second array to be sorted.
  * @param from the index of the first element of {@code perm} (inclusive) to be permuted.
  * @param to the index of the last element of {@code perm} (exclusive) to be permuted.
  * @param stable whether the sorting algorithm should be stable.
  */
 public static void radixSortIndirect(final int[] perm, final KEY_TYPE[] a, final KEY_TYPE[] b, final int from, final int to, final boolean stable) {
  if (to - from < RADIXSORT_NO_REC) {
   insertionSortIndirect(perm, a, b, from, to);
   return;
  }

  final int layers = 2;
  final int maxLevel = DIGITS_PER_ELEMENT * layers - 1;

  final int stackSize = ((1 << DIGIT_BITS) - 1) * (layers * DIGITS_PER_ELEMENT - 1) + 1;
  int stackPos = 0;
  final int[] offsetStack = new int[stackSize];
  final int[] lengthStack = new int[stackSize];
  final int[] levelStack = new int[stackSize];

  offsetStack[stackPos] = from;
  lengthStack[stackPos] = to - from;
  levelStack[stackPos++] = 0;

  final int[] count = new int[1 << DIGIT_BITS];
  final int[] pos = new int[1 << DIGIT_BITS];
  final int[] support = stable ? new int[perm.length] : null;

  while(stackPos > 0) {
   final int first = offsetStack[--stackPos];
   final int length = lengthStack[stackPos];
   final int level = levelStack[stackPos];
#if KEY_CLASS_Character
   final int signMask = 0;
#else
   final int signMask = level % DIGITS_PER_ELEMENT == 0 ? 1 << DIGIT_BITS - 1 : 0;
#endif

   final KEY_TYPE[] k = level < DIGITS_PER_ELEMENT ? a : b; // This is the key array
   final int shift = (DIGITS_PER_ELEMENT - 1 - level % DIGITS_PER_ELEMENT) * DIGIT_BITS; // This is the shift that extract the right byte from a key

   // Count keys.
   for(int i = first + length; i-- != first;) count[INT(KEY2LEXINT(k[perm[i]]) >>> shift & DIGIT_MASK ^ signMask)]++;

   // Compute cumulative distribution
```

```
  int lastUsed = -1;
  for (int i = 0, p = stable ? 0 : first; i < 1 << DIGIT_BITS; i++) {
   if (count[i] != 0) lastUsed = i;
   pos[i] = (p += count[i]);
  }

  if (stable) {
   for(int i = first + length; i-- != first;) support[--pos[INT(KEY2LEXINT(k[perm[i]]) >>> shift & DIGIT_MASK ^
signMask)]] = perm[i];
   System.arraycopy(support, 0, perm, first, length);
   for(int i = 0, p = first; i < 1 << DIGIT_BITS; i++) {
    if (level < maxLevel && count[i] > 1) {
     if (count[i] < RADIXSORT_NO_REC) insertionSortIndirect(perm, a, b, p, p + count[i]);
     else {
      offsetStack[stackPos] = p;
      lengthStack[stackPos] = count[i];
      levelStack[stackPos++] = level + 1;
     }
    }
    p += count[i];
   }
   java.util.Arrays.fill(count, 0);
  }
  else {
   final int end = first + length - count[lastUsed];
   // i moves through the start of each block
   for(int i = first, c = -1, d; i <= end; i += count[c], count[c] = 0) {
    int t = perm[i];
    c = INT(KEY2LEXINT(k[t]) >>> shift & DIGIT_MASK ^ signMask);

    if (i < end) { // When all slots are OK, the last slot is necessarily OK.
     while((d = --pos[c]) > i) {
      final int z = t;
      t = perm[d];
      perm[d] = z;
      c = INT(KEY2LEXINT(k[t]) >>> shift & DIGIT_MASK ^ signMask);
     }
     perm[i] = t;
    }


    if (level < maxLevel && count[c] > 1) {
     if (count[c] < RADIXSORT_NO_REC) insertionSortIndirect(perm, a, b, i, i + count[c]);
     else {
      offsetStack[stackPos] = i;
      lengthStack[stackPos] = count[c];
      levelStack[stackPos++] = level + 1;
     }
```

```
    }
   }
  }
 }
}
```

```
private static void selectionSort(final KEY_TYPE[][] a, final int from, final int to, final int level) {
 final int layers = a.length;
 final int firstLayer = level / DIGITS_PER_ELEMENT;

 for(int i = from; i < to - 1; i++) {
  int m = i;
  for(int j = i + 1; j < to; j++) {
   for(int p = firstLayer; p < layers; p++) {
    if (a[p][j] < a[p][m]) {
     m = j;
     break;
    }
    else if (a[p][j] > a[p][m]) break;
   }
  }
  if (m != i) {
   for(int p = layers; p-- != 0;) {
    final KEY_TYPE u = a[p][i];
    a[p][i] = a[p][m];
    a[p][m] = u;
   }
  }
 }
}
```

```
/** Sorts the specified array of arrays lexicographically using radix sort.
 *
 * <p>The sorting algorithm is a tuned radix sort adapted from Peter M. McIlroy, Keith Bostic and M. Douglas
 * McIlroy, &ldquo;Engineering radix sort&rdquo;, <i>Computing Systems</i>, 6(1), pages 5&minus;27 (1993).
 *
 * <p>This method implements a <em>lexicographical</em> sorting of the provided arrays. Tuples of elements
 * in the same position will be considered a single key, and permuted
 * accordingly.
 *
 * @param a an array containing arrays of equal length to be sorted lexicographically in parallel.
 */
public static void radixSort(final KEY_TYPE[][] a) {
```

```
 radixSort(a, 0, a[0].length);
 }

 /** Sorts the specified array of arrays lexicographically using radix sort.
  *
  * <p>The sorting algorithm is a tuned radix sort adapted from Peter M. McIlroy, Keith Bostic and M. Douglas
  * McIlroy, &ldquo;Engineering radix sort&rdquo;, <i>Computing Systems</i>, 6(1), pages 5&minus;27 (1993).
  *
  * <p>This method implements a <em>lexicographical</em> sorting of the provided arrays. Tuples of elements
  * in the same position will be considered a single key, and permuted
  * accordingly.
  *
  * @param a an array containing arrays of equal length to be sorted lexicographically in parallel.
  * @param from the index of the first element (inclusive) to be sorted.
  * @param to the index of the last element (exclusive) to be sorted.
  */
 public static void radixSort(final KEY_TYPE[][] a, final int from, final int to) {
  if (to - from < RADIXSORT_NO_REC) {
   selectionSort(a, from, to, 0);
   return;
  }

  final int layers = a.length;
  final int maxLevel = DIGITS_PER_ELEMENT * layers - 1;
  for(int p = layers, l = a[0].length; p-- != 0;) if (a[p].length != l) throw new IllegalArgumentException("The array of
index " + p + " has not the same length of the array of index 0.");

  final int stackSize = ((1 << DIGIT_BITS) - 1) * (layers * DIGITS_PER_ELEMENT - 1) + 1;
  int stackPos = 0;
  final int[] offsetStack = new int[stackSize];
  final int[] lengthStack = new int[stackSize];
  final int[] levelStack = new int[stackSize];

  offsetStack[stackPos] = from;
  lengthStack[stackPos] = to - from;
  levelStack[stackPos++] = 0;

  final int[] count = new int[1 << DIGIT_BITS];
  final int[] pos = new int[1 << DIGIT_BITS];
  final KEY_TYPE[] t = new KEY_TYPE[layers];

  while(stackPos > 0) {
   final int first = offsetStack[--stackPos];
   final int length = lengthStack[stackPos];
   final int level = levelStack[stackPos];
#if KEY_CLASS_Character
   final int signMask = 0;
#else
```

```
  final int signMask = level % DIGITS_PER_ELEMENT == 0 ? 1 << DIGIT_BITS - 1 : 0;
#endif

  final KEY_TYPE[] k = a[level / DIGITS_PER_ELEMENT]; // This is the key array
  final int shift = (DIGITS_PER_ELEMENT - 1 - level % DIGITS_PER_ELEMENT) * DIGIT_BITS; // This is the shift that extract the right byte from a key

  // Count keys.
  for(int i = first + length; i-- != first;) count[INT(KEY2LEXINT(k[i]) >>> shift & DIGIT_MASK ^ signMask)]++;

  // Compute cumulative distribution
  int lastUsed = -1;
  for (int i = 0, p = first; i < 1 << DIGIT_BITS; i++) {
   if (count[i] != 0) lastUsed = i;
   pos[i] = (p += count[i]);
  }

  final int end = first + length - count[lastUsed];
  // i moves through the start of each block
  for(int i = first, c = -1, d; i <= end; i += count[c], count[c] = 0) {
   for(int p = layers; p-- != 0;) t[p] = a[p][i];
   c = INT(KEY2LEXINT(k[i]) >>> shift & DIGIT_MASK ^ signMask);

   if (i < end) { // When all slots are OK, the last slot is necessarily OK.
    while((d = --pos[c]) > i) {
     c = INT(KEY2LEXINT(k[d]) >>> shift & DIGIT_MASK ^ signMask);
     for(int p = layers; p-- != 0;) {
      final KEY_TYPE u = t[p];
      t[p] = a[p][d];
      a[p][d] = u;
     }
    }
    for(int p = layers; p-- != 0;) a[p][i] = t[p];
   }

   if (level < maxLevel && count[c] > 1) {
    if (count[c] < RADIXSORT_NO_REC) selectionSort(a, i, i + count[c], level + 1);
    else {
     offsetStack[stackPos] = i;
     lengthStack[stackPos] = count[c];
     levelStack[stackPos++] = level + 1;
    }
   }
  }
 }
}
```

```
#endif

#endif

 /** Shuffles the specified array fragment using the specified pseudorandom number generator.
  *
  * @param a the array to be shuffled.
  * @param from the index of the first element (inclusive) to be shuffled.
  * @param to the index of the last element (exclusive) to be shuffled.
  * @param random a pseudorandom number generator.
  * @return {@code a}.
  */
 public static KEY_GENERIC KEY_GENERIC_TYPE[] shuffle(final KEY_GENERIC_TYPE[] a, final int from,
final int to, final Random random) {
  for(int i = to - from; i-- != 0;) {
   final int p = random.nextInt(i + 1);
   final KEY_GENERIC_TYPE t = a[from + i];
   a[from + i] = a[from + p];
   a[from + p] = t;
  }
  return a;
 }

 /** Shuffles the specified array using the specified pseudorandom number generator.
  *
  * @param a the array to be shuffled.
  * @param random a pseudorandom number generator.
  * @return {@code a}.
  */
 public static KEY_GENERIC KEY_GENERIC_TYPE[] shuffle(final KEY_GENERIC_TYPE[] a, final Random
random) {
  for(int i = a.length; i-- != 0;) {
   final int p = random.nextInt(i + 1);
   final KEY_GENERIC_TYPE t = a[i];
   a[i] = a[p];
   a[p] = t;
  }
  return a;
 }

 /** Reverses the order of the elements in the specified array.
  *
  * @param a the array to be reversed.
  * @return {@code a}.
  */
 public static KEY_GENERIC KEY_GENERIC_TYPE[] reverse(final KEY_GENERIC_TYPE[] a) {
  final int length = a.length;
  for(int i = length / 2; i-- != 0;) {
```

```
  final KEY_GENERIC_TYPE t = a[length - i - 1];
  a[length - i - 1] = a[i];
  a[i] = t;
  }
  return a;
 }

 /** Reverses the order of the elements in the specified array fragment.
  *
  * @param a the array to be reversed.
  * @param from the index of the first element (inclusive) to be reversed.
  * @param to the index of the last element (exclusive) to be reversed.
  * @return {@code a}.
  */
 public static KEY_GENERIC KEY_GENERIC_TYPE[] reverse(final KEY_GENERIC_TYPE[] a, final int from,
final int to) {
  final int length = to - from;
  for(int i = length / 2; i-- != 0;) {
   final KEY_GENERIC_TYPE t = a[from + length - i - 1];
   a[from + length - i - 1] = a[from + i];
   a[from + i] = t;
  }
  return a;
 }

 /** A type-specific content-based hash strategy for arrays. */

 private static final class ArrayHashStrategy KEY_GENERIC implements
Hash.Strategy<KEY_GENERIC_TYPE[]>, java.io.Serializable {
  private static final long serialVersionUID = -7046029254386353129L;

  @Override
  public int hashCode(final KEY_GENERIC_TYPE[] o) { return java.util.Arrays.hashCode(o); }

  @Override
  public boolean equals(final KEY_GENERIC_TYPE[] a, final KEY_GENERIC_TYPE[] b) { return
java.util.Arrays.equals(a, b); }
 }

 /** A type-specific content-based hash strategy for arrays.
  *
  * <p>This hash strategy may be used in custom hash collections whenever keys are
  * arrays, and they must be considered equal by content. This strategy
  * will handle {@code null} correctly, and it is serializable.
  */

#if KEYS_PRIMITIVE
 public static final Hash.Strategy<KEY_TYPE[]> HASH_STRATEGY = new ArrayHashStrategy();
```

```
#else
 @SuppressWarnings({"rawtypes"})
 public static final Hash.Strategy HASH_STRATEGY = new ArrayHashStrategy();
#endif

}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/Arrays.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2003-2017 Paolo Boldi and Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

#if KEY_CLASS_Object
import java.util.Comparator;
import it.unimi.dsi.fastutil.IndirectPriorityQueue;
#endif

import it.unimi.dsi.fastutil.ints.IntArrays;

import java.util.NoSuchElementException;

/** A type-specific array-based semi-indirect priority queue.
*
* <p>Instances of this class use as reference list a <em>reference array</em>,
* which must be provided to each constructor, and represent a priority queue
* using a backing array of integer indices&amp;mdash;all operations are performed
* directly on the array. The array is enlarged as needed, but it is never
* shrunk. Use the {@link #trim()} method to reduce its size, if necessary.
*
```

```
* <p>This implementation is extremely inefficient, but it is difficult to beat
* when the size of the queue is very small. Moreover, it allows to enqueue several
* time the same index, without limitations.
*/

public class ARRAY_INDIRECT_PRIORITY_QUEUE KEY_GENERIC implements
INDIRECT_PRIORITY_QUEUE KEY_GENERIC {

 /** The reference array. */
 protected KEY_GENERIC_TYPE refArray[];

 /** The backing array. */
 protected int array[] = IntArrays.EMPTY_ARRAY;

 /** The number of elements in this queue. */
 protected int size;

 /** The type-specific comparator used in this queue. */
 protected KEY_COMPARATOR KEY_SUPER_GENERIC c;

 /** The first index, cached, if {@link #firstIndexValid} is true. */
 protected int firstIndex;

 /** Whether {@link #firstIndex} contains a valid value. */
 protected boolean firstIndexValid;


 /** Creates a new empty queue without elements with a given capacity and comparator.
  *
  * @param refArray the reference array.
  * @param capacity the initial capacity of this queue.
  * @param c the comparator used in this queue, or {@code null} for the natural order.
  */
 public ARRAY_INDIRECT_PRIORITY_QUEUE(KEY_GENERIC_TYPE[] refArray, int capacity,
KEY_COMPARATOR KEY_SUPER_GENERIC c) {
  if (capacity > 0) this.array = new int[capacity];
  this.refArray = refArray;
  this.c = c;
 }

 /** Creates a new empty queue with given capacity and using the natural order.
  *
  * @param refArray the reference array.
  * @param capacity the initial capacity of this queue.
  */
 public ARRAY_INDIRECT_PRIORITY_QUEUE(KEY_GENERIC_TYPE[] refArray, int capacity) {
  this(refArray, capacity, null);
 }
```

```
 /** Creates a new empty queue with capacity equal to the length of the reference array and a given comparator.
  *
  * @param refArray the reference array.
  * @param c the comparator used in this queue, or {@code null} for the natural order.
  */
 public ARRAY_INDIRECT_PRIORITY_QUEUE(KEY_GENERIC_TYPE[] refArray, KEY_COMPARATOR
KEY_SUPER_GENERIC c) {
  this(refArray, refArray.length, c);
 }

 /** Creates a new empty queue with capacity equal to the length of the reference array and using the natural order.
  * @param refArray the reference array.
  */
 public ARRAY_INDIRECT_PRIORITY_QUEUE(KEY_GENERIC_TYPE[] refArray) {
  this(refArray, refArray.length, null);
 }


 /** Wraps a given array in a queue using a given comparator.
  *
  * <p>The queue returned by this method will be backed by the given array.
  *
  * @param refArray the reference array.
  * @param a an array of indices into {@code refArray}.
  * @param size the number of elements to be included in the queue.
  * @param c the comparator used in this queue, or {@code null} for the natural order.
  */
 public ARRAY_INDIRECT_PRIORITY_QUEUE(final KEY_GENERIC_TYPE[] refArray, final int[] a, int size,
final KEY_COMPARATOR KEY_SUPER_GENERIC c) {
  this(refArray, 0, c);
  this.array = a;
  this.size = size;
 }


 /** Wraps a given array in a queue using a given comparator.
  *
  * <p>The queue returned by this method will be backed by the given array.
  *
  * @param refArray the reference array.
  * @param a an array of indices into {@code refArray}.
  * @param c the comparator used in this queue, or {@code null} for the natural order.
  */
 public ARRAY_INDIRECT_PRIORITY_QUEUE(final KEY_GENERIC_TYPE[] refArray, final int[] a, final
KEY_COMPARATOR KEY_SUPER_GENERIC c) {
  this(refArray, a, a.length, c);
 }
```

```
/** Wraps a given array in a queue using the natural order.
 *
 * <p>The queue returned by this method will be backed by the given array.
 *
 * @param refArray the reference array.
 * @param a an array of indices into {@code refArray}.
 * @param size the number of elements to be included in the queue.
 */
public ARRAY_INDIRECT_PRIORITY_QUEUE(final KEY_GENERIC_TYPE[] refArray, final int[] a, int size) {
 this(refArray, a, size, null);
}


/** Wraps a given array in a queue using the natural order.
 *
 * <p>The queue returned by this method will be backed by the given array.
 *
 * @param refArray the reference array.
 * @param a an array of indices into {@code refArray}.
 */
public ARRAY_INDIRECT_PRIORITY_QUEUE(final KEY_GENERIC_TYPE[] refArray, final int[] a) {
 this(refArray, a, a.length);
}


/** Returns the index (in {@link #array}) of the smallest element. */

SUPPRESS_WARNINGS_KEY_UNCHECKED
private int findFirst() {
 if (firstIndexValid) return this.firstIndex;
 firstIndexValid = true;
 int i = size;
 int firstIndex = --i;
 KEY_GENERIC_TYPE first = refArray[array[firstIndex]];

 if (c == null) while(i-- != 0) { if (KEY_LESS(refArray[array[i]], first)) first = refArray[array[firstIndex = i]]; }
 else while(i-- != 0) { if (c.compare(refArray[array[i]], first) < 0) first = refArray[array[firstIndex = i]]; }

 return this.firstIndex = firstIndex;
}

/** Returns the index (in {@link #array}) of the largest element. */

SUPPRESS_WARNINGS_KEY_UNCHECKED
private int findLast() {
 int i = size;
 int lastIndex = --i;
```

```
 KEY_GENERIC_TYPE last = refArray[array[lastIndex]];

 if (c == null) { while(i-- != 0) if (KEY_LESS(last, refArray[array[i]])) last = refArray[array[lastIndex = i]]; }
 else { while(i-- != 0) if (c.compare(last, refArray[array[i]]) < 0) last = refArray[array[lastIndex = i]]; }

 return lastIndex;
}

protected final void ensureNonEmpty() {
 if (size == 0) throw new NoSuchElementException();
}

/** Ensures that the given index is a firstIndexValid reference.
 *
 * @param index an index in the reference array.
 * @throws IndexOutOfBoundsException if the given index is negative or larger than the reference array length.
 */
protected void ensureElement(final int index) {
 if (index < 0)  throw new IndexOutOfBoundsException("Index (" + index + ") is negative");
 if (index >= refArray.length) throw new IndexOutOfBoundsException("Index (" + index + ") is larger than or equal
to reference array size (" + refArray.length + ")");
}

/** {@inheritDoc}
 *
 * <p>Note that for efficiency reasons this method will <em>not</em> throw an exception
 * when {@code x} is already in the queue. However, the queue state will become
 * inconsistent and the following behaviour will not be predictable.
 */
@Override
SUPPRESS_WARNINGS_KEY_UNCHECKED
public void enqueue(int x) {
 ensureElement(x);

 if (size == array.length) array = IntArrays.grow(array, size + 1);
 if (firstIndexValid) {
  if (c == null) { if (KEY_LESS(refArray[x], refArray[array[firstIndex]])) firstIndex = size; }
  else if (c.compare(refArray[x], refArray[array[firstIndex]]) < 0) firstIndex = size;
 }
 else firstIndexValid = false;
 array[size++] = x;
}

@Override
public int dequeue() {
 ensureNonEmpty();
 final int firstIndex = findFirst();
 final int result = array[firstIndex];
```

```
 if (--size != 0) System.arraycopy(array, firstIndex + 1, array, firstIndex, size - firstIndex);
 firstIndexValid = false;
 return result;
}

@Override
public int first() {
 ensureNonEmpty();
 return array[findFirst()];
}

@Override
public int last() {
 ensureNonEmpty();
 return array[findLast()];
}

@Override
public void changed() {
 ensureNonEmpty();
 firstIndexValid = false;
}

/** {@inheritDoc}
 *
 * <p>Note that for efficiency reasons this method will <em>not</em> throw an exception
 * when {@code index} is not in the queue.
 */
@Override
public void changed(int index) {
 ensureElement(index);
 if (index == firstIndex) firstIndexValid = false;
}

/** Signals the queue that all elements have changed. */
@Override
public void allChanged() { firstIndexValid = false; }

@Override
public boolean remove(int index) {
 ensureElement(index);
 final int[] a = array;
 int i = size;
 while(i-- != 0) if (a[i] == index) break;
 if (i < 0) return false;
 firstIndexValid = false;
 if (--size != 0) System.arraycopy(a, i + 1, a, i, size - i);
 return true;
```

```
}

/** Writes in the provided array the <em>front</em> of the queue, that is, the set of indices
 * whose elements have the same priority as the top.
 * @param a an array whose initial part will be filled with the frnot (must be sized as least as the heap size).
 * @return the number of elements of the front.
 */
@Override
public int front(int[] a) {
 final KEY_GENERIC_TYPE top = refArray[array[findFirst()]];
 int i = size, c = 0;
 while(i-- != 0) if (KEY_EQUALS_NOT_NULL(top, refArray[array[i]])) a[c++] = array[i];
 return c;
}

@Override
public int size() { return size; }

@Override
public void clear() { size = 0; firstIndexValid = false; }

/** Trims the backing array so that it has exactly {@link #size()} elements. */
public void trim() {
 array = IntArrays.trim(array, size);
}

@Override
public KEY_COMPARATOR KEY_SUPER_GENERIC comparator() { return c; }

@Override
public String toString() {
 StringBuffer s = new StringBuffer();
 s.append("[");
 for (int i = 0; i < size; i++) {
  if (i != 0) s.append(", ");
  s.append(refArray[array [i]]);
 }
 s.append("]");
 return s.toString();
}


#ifdef TEST
 private static long seed = System.currentTimeMillis();
 private static java.util.Random r = new java.util.Random(seed);

 private static KEY_TYPE genKey() {
#if KEY_CLASS_Byte || KEY_CLASS_Short || KEY_CLASS_Character
```

```
  return (KEY_TYPE)(r.nextInt());
#elif KEYS_PRIMITIVE
  return r.NEXT_KEY();
#elif KEY_CLASS_Object
  return Integer.toBinaryString(r.nextInt());
#else
  return new java.io.Serializable() {};
#endif
 }

 private static java.text.NumberFormat format = new java.text.DecimalFormat("#,###.00");
 private static java.text.FieldPosition p = new java.text.FieldPosition(0);

 private static String format(double d) {
  StringBuffer s = new StringBuffer();
  return format.format(d, s, p).toString();
 }

 private static void speedTest(int n, boolean comp) {
  int i, j, s;
  ARRAY_INDIRECT_PRIORITY_QUEUE[] m = new ARRAY_INDIRECT_PRIORITY_QUEUE[100000];
  HEAP_INDIRECT_PRIORITY_QUEUE[] t = new HEAP_INDIRECT_PRIORITY_QUEUE[m.length];
  KEY_TYPE k[] = new KEY_TYPE[n];
  KEY_TYPE nk[] = new KEY_TYPE[m.length];
  long ms;

  for(i = 0; i < n; i++) k[i] = genKey();
  for(i = 0; i < m.length; i++) nk[i] = genKey();

  double totEnq = 0, totDeq = 0, totChange = 0, d;

  for(i = 0; i < m.length; i++) {
   t[i] = new HEAP_INDIRECT_PRIORITY_QUEUE(k);
   m[i] = new ARRAY_INDIRECT_PRIORITY_QUEUE(k);
  }

  if (comp) {
   for(j = 0; j < 20; j++) {

    for(i = 0; i < m.length; i++) t[i].clear();

    ms = System.currentTimeMillis();
    s = m.length;
    while(s-- != 0) { i = n; while(i-- != 0) t[s].enqueue(i); }
    d = System.currentTimeMillis() - ms;
    if (j > 2) totEnq += d;
    System.out.print("Enqueue: " + format(m.length * n/d) +" K/s ");
```

```
  ms = System.currentTimeMillis();
  s = m.length;
  while(s-- != 0) { i = n; while(i-- != 0) { k[t[s].first()] = nk[i]; t[s].changed(); } }
  d = System.currentTimeMillis() - ms;
  if (j > 2) totChange += d;
  System.out.print("Change: " + format(m.length * n/d) +" K/s ");

  ms = System.currentTimeMillis();
  s = m.length;
  while(s-- != 0) { i = n; while(i-- != 0) t[s].dequeue(); }
  d = System.currentTimeMillis() - ms;
  if (j > 2) totDeq += d;
  System.out.print("Dequeue: " + format(m.length * n/d) +" K/s ");

  System.out.println();
  }

  System.out.println();
  System.out.println("Heap:  Enqueue: " + format(m.length * (j-3)*n/totEnq) + " K/s Dequeue: " + format(m.length *
(j-3)*n/totDeq)  + " K/s Change: " + format(m.length * (j-3)*n/totChange) + " K/s");

  System.out.println();

  totEnq = totChange = totDeq = 0;

  }

  for(j = 0; j < 20; j++) {

  for(i = 0; i < m.length; i++) m[i].clear();

  ms = System.currentTimeMillis();
  s = m.length;
  while(s-- != 0) { i = n; while(i-- != 0) m[s].enqueue(i); }
  d = System.currentTimeMillis() - ms;
  if (j > 2) totEnq += d;
  System.out.print("Enqueue: " + format(m.length * n/d) +" K/s ");

  ms = System.currentTimeMillis();
  s = m.length;
  while(s-- != 0) { i = n; while(i-- != 0) { k[m[s].first()] = nk[i]; m[s].changed(); } }
  d = System.currentTimeMillis() - ms;
  if (j > 2) totChange += d;
  System.out.print("Change: " + format(m.length * n/d) +" K/s ");

  ms = System.currentTimeMillis();
  s = m.length;
  while(s-- != 0) { i = n; while(i-- != 0) m[s].dequeue(); }
```

```
  d = System.currentTimeMillis() - ms;
  if (j > 2) totDeq += d;
  System.out.print("Dequeue: " + format(m.length * n/d) +" K/s ");

  System.out.println();
  }


 System.out.println();
 System.out.println("Array:  Enqueue: " + format(m.length * (j-3)*n/totEnq) + " K/s Dequeue: " + format(m.length *
(j-3)*n/totDeq)  + " K/s Change: " + format(m.length * (j-3)*n/totChange) + " K/s");

 System.out.println();
 }


 private static void fatal(String msg) {
  System.out.println(msg);
  System.exit(1);
 }

 private static void ensure(boolean cond, String msg) {
  if (cond) return;
  fatal(msg);
 }

 private static boolean heapEqual(int[] a, int[] b, int sizea, int sizeb) {
  if (sizea != sizeb) return false;
  KEY_TYPE[] aa = new KEY_TYPE[sizea];
  KEY_TYPE[] bb = new KEY_TYPE[sizea];
  for(int i = 0; i < sizea; i++) {
   aa[i] = ref[a[i]];
   bb[i] = ref[b[i]];
  }
  java.util.Arrays.sort(aa);
  java.util.Arrays.sort(bb);
  while(sizea-- != 0) if (!KEY_EQUALS(aa[sizea], bb[sizea])) return false;
  return true;
 }

 private static KEY_TYPE[] ref;

 protected static void runTest(int n) {
  long ms;
  Exception mThrowsIllegal, tThrowsIllegal, mThrowsOutOfBounds, tThrowsOutOfBounds, mThrowsNoElement,
tThrowsNoElement;
  int rm = 0, rt = 0;
```

```
  ref = new KEY_TYPE[n];

  for(int i = 0; i < n; i++) ref[i] = genKey();

  ARRAY_INDIRECT_PRIORITY_QUEUE m = new ARRAY_INDIRECT_PRIORITY_QUEUE(ref);
  HEAP_INDIRECT_PRIORITY_QUEUE t = new HEAP_INDIRECT_PRIORITY_QUEUE(ref);

  /* We add pairs to t. */
  for(int i = 0; i < n / 2;  i++) {
   t.enqueue(i);
   m.enqueue(i);
  }

  ensure(heapEqual(m.array, t.heap, m.size(), t.size()), "Error (" + seed + "): m and t differ after creation (" + m + ", "
+ t + ")");

  /* Now we add and remove random data in m and t, checking that the result is the same. */

  for(int i=0; i<2*n;  i++) {
   if (r.nextDouble() < 0.01) {
    t.clear();
    m.clear();
    for(int j = 0; j < n / 2;  j++) {
     t.enqueue(j);
     m.enqueue(j);
    }
   }

   int T = r.nextInt(2 * n);

   mThrowsNoElement = tThrowsNoElement = mThrowsOutOfBounds = tThrowsOutOfBounds = mThrowsIllegal =
tThrowsIllegal = null;

   try {
    t.enqueue(T);
   }
   catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }
   catch (IllegalArgumentException e) { tThrowsIllegal = e; }

   if (tThrowsIllegal == null) { // To skip duplicates
    try {
     m.enqueue(T);
    }
    catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }
    catch (IllegalArgumentException e) { mThrowsIllegal = e; }
   }

   mThrowsIllegal = tThrowsIllegal = null; // To skip duplicates
```

```
  ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + seed + "): enqueue() divergence in IndexOutOfBoundsException for " + T + " (" + mThrowsOutOfBounds + ", " + tThrowsOutOfBounds + ")");
  ensure((mThrowsIllegal == null) == (tThrowsIllegal == null), "Error (" + seed + "): enqueue() divergence in IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");

  ensure(heapEqual(m.array, t.heap, m.size(), t.size()), "Error (" + seed + "): m and t differ after enqueue (" + m + ", " + t + ")");

  if (m.size() != 0) {
   ensure(KEY_EQUALS(ref[m.first()], ref[t.first()]), "Error (" + seed + "): m and t differ in first element after enqueue (" + m.first() + "->" + ref[m.first()] + ", " + t.first() + "->" + ref[t.first()] + ")");
  }

  mThrowsNoElement = tThrowsNoElement = mThrowsOutOfBounds = tThrowsOutOfBounds = mThrowsIllegal = tThrowsIllegal = null;

  try {
   rm = m.dequeue();
   while(! m.isEmpty() && KEY_EQUALS(ref[m.first()], ref[rm])) m.dequeue();
  }
  catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }
  catch (IllegalArgumentException e) { mThrowsIllegal = e; }
  catch (java.util.NoSuchElementException e) { mThrowsNoElement = e; }

  try {
   rt = t.dequeue();
   while(! t.isEmpty() && KEY_EQUALS(ref[t.first()], ref[rt])) t.dequeue();
  }
  catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }
  catch (IllegalArgumentException e) { tThrowsIllegal = e; }
  catch (java.util.NoSuchElementException e) { tThrowsNoElement = e; }

  ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + seed + "): dequeue() divergence in IndexOutOfBoundsException (" + mThrowsOutOfBounds + ", " + tThrowsOutOfBounds + ")");
  ensure((mThrowsIllegal == null) == (tThrowsIllegal == null), "Error (" + seed + "): dequeue() divergence in IllegalArgumentException  (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
  ensure((mThrowsNoElement == null) == (tThrowsNoElement == null), "Error (" + seed + "): dequeue() divergence in java.util.NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
  if (mThrowsOutOfBounds == null) ensure(KEY_EQUALS(ref[rt], ref[rm]), "Error (" + seed + "): divergence in dequeue() between m and t (" + rm + "->" + ref[rm] + ", " + rt + "->" + ref[rt] + ")");


  ensure(heapEqual(m.array, t.heap, m.size(), t.size()), "Error (" + seed + "): m and t differ after dequeue (" + m + ", " + t + ")");

  if (m.size() != 0) {
```

```
    ensure(KEY_EQUALS(ref[m.first()], ref[t.first()]), "Error (" + seed + "): m and t differ in first element after dequeue (" + m.first() + "->" + ref[m.first()] + ", " + t.first() + "->" + ref[t.first()] + ")");
   }

   mThrowsNoElement = tThrowsNoElement = mThrowsOutOfBounds = tThrowsOutOfBounds = mThrowsIllegal = tThrowsIllegal = null;


   int pos = r.nextInt(n * 2);

   try {
    m.remove(pos);
   }
   catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }
   catch (IllegalArgumentException e) { mThrowsIllegal = e; }
   catch (java.util.NoSuchElementException e) { mThrowsNoElement = e; }

   try {
    t.remove(pos);
   }
   catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }
   catch (IllegalArgumentException e) { tThrowsIllegal = e; }
   catch (java.util.NoSuchElementException e) { tThrowsNoElement = e; }

   ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + seed + "): remove(int) divergence in IndexOutOfBoundsException (" + mThrowsOutOfBounds + ", " + tThrowsOutOfBounds + ")");
   ensure((mThrowsIllegal == null) == (tThrowsIllegal == null), "Error (" + seed + "): remove(int) divergence in IllegalArgumentException  (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
   ensure((mThrowsNoElement == null) == (tThrowsNoElement == null), "Error (" + seed + "): remove(int) divergence in java.util.NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");

   ensure(heapEqual(m.array, t.heap, m.size(), t.size()), "Error (" + seed + "): m and t differ after remove(int) (" + m + ", " + t + ")");

   if (m.size() != 0) {
    ensure(KEY_EQUALS(ref[m.first()], ref[t.first()]), "Error (" + seed + "): m and t differ in first element after remove(int) (" + m.first() + "->" + ref[m.first()] + ", " + t.first() + "->" + ref[t.first()] + ")");
   }


   mThrowsNoElement = tThrowsNoElement = mThrowsOutOfBounds = tThrowsOutOfBounds = mThrowsIllegal = tThrowsIllegal = null;

   pos = r.nextInt(n);

   try {
    t.changed(pos);
   }
```

```
  catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }
  catch (IllegalArgumentException e) { tThrowsIllegal = e; }
  catch (java.util.NoSuchElementException e) { tThrowsNoElement = e; }

  if (tThrowsIllegal == null) {
   try {
    m.changed(pos);
   }
   catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }
   catch (IllegalArgumentException e) { mThrowsIllegal = e; }
   catch (java.util.NoSuchElementException e) { mThrowsNoElement = e; }
  }

  ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + seed + "): change(int)
divergence in IndexOutOfBoundsException (" + mThrowsOutOfBounds + ", " + tThrowsOutOfBounds + ")");
  //ensure((mThrowsIllegal == null) == (tThrowsIllegal == null), "Error (" + seed + "): change(int) divergence in
IllegalArgumentException  (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
  ensure((mThrowsNoElement == null) == (tThrowsNoElement == null), "Error (" + seed + "): change(int)
divergence in java.util.NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");

  ensure(heapEqual(m.array, t.heap, m.size(), t.size()), "Error (" + seed + "): m and t differ after change(int) (" + m +
", " + t + ")");

  if (m.size() != 0) {
   ensure(KEY_EQUALS(ref[m.first()], ref[t.first()]), "Error (" + seed + "): m and t differ in first element after
change(int) (" + m.first() + "->" + ref[m.first()] + ", " + t.first() + "->" + ref[t.first()] + ")");
  }

  int[] temp = (int[])t.heap.clone();
  java.util.Arrays.sort(temp, 0, t.size()); // To scramble a bit
  m = new ARRAY_INDIRECT_PRIORITY_QUEUE(m.refArray, temp, t.size());

  ensure(heapEqual(m.array, t.heap, m.size(), t.size()), "Error (" + seed + "): m and t differ after wrap (" + m + ", " + t
+ ")");

  if (m.size() != 0) {
   ensure(KEY_EQUALS(ref[m.first()], ref[t.first()]), "Error (" + seed + "): m and t differ in first element after wrap ("
+ m.first() + "->" + ref[m.first()] + ", " + t.first() + "->" + ref[t.first()] + ")");
  }

  if (m.size() != 0 && ((new it.unimi.dsi.fastutil.ints.IntOpenHashSet(m.array, 0, m.size())).size() == m.size())) {

   int first = m.first();
   ref[first] = genKey();

   //System.err.println("Pre-change m: " +m);
   //System.err.println("Pre-change t: " +t);
   m.changed();
```

```
    t.changed(first);

    //System.err.println("Post-change m: " +m);
    //System.err.println("Post-change t: " +t);

    ensure(heapEqual(m.array, t.heap, m.size(), t.size()), "Error (" + seed + "): m and t differ after change (" + m + ", " +
t + ")");

    if (m.size() != 0) {
     ensure(KEY_EQUALS(ref[m.first()], ref[t.first()]), "Error (" + seed + "): m and t differ in first element after change
(" + m.first() + "->" + ref[m.first()] + ", " + t.first() + "->" + ref[t.first()] + ")");
    }
   }
  }


  /* Now we check that m actually holds the same data. */

  m.clear();
  ensure(m.isEmpty(), "Error (" + seed + "): m is not empty after clear()");

  System.out.println("Test OK");
 }



 public static void main(String args[]) {
  int n  = Integer.parseInt(args[1]);
  if (args.length > 2) r = new java.util.Random(seed = Long.parseLong(args[2]));


  try {
   if ("speedTest".equals(args[0]) || "speedComp".equals(args[0])) speedTest(n, "speedComp".equals(args[0]));
   else if ("test".equals(args[0])) runTest(n);
  } catch(Throwable e) {
   e.printStackTrace(System.err);
   System.err.println("seed: " + seed);
  }
 }

#endif
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/ArrayIndirectPriorityQueue.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2010-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*
* For the sorting code:
*
* Copyright (C) 1999 CERN - European Organization for Nuclear Research.
*
*   Permission to use, copy, modify, distribute and sell this software and
*   its documentation for any purpose is hereby granted without fee,
*   provided that the above copyright notice appear in all copies and that
*   both that copyright notice and this permission notice appear in
*   supporting documentation. CERN makes no representations about the
*   suitability of this software for any purpose. It is provided "as is"
*   without expressed or implied warranty.
*/
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/BigArrays.java
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
```

```
package PACKAGE;

import VALUE_PACKAGE.VALUE_COLLECTION;
import VALUE_PACKAGE.VALUE_ABSTRACT_COLLECTION;
import VALUE_PACKAGE.VALUE_ITERATOR;

#if KEYS_PRIMITIVE && VALUES_PRIMITIVE
import it.unimi.dsi.fastutil.objects.ObjectIterator;
#endif
import it.unimi.dsi.fastutil.objects.AbstractObjectSet;

import java.util.Iterator;
import java.util.Map;

/** An abstract class providing basic methods for maps implementing a type-specific interface.
*
* <p>Optional operations just throw an {@link
* UnsupportedOperationException}. Generic versions of accessors delegate to
* the corresponding type-specific counterparts following the interface rules
* (they take care of returning {@code null} on a missing key).
*
* <p>As a further help, this class provides a {@link BasicEntry BasicEntry} inner class
* that implements a type-specific version of {@link java.util.Map.Entry}; it
* is particularly useful for those classes that do not implement their own
* entries (e.g., most immutable maps).
*/

public abstract class ABSTRACT_MAP KEY_VALUE_GENERIC extends ABSTRACT_FUNCTION
KEY_VALUE_GENERIC implements MAP KEY_VALUE_GENERIC, java.io.Serializable {

 private static final long serialVersionUID = -4940583368468432370L;

 protected ABSTRACT_MAP() {}

 @Override
 public boolean containsValue(final VALUE_TYPE v) {
  return values().contains(v);
 }

 @Override
 public boolean containsKey(final KEY_TYPE k) {
  final ObjectIterator<MAP.Entry KEY_VALUE_GENERIC> i = ENTRYSET().iterator();
  while(i.hasNext())
   if (i.next().ENTRY_GET_KEY() == k)
    return true;
```

```
 return false;
}

@Override
public boolean isEmpty() {
 return size() == 0;
}

/** This class provides a basic but complete type-specific entry class for all those maps implementations
 * that do not have entries on their own (e.g., most immutable maps).
 *
 * <p>This class does not implement {@link java.util.Map.Entry#setValue(Object) setValue()}, as the modification
 * would not be reflected in the base map.
 */

public static class BasicEntry KEY_VALUE_GENERIC implements MAP.Entry KEY_VALUE_GENERIC {
 protected KEY_GENERIC_TYPE key;
 protected VALUE_GENERIC_TYPE value;

 public BasicEntry() {}

 public BasicEntry(final KEY_GENERIC_CLASS key, final VALUE_GENERIC_CLASS value) {
  this.key = KEY_CLASS2TYPE(key);
  this.value = VALUE_CLASS2TYPE(value);
 }

#if KEYS_PRIMITIVE || VALUES_PRIMITIVE

 public BasicEntry(final KEY_GENERIC_TYPE key, final VALUE_GENERIC_TYPE value) {
  this.key = key;
  this.value = value;
 }

#endif

 @Override
 public KEY_GENERIC_TYPE ENTRY_GET_KEY() {
  return key;
 }

 @Override
 public VALUE_GENERIC_TYPE ENTRY_GET_VALUE() {
  return value;
 }

 @Override
 public VALUE_GENERIC_TYPE setValue(final VALUE_GENERIC_TYPE value) {
  throw new UnsupportedOperationException();
```

```
  }

  SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
  @Override
  public boolean equals(final Object o) {
   if (!(o instanceof Map.Entry)) return false;
   if (o instanceof MAP.Entry) {
    final MAP.Entry KEY_VALUE_GENERIC e = (MAP.Entry KEY_VALUE_GENERIC) o;
    return KEY_EQUALS(key, e.ENTRY_GET_KEY()) && VALUE_EQUALS(value, e.ENTRY_GET_VALUE());
   }

   final Map.Entry<?,?> e = (Map.Entry<?,?>)o;
   final Object key = e.getKey();
#if KEYS_PRIMITIVE
   if (key == null || !(key instanceof KEY_CLASS)) return false;
#endif
   final Object value = e.getValue();
#if VALUES_PRIMITIVE
   if (value == null || !(value instanceof VALUE_CLASS)) return false;
#endif
   return KEY_EQUALS(this.key, KEY_OBJ2TYPE(key)) && VALUE_EQUALS(this.value,
VALUE_OBJ2TYPE(value));
  }

  @Override
  public int hashCode() {
   return KEY2JAVAHASH(key) ^ VALUE2JAVAHASH(value);
  }

  @Override
  public String toString() {
   return key + "->" + value;
  }
 }

 /** This class provides a basic implementation for an Entry set which forwards some queries to the map.
  */

 public abstract static class BasicEntrySet KEY_VALUE_GENERIC extends AbstractObjectSet<Entry
KEY_VALUE_GENERIC> {
  protected final MAP KEY_VALUE_GENERIC map;

  public BasicEntrySet(final MAP KEY_VALUE_GENERIC map) {
   this.map = map;
  }

  SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
  @Override
```

```
  public boolean contains(final Object o) {
   if (!(o instanceof Map.Entry)) return false;

   if (o instanceof MAP.Entry) {
    final MAP.Entry KEY_VALUE_GENERIC e = (MAP.Entry KEY_VALUE_GENERIC) o;
    final KEY_GENERIC_TYPE k = e.ENTRY_GET_KEY();
    return map.containsKey(k) && VALUE_EQUALS(map.GET_VALUE(k), e.ENTRY_GET_VALUE());
   }

   final Map.Entry<?, ?> e = (Map.Entry<?, ?>) o;

#if KEYS_PRIMITIVE
   final Object key = e.getKey();
   if (key == null || !(key instanceof KEY_GENERIC_CLASS)) return false;
   final KEY_TYPE k = KEY_OBJ2TYPE(key);
#else
   final Object k = e.getKey();
#endif
   final Object value = e.getValue();
#if VALUES_PRIMITIVE
   if (value == null || !(value instanceof VALUE_GENERIC_CLASS)) return false;
#endif

   return map.containsKey(k) && VALUE_EQUALS(map.GET_VALUE(k), VALUE_OBJ2TYPE(value));
  }

  SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
  @Override
  public boolean remove(final Object o) {
   if (!(o instanceof Map.Entry)) return false;

   if (o instanceof MAP.Entry) {
    final MAP.Entry KEY_VALUE_GENERIC e = (MAP.Entry KEY_VALUE_GENERIC) o;
    return map.remove(e.ENTRY_GET_KEY(), e.ENTRY_GET_VALUE());
   }

   Map.Entry<?, ?> e = (Map.Entry<?, ?>) o;

#if KEYS_PRIMITIVE
   final Object key = e.getKey();
   if (key == null || !(key instanceof KEY_GENERIC_CLASS)) return false;
   final KEY_TYPE k = KEY_OBJ2TYPE(key);
#else
   final Object k = e.getKey();
#endif
#if VALUES_PRIMITIVE
   final Object value = e.getValue();
   if (value == null || !(value instanceof VALUE_GENERIC_CLASS)) return false;
```

```
   final VALUE_TYPE v = VALUE_OBJ2TYPE(value);
#else
   final Object v = e.getValue();
#endif

   return map.remove(k, v);
  }

  @Override
  public int size() {
   return map.size();
  }
 }


 /** Returns a type-specific-set view of the keys of this map.
  *
  * <p>The view is backed by the set returned by {@link Map#entrySet()}. Note that
  * <em>no attempt is made at caching the result of this method</em>, as this would
  * require adding some attributes that lightweight implementations would
  * not need. Subclasses may easily override this policy by calling
  * this method and caching the result, but implementors are encouraged to
  * write more efficient ad-hoc implementations.
  *
  * @return a set view of the keys of this map; it may be safely cast to a type-specific interface.
  */
 @Override
 public SET KEY_GENERIC keySet() {
  return new ABSTRACT_SET KEY_GENERIC() {
    @Override
    public boolean contains(final KEY_TYPE k) { return containsKey(k); }
    @Override
    public int size() { return ABSTRACT_MAP.this.size(); }
    @Override
    public void clear() { ABSTRACT_MAP.this.clear(); }
    @Override
    public KEY_ITERATOR KEY_GENERIC iterator() {
     return new KEY_ITERATOR KEY_GENERIC() {
       private final ObjectIterator<MAP.Entry KEY_VALUE_GENERIC> i =
MAPS.fastIterator(ABSTRACT_MAP.this);
       @Override
       public KEY_GENERIC_TYPE NEXT_KEY() { return i.next().ENTRY_GET_KEY(); };
       @Override
       public boolean hasNext() { return i.hasNext(); }
       @Override
       public void remove() { i.remove(); }
      };
    }
```

```
  };
 }

 /** Returns a type-specific-set view of the values of this map.
  *
  * <p>The view is backed by the set returned by {@link Map#entrySet()}. Note that
  * <em>no attempt is made at caching the result of this method</em>, as this would
  * require adding some attributes that lightweight implementations would
  * not need. Subclasses may easily override this policy by calling
  * this method and caching the result, but implementors are encouraged to
  * write more efficient ad-hoc implementations.
  *
  * @return a set view of the values of this map; it may be safely cast to a type-specific interface.
  */
 @Override
 public VALUE_COLLECTION VALUE_GENERIC values() {
  return new VALUE_ABSTRACT_COLLECTION VALUE_GENERIC() {
   @Override
   public boolean contains(final VALUE_TYPE k) { return containsValue(k); }
   @Override
   public int size() { return ABSTRACT_MAP.this.size(); }
   @Override
   public void clear() { ABSTRACT_MAP.this.clear(); }

   @Override
   public VALUE_ITERATOR VALUE_GENERIC iterator() {
    return new VALUE_ITERATOR VALUE_GENERIC() {
     private final ObjectIterator<MAP.Entry KEY_VALUE_GENERIC> i =
MAPS.fastIterator(ABSTRACT_MAP.this);
     @Override
     public VALUE_GENERIC_TYPE NEXT_VALUE() { return i.next().ENTRY_GET_VALUE(); }
     @Override
     public boolean hasNext() { return i.hasNext(); }
    };
   }
  };
 }

 /** {@inheritDoc} */
 @SuppressWarnings({"unchecked","deprecation"})
 @Override
 public void putAll(final Map<? extends KEY_GENERIC_CLASS,? extends VALUE_GENERIC_CLASS> m) {
  if (m instanceof MAP) {
   ObjectIterator<MAP.Entry KEY_VALUE_GENERIC> i = MAPS.fastIterator((MAP KEY_VALUE_GENERIC)
m);

   while (i.hasNext()) {
    final MAP.Entry KEY_VALUE_EXTENDS_GENERIC e = i.next();
```

```
   put(e.ENTRY_GET_KEY(), e.ENTRY_GET_VALUE());
  }
 } else {
  int n = m.size();
  final Iterator<? extends Map.Entry<? extends KEY_GENERIC_CLASS,? extends VALUE_GENERIC_CLASS>> i
= m.entrySet().iterator();
  Map.Entry<? extends KEY_GENERIC_CLASS,? extends VALUE_GENERIC_CLASS> e;
  while (n-- != 0) {
   e = i.next();
   put(e.getKey(), e.getValue());
  }
 }
}

/** Returns a hash code for this map.
 *
 * The hash code of a map is computed by summing the hash codes of its entries.
 *
 * @return a hash code for this map.
 */
@Override
public int hashCode() {
 int h = 0, n = size();
 final ObjectIterator<MAP.Entry KEY_VALUE_GENERIC> i = MAPS.fastIterator(this);

 while(n-- != 0) h += i.next().hashCode();
 return h;
}

@Override
public boolean equals(Object o) {
 if (o == this) return true;
 if (! (o instanceof Map)) return false;

 final Map<?,?> m = (Map<?,?>)o;
 if (m.size() != size()) return false;
 return ENTRYSET().containsAll(m.entrySet());
}

@Override
public String toString() {
 final StringBuilder s = new StringBuilder();
 final ObjectIterator<MAP.Entry KEY_VALUE_GENERIC> i = MAPS.fastIterator(this);
 int n = size();
 MAP.Entry KEY_VALUE_GENERIC e;
 boolean first = true;

 s.append("{");
```

```
  while(n-- != 0) {
   if (first) first = false;
   else s.append(", ");

   e = i.next();

#if KEYS_REFERENCE
   if (this == e.getKey()) s.append("(this map)"); else
#endif
    s.append(String.valueOf(e.ENTRY_GET_KEY()));
   s.append("=>");
#if VALUES_REFERENCE
   if (this == e.getValue()) s.append("(this map)"); else
#endif
    s.append(String.valueOf(e.ENTRY_GET_VALUE()));
  }

  s.append("}");
  return s.toString();
 }
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/AbstractMap.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import it.unimi.dsi.fastutil.Hash;
```

```
/** A type-specific {@link Hash} interface.
*
* @see Hash
*/

public interface HASH {

 /** A type-specific hash strategy.
  *
  * @see it.unimi.dsi.fastutil.Hash.Strategy
  */

 interface Strategy {

  /** Returns the hash code of the specified element with respect to this hash strategy.
   *
   * @param e an element.
   * @return the hash code of the given element with respect to this hash strategy.
   */

  int hashCode(KEY_TYPE e);

  /** Returns true if the given elements are equal with respect to this hash strategy.
   *
   * @param a an element.
   * @param b another element.
   * @return true if the two specified elements are equal with respect to this hash strategy.
   */

  boolean equals(KEY_TYPE a, KEY_TYPE b);
 }
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/Hash.drv
No license file was found, but licenses were detected in source scan.

8.2.3

- Array-based lists were not allocating a backing array even if the
 required capacity was greater than the default capacity, violating
 the contract. Thanks to for reporting this bug.

- FastMultiByteArrayInputStream had since 2014 the wrong slice size
 (1Ki instead of 1Gi). Thanks to Thibault Allançon for his detailed
 reports, which dug out this old one.

8.2.2

- Fixed small bug in new lazy allocation scheme for array-based lists.
Resizing would throw an exception in certain circumstances.

- Now strategies must accept a supertype of the base type. Thanks to
shevek@github.com for suggesting this change.

- The Maven source jar does not contain tests anymore.

8.2.1

- Added default modularization to OSGi version, too.

8.2.0

- Added default modularization. Thanks to Joshua Popoff for taking care of
this issue.

- Implemented lazy allocation of arrays in array-based lists. Thanks to
Zheka Kozlov for suggesting this feature.

- Fixed a long-standing issue: the allocation of big lists based on big
arrays could not go beyond 2^31 because of a cut-and-paste bug.

- In line with the JDK, the default initial capacity for lists is 10 and
backing arrays (and, in generally, arrays) are automatically enlarged by
a 50% factor instead of a 100% factor.

- Included dependency-finding scripts by Tobias Meggendorfer.

8.1.1

- More default methods.

- Fixed lack of proper forEach()/fastForEach() method for entry sets of
linked maps (the order of iteration would be random). Thanks to Søren
Gjesse for reporting this bug.

- Fixed lack of proper licensing in test sources.

- A new NO_SMALL_TYPES makefile variable makes it possible to compile
a version of fastutil for ints, longs and doubles only. Thanks to
Tobias Meggendorfer for implementing this feature.

8.1.0

- WARNING: backward-incompatible name change for a few "compute"
methods.

- Implemented efficient new map methods and new iterable methods
in hash-based maps.

- A number of minor glitches has been fixed.

- FastIterable now has a fastForEach() method.

- Fixed ancient bug in the remove() method of entry sets of
tree-based maps.

- Array-based maps have a working key set and value collection.

8.0.0

- First release for Java 8 only, with implementation of new default
methods for iterators and maps. Thanks to Tobias Meggendorfer and
Salomon Sickert from the Technical University of Munich for help in the
implementation. Structure-specific, more efficient code will be
added in the near future. All wrapper (synchronized, etc.) have
been updated.

- Abstract type-specific comparators are deprecated: their only abstract
method has been pulled up as a default method of the type-specific
interface, which makes it possible to use lambda expressions to define
type-specific comparators.

- The default return value setter/getter are now optional for functions.
In this way, functions can now be defined using lambda expressions (and
they implement java.util.function.Function).

- New static methods in type-specific Maps and SortedMaps make it possible
to use easily fast iterators, even within for loops. Thanks to Tobias
Meggendorfer and Salomon Sickert from the Technical University of Munich
for implementing this feature.

- All hash-based structure now keep track of the size of their backing
array at creation time, and will never rehash below that size. Thanks
to Patrick Julien for suggesting this feature.

- Every deprecated method marked to be removed, or replaced by another
method, has been eliminated.

- Many abstract classes are now obsolete as their abstract methods have
become default methods.

- The type hierarchy of big-list iterators has been fixed.

7.2.0

- Major code cleanup. Several unnecessary method implementations have been deleted. Several missing methods (e.g., equals()/hashCode() in wrappers) have been implemented. All methods should now have a reasonable Javadoc comment.

- Clarified the rem()/remove() conundrum: type-specific collections should use and override rem(), but type-specific sets should use and override remove().

- Fixed circular implementation of containsKey() in type-specific abstract collections.

- Pulled up deprecation at the highest possible (interface) level.

- Collection.*All() methods do not assume anymore that they can rely on the argument size().

7.1.1

- Fixed decade-old efficiency bug: implementation of RandomAccess was hidden from synchronized and unmodifiable wrappers. Thanks to Peter Burka for reporting this bug.

- The defaultReturnValue() getter in unmodifiable maps was throwing an UnsupportedOperationException instead of delegating the call to the underlying map. Thanks to Alex Fiennes for reporting this bug.

7.1.0

- Fixed decade-old efficiency bug. Due to a name clash between lists and sets, the type-specific deletion method of a type-specific collection is rem(), not remove(). The latter is reinstated in sets (but not, for example, in lists) by the type-specific abstract set classes. Nonetheless, implementors of subclasses must override rem(), not remove(), as methods such as the type-specific version of removeAll() invoke necessarily invoke rem() rather than remove(). Up to this version, all concrete set implementations were overriding remove(), instead, causing inefficiencies in the inherited methods. Thanks to Christian Habermehl for reporting this bug.

- Fixed a bug introduced with the removal of old-style gcc assertions: all load methods in BinIO that did not specify the number of elements to read were computing the number of items in the loaded file incorrectly, causing an EOFException (except for booleans and bytes).

7.0.13

- Fixed inheritance problem that would surface as key sets of
maps not implementing remove(). Thanks to Luke Nezda for
reporting this bug.

7.0.12

- Collection.isEmpty() was checking for iterator().hasNext() instead
of the opposite. Thanks to Olaf Krische for reporting this bug.

- Fixed lack of test for null/wrong class when testing entries.

7.0.11

- Several small glitches that were making fastutil's classes behave
differently from those of java.util have been fixed. Thanks to Balázs
Attila-Mihály or reporting these bug (obtained by massive testing using
Guava's battery of unit tests).

7.0.10

- The infinite-loop bug was affecting trim(int) besides trim(). Thanks
to Igor Kabiljo for reporting this bug.

- With the help of Erich Schubert, all methods with a type-specific,
more efficient counterpart have been deprecated.

7.0.9

- A subtle infinite-loop bug in hash-based structures (happening with load
factor 1 and tiny structures) has been fixed. Thanks to Tuomas Välimäki
and Jarkko Mönkkönen for reporting this bug.

- Now tree-based map have an addTo() method analogous to that of
hash-based maps. Thanks to Almog Gavra for implementing the method.

7.0.8

- Non-indirect priority queues are now serializable.

- Fixed implementation of structures based on a custom hash: keys
strategy-equal to zero zero would not be managed correctly. Thanks to
Shawn Cao for reporting this bug.

- Natural/opposite/abstract comparators are now serializable.

7.0.7

- Now we check whether ranges of parallel sorting algorithms
 are too small *before* creating the thread pool.

- Merged Erich Schubert's fix for Object{AVL,RB}TreeSet.get().

7.0.6

- Faster priority queues: better variable caching, deleted
 a spurious check, tests for parameters turned into assertions.

- New collection-based constructors for heap-based priority queues.

- Reviewed ObjectArrays.newArray() so that there is a fast track
 for reallocation of arrays of type Object[].

7.0.4

- Fixed old-standing bug: iterators in linked maps would
 return bogus data on entrySet().next()/entrySet().previous()
 when no element is available instead of throwing an exception.

7.0.3

- Fixed wrong generation of custom-hash classes with primitive
 keys. Thanks to Michael Henke for reporting this bug.

7.0.2

- Now we shutdown() correctly ForkJoinPool's.

- Constants limiting parallelism and recursion have been tuned.

- New implementations of indirect [parallel] quicksort (in ascending order
 only).

- New stabilization method for post-processing of non-stable indirect
 sorts.

7.0.1

- Now generated sources are formatted using the Eclipse command-line
 facility.

7.0.0

- Now we need Java 7.

- New parallel versions of radix sort and quicksort. The sequential
implementations have been further improved.

- Restored the previous constants in mixing functions.

6.6.4

- Hopefully better mixing functions created by a genetic algorithm.

- Fixed a bug in floating-point hash-based containers: -0.0 and +0.0
were both converted to +0.0. Thanks to Dawid Weiss for reporting
this bug.

6.6.3

- Fixed subtle wrap-around bug in removal from iterator. Thanks to Eugene
Yakavets for reporting this bug.

6.6.2

- We now reduce backing arrays of hash-based classes when they are filled
below one fourth of the load factor. The reduction is not performed when
deleting from an iterator, as it would make iteration impossible.

- Significant simplification of Iterator.remove()'s implementations
for hash-based data structures.

6.6.1

- Fixed missed implementation: setValue() was not implemented for fast
iterators in hash-based maps.

6.6.0

- Major (transparent) rewrite of all hash-based classes inspired by the
Goldman-Sachs collections. We no longer allocate a byte array to store
the status of each slot: a null (or zero) key denotes an empty slot. The
null key is handled separately. The reduction in memory accesses makes
the cost of the additional logic negligible, and brings in significant
performance improvements. The code is actually simplified, as all loops
become a search for a nonzero element.

- Partial (one-step) unrolling of all lookup loops, following the strategy
used in Koloboke.

- Fixed an old bug: entrySet().remove(Entry) would remove entries checking
the value of the key, only.

- Fixed a bug in the iterator over hash big sets.

- OSGI metadata, thanks to Benson Margulies.

6.5.17

- Now TextIO methods trim strings before parsing numbers. This avoids obnoxious exceptions when numbers are followed by whitespace.

6.5.16

- Improved speed of FastMultiByteArrayInputStream, and removed support for mark()/reset().

- Deprecated array fill() methods in favour of java.util's.

6.5.15

- De-deprecated quicksort methods for primitive-type arrays. It turned out that Java's Arrays.sort() switches to mergesort on large, semi-sorted arrays. Moreover, in Java 7 the support array is allocated of the same size of the argument array, not of the sorted fragment. This performance bug was entirely killing the performance of Transform.transposeOffline() and other methods. Until that bug is fixed, we will have to rely on our quicksort method (which is a pity, because Java's sort is, for the rest, so beautifully engineered).

6.5.14

- Equality in type-specific hash-based data structures with float or double keys is now checked by converting to int/long bits using the conversion method of the appropriate class. Previously, using NaNs as keys would have led to misbehaviour. Thanks to Davide Savazzi for reporting this bug.

6.5.13

- Fixed a very unlikely corner case that might have led to reduction in size of an array instead of a growth. Thanks to Ernst Reissner for reporting this bug.

- InspectableFileCachedInputStream no longer performs a call to RandomAccessFile.position() when the end of file has been reached and the file is entirely held in memory.

- All front-coded lists now implement java.util.RandomAccess.

6.5.12

- Removed some useless wrapper creation in a few methods of tree-based map classes.

- Fixed pathological maxFill computation for very small-sized big open hash sets.

6.5.11

- A very old and subtle performance bug in hash-based data structures has been fixed. Backing arrays were allocated using the number of expected elements divided by the load factor. However, since the test for rehashing was fired by equality with the table size multiplied by the load factor, if the expected number of elements multiplied by the load factor was an integer a useless rehash would happen for the very last added element. The only effect was an useless increase in object creation.

6.5.10

- Now iterators in object set constructors are of type Iterator, and not anymore ObjectIterator. The kind of allowed iterators has been rationalised and made uniform through all classes implementing Set.

6.5.9

- New methods to get a type-specific Iterable from binary or text files.

6.5.8

- Fixed stupid bug in creation of array-based FIFO queues.

6.5.7

- Fixed a very subtle bug in hash-based data structures: addAll() to a newly created structure could require a very long time due to correlation between the positions in structures with different table sizes.

6.5.6

- equals() method between arrays have been deprecated in favour of the java.util.Arrays version, which is intrinsified in recent JVMs.

- InspectableFileCachedInputStream.reopen() makes it possible to read again from the start an instance on which close() was

invoked.

6.5.5

- The abstract implementation of equals() between (big) lists now uses type-specific access methods (as the compareTo() method was already doing) to avoid massive boxing/unboxing. Thanks to Adrien Grand for suggesting this improvement.

- FIFO array-based queues are now serializable.

6.5.4

- Further fixes related to NaNs in sorting.

- Fixed very old bug in FastByteArrayOutputStream.write(int). Thanks to Massimo Santini for reporting this bug.

- We now use Arrays.MAX_ARRAY_SIZE, which is equal to Integer.MAX_VALUE minus 8, to bound all array allocations. Previously, it might happen that grow() and other array-related functions could try to allocate an array of size Integer.MAX_VALUE, which is technically correct from the JLS, but will not work on most JVMs. The maximum length we use now is the same value as that used by java.util.ArrayList. Thanks to William Harvey for suggesting this change.

6.5.3

- Corrected erroneous introduction of compare() methods on integral classes (they appeared in Java 7).

6.5.2

- A few changes were necessary to make fastutil behave as Java on NaNs when sorting. Double.compareTo() and Float.compareTo() treat Double.NaN as greater than Double.POSITIVE_INFINITY, and fastutil was not doing it. As part of the change, now all comparisons between primitive types are performed using the compare() method of the wrapper class (microbenchmarks confirmed that there is no speed penalty for that, probably due to inlining or even intrinsification). Thanks to Adam Klein for reporting this bug.

- All quickSort() implementations that do not involve a comparator are now deprecated, as there are equivalent/better versions in java.util.Arrays.

6.5.0 -> 6.5.1

- Now FastBuffered{Input/Output}Stream has a constructor with an

explicitly given buffer.

- Abandoned golden-ratio based expansion of arrays and lists in favour of a (more standard) doubling approach.

- Array-based FIFO queues now reduce their capacity automatically by halving when the size becomes one fourth of the length.

- The add() method for open hash maps has been deprecated and replaced by addTo(), as the name choice proved to be a recipe for disaster.

- New InspectableFileCachedInputStream for caching easily large byte streams partially on file and partially in memory.

- The front() method for semi-indirect heaps took no comparator, but was used in queues in which you could support a comparator. There is now a further version accepting a comparator.

- Serial Version UIDs are now private.

6.4.6 -> 6.5.0

- Fixed type of array hash strategies.

- Fixed use of equals() instead of compareTo() in SemiIndirectHeaps.front(). Thanks to Matthew Hatem for reporting this bug.

- Now we generate custom hash maps for primite types, too (as we were already doing for sets).

6.4.5 -> 6.4.6

- In array-based priority queues changed() would not invalidate the cached index of the smallest element.

6.4.4 -> 6.4.5

- In some very rare circumstances, enumeration of hash sets or maps combined with massive element removal (using the iterator remove() method) could have led to inconsistent enumeration (duplicates and missing elements). Thanks to Hamish Morgan for reporting this bug.

6.4.3 -> 6.4.4

- Array-based maps were not implementing correctly entrySet().contains(), and as a consequence equals() between such maps was broken. Thanks to Benson Margulies for reporting this bug.

6.4.2 -> 6.4.3

- Now array-based priority queue cache their first element. Moreover,
 they implement the correct type-specific interface.

6.4.1 -> 6.4.2

- Now we have indirect lexicographical radix sort on pairs of arrays,
 mainly used to compute quickly Kendall's tau.

- New reverse method for arrays (useful for radix descending sorts).

- Radix sort (one or two arrays) for big arrays.

- Now radix sort uses correctly (minimally) sized support arrays when
 sorting subarrays.

6.4 -> 6.4.1

- Now we have a separate directory, settable in the makefile, to generate
 sources. This makes Maven integration easier.

- The store methods in TextIO for big arrays were broken.

- Now big-array lists implement the Stack interface.

- Fixed subtle bug in rehash() methods of big hash sets.

6.3 -> 6.4

- WARNING: Indirect queues must obviously have a way to determine whether
 an index is in the queue. It was an oversight in the interface design
 that a contains() method was not present. We wook the risk of adding it
 now. At the same time, we modified remove() so that now returns a
 boolean specifying whether the index to be removed was actually in the
 queue, as this is more in line with the Java Collections Framework.

- Removed unused double-priority queue related classes.

- Now array-based sets and maps have a constructor based on
 java.util.Collection and java.util.Map (as for the other
 kind of sets and maps).

- New doubly linked implementation for linked hash maps and sets. It uses
 twice the space for pointers, but mixes well with linear probing, so we
 have again constant-time true deletions. Moreover, iterators can be
 started from any key in constant time (albeit the first access to the

index of the list iterator will require a linear scan, unless the
iterator started from the first or the last key). Additional methods
such as getAndMoveToFirst() make the creation of LRU caches very easy.
Thanks to Brien Colwell for donating the code.

- Now object-based array FIFO queues provide deque methods. Moreover,
they clean up the backing array after returning an object or when
performing a clear().

- New get() method in set implementations makes it possible to recover
the actual object int the collection that is equal to the query key.

- A number of bugs were found and fixed by Christian Falz (thanks!). In
all binary search code the "to" parameter was *inclusive*, but the
documentation said *exclusive*, with obvious problems. Hash map
iterators could return under some very subtle and almost irreproducible
circumstances a previously deleted slot. Deleted hash map entries would
return spurious null values.

6.2.2 -> 6.3

- We now have radix sort. It's much faster than quicksort, but it can
only sort keys in their natural order. There are multiple-array
and indirect (and possibly stable) versions available.

- There are now custom hash sets also for type-specific keys. This makes
it possible to use hash sets to index data indirectly (e.g., using
integer or long just as indices).

- Shuffling static methods for all kinds of (big) list and arrays.

6.2.1 -> 6.2.2

- A new add() method makes the usage of maps as counters easier
and faster.

6.2.0 -> 6.2.1

- A very stupid bug was causing twice the rehashing that was
necessary. Now insertions in hash-based classes are significantly faster.

6.1.0 -> 6.2.0

- A better structure of the scan loop for hash tables borrowed
from HPPC (http://labs.carrotsearch.com/hppc.html) gives some
speed improvement to hash-based classes.

6.0.0 -> 6.1.0

- Hash-based classes have been rewritten using linear probing and
a good hash (MurmurHash3). The old classes can be still generated
using the target oldsources.

- Bizarre queues (double- and sesqui-indirect) have been removed
from the standard jar, but they can be still generated using the
target oldsources.

5.1.5 -> 6.0.0

- WARNING: the jar file is now fastutil.jar (not fastutil5.jar), again.

- WARNING: now fastutil requires Java 6+.

- fastutil is now released under the Apache License 2.0.

- New framework for big arrays, represented as arrays-of-arrays.
BigArrays and the type-specific counterparts provide static
methods of all kinds.

- New Size64 interface for classes implementing big collections.

- New framework for big lists--lists with longs as indices. The only
present implementation uses big arrays, but, for instance,
Sux4J's succinct lists will be retrofitted to LongBigList
(presently they implement LongBigList from dsiutils, which will
be deprecated).

- List.iterator() now returns a ListIterator. There is no real reason
not to do this, and the API change is handled from an implementation
viewpoint in AbstractList, so nodoby should really notice.

- New Collections.asCollection(Iterable) method to expose iterables as
collections (missing methods are computed using the iterator). This was
also the occasion to streamline type-specific abstract collections,
which now inherit from java.util.AbstractCollection, so we support
contains, clear, etc. methods as long as there is an iterator.

- Fixed bugged array-based constructors of ArrayMap and ArraySet.

- Fixed bugged put/remove methods in abstract functions. Thanks to
Katja Filippova for reporting this bug.

- New front-coded lists use big arrays, so they can store much more
(in fact, unlimited) data. Unfortunately, they are no longer
serialisation-compatible with previous versions.

- New MeasurableStream interface that is implemented by
MeasurableInputStream and by a new, analogous MeasurableOutputStream.

- Better FastBufferedOutputStream and FastByteArrayOutputStream that
are measurable and positionable.

- Now all clone() methods override covariantly the defult return type
(Object).

5.1.4 -> 5.1.5

- ArraySet was implementing isEmpty() with inverted logic (thanks to
Marko Srdanovic for reporting this bug).

- New constructor for FastMultiByteArrayInputStream: it takes a
MeasurableInputStream and uses length() to determine the number
of bytes to load into memory.

5.1.3 -> 5.1.4

- The implementation of RepositionableStream in FastByteArrayOutputStream
was fraught with a horrendous bug (thanks to Claudio Corsi for reporting),
in spite of extensive unit tests.

5.1.2 -> 5.1.3

- A bug existing since the first release was preventing tables
larger than 2^30 bits to work (the computation of the next bucket
to look at would cause an integer overflow).

- FastByteArrayOutputStream now implements RepositionableStream.

- Type-specific versions of Iterable.

- Some methods (e.g., iterator() and values()) are now explicitly
re-strengthened wherever necessary to avoid complaints about
ambiguous method invocations by some compilers.

- The introduction of functions added several bugs to the empty/singleton
map classes. Inheriting from the respective function counterparts left
several methods underspecified (equals(), etc.). This has been
(hopefully) fixed.

5.1.1 -> 5.1.2

- FastBufferedInputStream now supportw length() by FileChannel-fetching
on FileInputStream instances (it already used to support position()
by the same mechanism).

5.1.0 -> 5.1.1

- Byte-array MG4J I/O classes have been moved here.

5.0.9 -> 5.1.0

- Fixed documentation for custom/noncustom maps (it was exchanged).

- New type-specify entrySet() methods that avoid complicated casting
 to get a type-specific entryset. Moreover, now entrySet() can
 return an object implementing Fast(Sorted)EntrySet to indicate
 that a fastIterator() method is available. Fast iterators can
 return always the same Entry object, suitably mutated. We thank
 Daniel Ramage for suggesting this feature.

- Several hundreds of new classes generated by the new Function interface,
 which represent mappings for which the entry set is not enumerable
 (e.g., hashes). Functions have their usual share of satellite objects
 (wrappers, etc.). There are no implementations--the main purpose of
 the new interfaces is to make Sux4J (http://sux.dsi.unimi.it/)
 more object-oriented.

5.0.8 -> 5.0.9

- Slightly reduced overhead for bound checks in heap-based queues.

- BinIO was loading byte arrays one byte at a time. Now some conditionally
 compiled code uses bulk-read methods instead. Moreover, horrible kluges
 to work around Java bug #6478546 have been included.

5.0.7 -> 5.0.8

- Faster array maps and sets: System.arraycopy() is very slow on small arrays
 (due to inherent costs of calling native code) and reflection-based array
 creation is a disaster. Now we use object arrays and loops.

- New clone() methods for array-based structures and custom serialisation.

- FastBuffered*Stream has been simplified and streamlined. No more block alignment.

5.0.6 -> 5.0.7

- Better algorithm for front() in heaps.

- New comprehensive collection of array-based maps and sets. The motivation
 behind such structures is the need for quick, low-footprint data
 structures for *very* small sets (say, less than 10 elements). For

instance, in MG4J we were using sparse reference-based hash tables, but
it turned out that System.identityHashCode() is *deadly* slow and
scanning linearly an array searching for the desired element is
significantly faster.

5.0.5 -> 5.0.6

- Due to erratic and unpredictable behaviour of InputStream.skip(), which
does not correspond to its specification and Sun refuses to fix (see bug
6222822; don't be fooled by the “closed, fixed” label),
FastBufferedInputStream now peeks at the underlying stream and if it is
System.in it uses repeated reads. Moreover, it will use alternatively
reads and skips to guarantee that the number of skipped bytes will be
smaller than requested only if end of file has been reached.

- The insertion and key retrieval methods of hash-based structures are
now protected and final.

- New front() method for indirect queues. It retrieves quickly the indices
associated to elements equal to the top.

- First JUnit tests.

5.0.4 -> 5.0.5

- Fixed possible overflow in FastBufferedInputStream.available().

- Indirect heaps have faster checks for elements belonging or not to the
queue. In particular, we just rely on array access for detecting indices
out of bounds. Profiling with LaMa4J showed that in some circumstances
checking explicitly the indices were within bounds was taking more time
that the actual heap inner workings.

- Fixed obnoxious bug dating to the first fastutil implementation. The
macro KEY_EQUALS_HASH(x,h,y), which checks for equality between x and y
given that the hash of x is h, was evaluating hashCode() on y without
guarantee that y was non-null. As a result, adding a null to a mapped
followed by the insertion of an element with hash code 0 would have
thrown a NullPointerException. The bug went unobserved for years because
no one use nulls as keys, and was actually detected by a bug in BUbiNG's
code (which was in turn mistakenly inserting nulls in a set).

5.0.3 -> 5.0.4

- Fixed missing declaration of generic type for HASH_STRATEGY.

- A new abstract class, MeasurableInputStream, is used for streams
whose length and current position are always known. This actually

was needed for BUbiNG development.

- New readLine() family of method for reading "lines" directly
from a FastBufferedInputStream.

- In FastBufferedInputStream, reset() has been deprecated in favour
of flush().

- Array-based lists of objects now reallocate the backing array
using reflection *only* if they were created by wrapping. This
won't change the previous behaviour, but at the price of a boolean
per list we have unbelievably faster array reallocation.

- New explicit fast load factors in Hash.

5.0.2 -> 5.0.3

- Bizarrily, java.util.List re-specifies iterator(), even if it extends
Collection. As a result, we need to re-strengthen it in type-specific lists.

- Fixed new horrible bug introduced by adding Booleans to BinIO and TextIO.
Problem is, I didn't know #assert is cumulative.

5.0.1 -> 5.0.2

- Fixed bug in sorted maps key sets and values that would cause a
stack overflow when calling size() and a few other methods.

- Fixed lack of booleans in BinIO and TextIO.

- BinIO now checks for too large files.

5.0 -> 5.0.1

- In BinIO, it was assumed that .SIZE would give the size of
primitive types in *bytes*. Bad mistake.

4.4.3 -> 5.0

- Java 5 only!

- Support for generics. This led to a number of backward-incompatible changes:
  * toArray(Object[]) does not accept any longer null as an argument;
  * singletons for empty collections (sets, lists, ecc.) are type-specific;
  * iterators on sorted collections are bidirectional *by specification*;
  * the new, covariantly stronger methods defined in all interfaces (e.g.,
    iterator() returning a type-specific iterator) are now the default,
    and in the abstract classes the old methods (e.g., objectIterator())

now just delegate to the standard method, which is the contrary
of what was happening before: you'll have to turn all methods
such as objectIterator() in iterator(), etc.
* all deprecated methods have been dropped.

- Array growth functions now will return the correct empty array for
object arrays (it used to return ObjectArrays.EMPTY_ARRAY).

- Strategies are generic and no longer required to accept REMOVED.

- Stale references could hang around in the nodePath array for
Red-Black trees and map; this has been fixed.

- The difference in semantics with the standard toArray(Object[])
specification, which has always been in place, is now exhaustively
explained.

- Major code cleanup (mostly code deletion) due to passing fastutil
into Eclipse to check unused code, etc.

4.4.2 -> 4.4.3

- Important bug fix in FastBufferedInputStream.

4.4.1 -> 4.4.2

- New reset() method to invalidate the buffer of a FastBufferedInputStream,
making it possible to read safely files written by other processes
(given, of course, that you are synchronising the accesses).

4.4.0 -> 4.4.1

- New parallel-array constructor for all maps. Very useful for
static final map initialisation.

- Following considerations in Jakarta Commons I/O, the standard
buffer size has be lowered to 8Ki.

- Some arguments were declared as DataInputStream instead of
DataInput.

- New methods for reading/writing objects from/to streams.

4.3.2 -> 4.4

- New static containers for reading and writing easily text and binary
data streams. They load/save arrays, iterators etc. to buffered readers
or streams.

- Moved here fast input/output buffered classes from MG4J. This makes
fastutil self-contained.

- The trivial implementation of the type-specific iterator was missing
from AbstractList.drv (surprisingly, not from the subclass implementation!).

- The sublist implementation in AbstractList.drv is now protected and static.
The attributes are protected, too.

- Now we compare booleans (false<true). As a result, also lists of
booleans do get lexicographical comparability.

- add(k) in AbstractList.drv now calls add(size(), k).

- Fixed error messages for out-of-bound indices in lists.

4.3.1 -> 4.3.2

- Fixed small innocuous bug: a code fragment related to non-linked
hash table was generated for linked hash tables, too, do to a
case type in a preprocessor directive. The code fragment, however,
had no effect.

- Fixed memory leak in OpenHashMap: the remove() method was not clearing
the key (whereas OpenHashSet was).

4.3 -> 4.3.1

- New fully indirect heap-based double priority queues.

- Fixed docs for queues: in 4.3, we were claiming that greater elements
are dequeued first, while the opposite happens.

4.2 -> 4.3

- New full-fledged set of unmodifiable structures *and* iterators.

- Removed about a dozen spurious final method modifiers.

- Made rehash() protected, so that everybody can play with different
rehashing strategies.

- trim() in array lists wasn't doing the right thing, because trim(int)
wasn't doing it in the first place. Now if n is smaller than the size
of the list, we trim at the list size (previously we were doing nothing).

- Analogously, trim() in hash-table-based structures was fixed so that

trimming a table below its size will result in rehashing to the minimum possible size.

4.1 -> 4.2

- Improved array methods: now all methods on objects (e.g., grow()) return an array of the same type of the array that was passed to them, similarly to toArray() in collections.

- Fixed missing macro substitution for empty iterator methods. In any case, they were already deprecated.

4.0 -> 4.1

- New classes for custom hashing methods (mainly thought for arrays). Correspondingly, methods for arrays have been implemented in the static containers.

- BasicEntry now throws an UnsupportedOperationException on calls to setValue(). If you ever used that method, you got wierd results, as it does not update the underlying map. The method is now implemented correctly in open hash maps, in which previously did not correctly update the underying map.

- Reimplemented copy of an entire array using clone().

- Fixed a bug in clear() for indirect heaps (the inversion array was not being cleared).

- Indirect priority queue interfaces now feature an optional allChanged() method that can be used to force a complete heap rebuild. It is implemented by all current array-based and head-based concrete classes.

3.1 -> 4.0

- IMPORTANT: The optimized methods that a type-specific must provide now include an addElements() method that quickly adds an array of elements. As usual, the method is fully implemented by the type-specific abstract lists.

- IMPORTANT: The abstract generic version of get(), put() and remove() for maps with non-object keys or values now always return null to denote a missing key. They used to return an object-wrapped default return value.

- Completely new and comprehensive implementation of priority queues, both direct and indirect. Implementations are by heaps and by flat arrays. There are also static containers with all relevant heap methods, for people wanting to do their own thing.

- New static containers for comparators.

- All singletons, empty sets and snychronized wrappers are public so
you can inherit from them.

- Abstract maps now provide keySet() and values() based on entrySet().

- New abstract classes for sorted sets and maps with delegators to
type-specific methods.

- New public methods in Arrays and in type-specific Arrays classes for checking
ranges.

- New static methods for type-specific arrays that allow to grow, enlarge
and trim them with ease.

- Clarified abstract implementation of default return values, and implemented
clarified specification. Just a couple of method in hash maps were not
already compliant.

- The pour() method now returns a list. The previous version was returning
a linked hash set, which was rather nonsensical anyway, since an iterator
build on the returned set could have been different from the original iterator.
You can always pour an iterator into a set by providing the set explicitly.

- An exception-throwing implementation of some methods in AbstractSet
was missing. Same for AbstractCollection, AbstractMap and AbstractList.

- New basic inner entry class for abstract maps, which makes it easier to write
entrySet() methods for classes that do not have their own entries.

- Added missing get(Object) method in AbstractMap (just delegates to
the type-specific version).

- For lazy people, now containsKey() and containsValue() in AbstractMap
are defined by looking into keySet() and values().

- Fixed a few methods of EMPTY_LIST which were throwing exception
semantically (see the introduction).

- The interval iterators are now list iterators, except for longs.

- Fixed a bug in size() for array lists (reducing the size of an array
would lead to an exception).

- Fixed double bug in hash tables: first of all, on very small sizes adding
growthFactor would have left the size unchanged, giving rise to infinite
loops. (Thanks to Heikki Uusitalo for reporting this bug.) Second, growthFactor

was not being used *at all* by hash maps.

- Fixed entries emitted by singleton maps. Now they are type-specific.

- Fixed a number of minor glitches in gencsource.sh, and added some comments.

- HashCommon.removed has been renamed HashCommon.REMOVED.

- Boolean objects are now generated using valueOf() instead of the constructor.

- New type-specific wrappers for list iterators.

3.0 -> 3.1

- IMPORTANT: it.unimi.dsi.fastutil.Iterators methods have
been spread in type-specific static containers.

- New Stack interface, implemented by type-specific lists.

- New static container classes Collections, Sets, and Lists.
Presently they just provide empty containers.

- New type-specific static contains (e.g., IntSets) providing
singletons and synchronized wrappers.

- Entry sets now have entries that are equal() to entries
coming from corresponding maps in java.util.

- Spelling everywhere changed to Pure American. "synchronized" in code and
"synchronise" in text side-by-side were looking really wierd...

3.0 -> 3.0.1

- New unwrap() methods for type-specific collections.

- Fixed old-as-world-bug, apparently wide but that evidently no one ever
noticed: AbstractMap was not serialisable, and, as a result, the default
return value was not serialised (I find sincerely counterintuitive that
making a class serialisable doesn't do the same for its supertypes). It
wasn't ever even *documented* as preserved, so probably everyone thought
this was my idea, too. Too bad this breaks once more serialisation
compatibility. Since I had to break some serialisation anyway, I decided
to eliminate the residual serialisation of p in hash table classes, too
(which breaks serialisation for all hash-based classes).

2.60 -> 3.0

- IMPORTANT: All classes have been repackaged following the type of

elements/keys. Sources will have to be retouched (just to change
the import clause) and recompiled.

- IMPORTANT: Because of an unavoidable name clash in the new type-specific list
interface, the method remove(int) of IntCollection has been renamed rem(int).
The only really unpleasant effect is that you must use rem(int) on variables of
type IntCollection that are not of type IntSet (as IntSet reinstates
remove(int) in its right place)--for instance, IntList.

- Brand-new implementation of type-specific lists, with all the features you'd
expect and more.

- Insertions for readObject() in hash tables are now handled in a special way
(20% faster).

- Implemented linear-time tree reconstruction for readObject() (in practice, more
than twice faster).

- Fixed a problem with serialisation of hash tables: the table would have
been reloaded with the same p, even if it was preposterous. We still
save p, however, to avoid breaking serialisation compatibility.

- Fixed missing implementation of type-specific sets, which should
have extended type-specific collections, but they weren't.

- The default return value is now protected.

- New family of pour() methods that pour an iterator into a set.

- New programmable growth factor for hash-table-based classes.

- Eliminated a few useless method calls in tree map.

- Wide range of complex assertions, which are compiled in or out using the
"private static final boolean" idiom.

- For references we now use System.identityHashCode(); this shouldn't change
much, but it seems definitely more sensible.

- Fixed major bug in subSet()/subMap(): creating a subMap of a tailMap (or
headMap) a right extreme (left, resp.) equal to 0 would have caused the
creation of a tailMap (or headMap, resp.), discarding the extreme. Very,
very unlikely, but it happened in a test.

- Fixed small bug in standard remove() method of submaps, which would have
returned a default return value wrapped in a suitable object instead of
null on non-existing keys.

2.52 -> 2.60

- IMPORTANT: Major overhaul of iterators. Now iterators must be skippable,
 so previous implementation of type-specific iterator interfaces will
 not work. However, new abstract classes allow to build iterator with ease
 by providing for free the skipping logic, and many useful static methods in
 Iterators allow to generate type-specific iterators wrapping standard
 iterators, arrays, etc.

- Better strategy for clear() on hash tables: we don't do anything only
 if all entries are free (which means that an empty table with deleted
 entry will be cleared).

2.51 -> 2.52

- IMPORTANT: The package name has changed to it.unimi.dsi.fastutil to be
 uniform with JPackage conventions. However, this means that you must manually
 erase the old one and update your sources.

- clear() doesn't do anything on empty hash tables.

2.50 -> 2.51
- New trim(int) method to reduce a hash table size avoiding to
 make it too small.

- serialVersionUID is now fixed, to avoid future incompatibilities.

2.11 -> 2.50

- IMPORTANT: The Collection interface now prescribes an iterator
 method with a type-specific name (e.g., intIterator()) that
 returns directly a type-specific iterator.

- New Reference maps and sets that allow to store more quickly
 canonised objects.

- New linked maps mimicking java.util's, but with a boatload
 of additional features.

- Small bug fix: the get(Object) method would return null
 instead of the default return value for maps with object
 keys.

- Major bug fix: iterating backwards on submaps was leading
 to unpredictable results.

- Major bug fix: cloning maps would have caused inconsistent behaviour.

- Major code redistribution: now whenever possible wrappers
belong to abstract superclasses.

2.1 -> 2.11

- Now we cache the hash of an object before entering
the hash table loop.

2.0 -> 2.1

- A simple optimisation in hash-table inner loops has given
quite a performance boost under certain conditions (we do
not compute the secondary hashing if it is not necessary). Inspired
by Gnu Trove.
- The trim() method would have in fact trimmed nothing, just
rehashed the table.
- The computed maxFill value was sligtly too small.
- Also tree sets now have constructors from arrays.
- More internal methods have been made final.

1.3 -> 2.0

- ALL MAPS AND SETS HAVE NEW NAMES DEPENDING ON THE IMPLEMENTATION.
- Introducing new high-performance, low memory-footprint implementation of
SortedMap and SortedSet.
- Two tree implementations are available: RB trees and
AVL trees. Both implementations are threaded. See the README.
- Fixed a bug in hashCode() and contains() for HashMap.drv (it was
considering keys only!).
- Fixed a bug in contains() for entrySet() in all maps (it was using
VALUE_EQUAL to test equality for values given as objects).
- I realised that a default return value can be useful also for maps and sets
returning objects, so now you have it. It is even independent for submaps and
subsets.
- Classes are no longer final. The performance gain is around 1%, and
the decrease in usefulness is orders of magnitudes greater.
- We now check equality using first hashCode() and then equals().
- The tests for speed now warm up the trees by doing repeated insertions
and deletions, so that the benefits of a better balancing criterion
are more evident.
- The regression tests are much more stringent.
- Fixed hashCode() for hash maps (wasn't conforming to the Map interface
specification).
- Implemented linear cloning for tree classes.

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/CHANGES

No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import java.util.AbstractCollection;

/** An abstract class providing basic methods for collections implementing a type-specific interface.
*
* <p>In particular, this class provide {@link #iterator()}, {@code add()}, {@link #remove(Object)} and
* {@link #contains(Object)} methods that just call the type-specific counterpart.
*
* <p><strong>Warning</strong>: Because of a name clash between the list and collection interfaces
* the type-specific deletion method of a type-specific abstract
* collection is {@code rem()}, rather then {@code remove()}. A
* subclass must thus override {@code rem()}, rather than
* {@code remove()}, to make all inherited methods work properly.
*/

public abstract class ABSTRACT_COLLECTION KEY_GENERIC extends
AbstractCollection<KEY_GENERIC_CLASS> implements COLLECTION KEY_GENERIC {

 protected ABSTRACT_COLLECTION() {}

 @Override
 public abstract KEY_ITERATOR KEY_GENERIC iterator();

#if KEYS_PRIMITIVE

 /** {@inheritDoc}
  *
  * <p>This implementation always throws an {@link UnsupportedOperationException}.
```

```
 */
@Override
public boolean add(final KEY_TYPE k) {
 throw new UnsupportedOperationException();
}

/** {@inheritDoc}
 *
 * <p>This implementation iterates over the elements in the collection,
 * looking for the specified element.
 */
@Override
public boolean contains(final KEY_TYPE k) {
 final KEY_ITERATOR iterator = iterator();
 while (iterator.hasNext()) if (k == iterator.NEXT_KEY()) return true;
 return false;
}

/** {@inheritDoc}
 *
 * <p>This implementation iterates over the elements in the collection,
 * looking for the specified element and tries to remove it.
 */
@Override
public boolean rem(final KEY_TYPE k) {
 final KEY_ITERATOR KEY_GENERIC iterator = iterator();
 while (iterator.hasNext())
  if (k == iterator.NEXT_KEY()) {
   iterator.remove();
   return true;
  }
 return false;
}

/** {@inheritDoc}
 * @deprecated Please use the corresponding type-specific method instead.
 */
@SuppressWarnings("deprecation")
@Deprecated
@Override
public boolean add(final KEY_GENERIC_CLASS key) {
 return COLLECTION.super.add(key);
}

/** {@inheritDoc}
 * @deprecated Please use the corresponding type-specific method instead.
 */
@Deprecated
```

```
@Override
public boolean contains(final Object key) {
 return COLLECTION.super.contains(key);
}

/** {@inheritDoc}
 * @deprecated Please use the corresponding type-specific method instead.
 */
@Deprecated
@Override
public boolean remove(final Object key) {
 return COLLECTION.super.remove(key);
}

@Override
public KEY_TYPE[] toArray(KEY_TYPE a[]) {
 if (a == null || a.length < size()) a = new KEY_TYPE[size()];
 ITERATORS.unwrap(iterator(), a);
 return a;
}

@Override
public KEY_TYPE[] TO_KEY_ARRAY() {
 return toArray((KEY_TYPE[]) null);
}

/** {@inheritDoc}
 * @deprecated Please use {@code toArray()} instead&mdash;this method is redundant and will be removed in the
future.
 */
@Deprecated
@Override
public KEY_TYPE[] TO_KEY_ARRAY(final KEY_TYPE a[]) {
 return toArray(a);
}

@Override
public boolean addAll(final COLLECTION c) {
 boolean retVal = false;

 for(final KEY_ITERATOR i = c.iterator(); i.hasNext();)
  if (add(i.NEXT_KEY())) retVal = true;
 return retVal;
}

@Override
public boolean containsAll(final COLLECTION c) {
 for(final KEY_ITERATOR i = c.iterator(); i.hasNext();)
```

```
  if (! contains(i.NEXT_KEY())) return false;
 return true;
 }

 @Override
 public boolean removeAll(final COLLECTION c) {
  boolean retVal = false;
  for(final KEY_ITERATOR i = c.iterator(); i.hasNext();)
   if (rem(i.NEXT_KEY())) retVal = true;
  return retVal;
 }

 @Override
 public boolean retainAll(final COLLECTION c) {
  boolean retVal = false;
  for(final KEY_ITERATOR i = iterator(); i.hasNext();)
   if (! c.contains(i.NEXT_KEY())) {
    i.remove();
    retVal = true;
   }
  return retVal;
 }
#endif

 @Override
 public String toString() {
  final StringBuilder s = new StringBuilder();
  final KEY_ITERATOR KEY_GENERIC i = iterator();
  int n = size();
  KEY_TYPE k;
  boolean first = true;

  s.append("{");

  while(n-- != 0) {
   if (first) first = false;
   else s.append(", ");
   k = i.NEXT_KEY();
#if KEYS_REFERENCE
   if (this == k) s.append("(this collection)"); else
#endif
    s.append(String.valueOf(k));
  }

  s.append("}");
  return s.toString();
 }
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/AbstractCollection.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import it.unimi.dsi.fastutil.objects.ObjectBidirectionalIterable;
import it.unimi.dsi.fastutil.objects.ObjectBidirectionalIterator;
import it.unimi.dsi.fastutil.objects.ObjectSortedSet;
import it.unimi.dsi.fastutil.objects.ObjectSortedSets;
import PACKAGE.SORTED_MAP.FastSortedEntrySet;

import java.util.Comparator;
import java.util.Map;
import java.util.SortedMap;
import java.util.NoSuchElementException;

/** A class providing static methods and objects that do useful things with type-specific sorted maps.
*
* @see java.util.Collections
*/

public final class SORTED_MAPS {

 private SORTED_MAPS() {}

 /** Returns a comparator for entries based on a given comparator on keys.
  *
  * @param comparator a comparator on keys.
```

```
 * @return the associated comparator on entries.
 */
 public static KEY_GENERIC Comparator<? super Map.Entry<KEY_GENERIC_CLASS, ?>>
entryComparator(final KEY_COMPARATOR KEY_SUPER_GENERIC comparator) {
  return (Comparator<Map.Entry<KEY_GENERIC_CLASS, ?>>) (x, y) ->
comparator.compare(KEY_CLASS2TYPE(x.getKey()), KEY_CLASS2TYPE(y.getKey()));
 }

 /** Returns a bidirectional iterator that will be {@linkplain FastSortedEntrySet fast}, if possible, on the {@linkplain
Map#entrySet() entry set} of the provided {@code map}.
  * @param map a map from which we will try to extract a (fast) bidirectional iterator on the entry set.
  * @return a bidirectional iterator on the entry set of the given map that will be fast, if possible.
  * @since 8.0.0
  */
 SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
 public static KEY_VALUE_GENERIC ObjectBidirectionalIterator<MAP.Entry KEY_VALUE_GENERIC>
fastIterator(SORTED_MAP KEY_VALUE_GENERIC map) {
  final ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> entries = map.ENTRYSET();
  return entries instanceof SORTED_MAP.FastSortedEntrySet ? ((SORTED_MAP.FastSortedEntrySet
KEY_VALUE_GENERIC) entries).fastIterator() : entries.iterator();
 }

 /** Returns an iterable yielding a bidirectional iterator that will be {@linkplain FastSortedEntrySet fast}, if possible,
on the {@linkplain Map#entrySet() entry set} of the provided {@code map}.
  * @param map a map from which we will try to extract an iterable yielding a (fast) bidirectional iterator on the
entry set.
  * @return an iterable yielding a bidirectional iterator on the entry set of the given map that will be fast, if possible.
  * @since 8.0.0
  */
 SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
 public static KEY_VALUE_GENERIC ObjectBidirectionalIterable<MAP.Entry KEY_VALUE_GENERIC>
fastIterable(SORTED_MAP KEY_VALUE_GENERIC map) {
  final ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> entries = map.ENTRYSET();
  return entries instanceof SORTED_MAP.FastSortedEntrySet ? ((SORTED_MAP.FastSortedEntrySet
KEY_VALUE_GENERIC)entries)::fastIterator : entries;
 }


 /** An immutable class representing an empty type-specific sorted map.
  *
  * <p>This class may be useful to implement your own in case you subclass
  * a type-specific sorted map.
  */

 public static class EmptySortedMap KEY_VALUE_GENERIC extends MAPS.EmptyMap
KEY_VALUE_GENERIC implements SORTED_MAP KEY_VALUE_GENERIC, java.io.Serializable, Cloneable
{
```

```
  private static final long serialVersionUID = -7046029254386353129L;

  protected EmptySortedMap() {}

  @Override
  public KEY_COMPARATOR KEY_SUPER_GENERIC comparator() { return null; }

  @SuppressWarnings("unchecked")
  @Override
  public ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> ENTRYSET() { return
ObjectSortedSets.EMPTY_SET; }

#if KEYS_PRIMITIVE || VALUES_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
#else
  /** {@inheritDoc} */
#endif
  @Override
  @SuppressWarnings("unchecked")
  public ObjectSortedSet<Map.Entry<KEY_GENERIC_CLASS, VALUE_GENERIC_CLASS>> entrySet() { return
ObjectSortedSets.EMPTY_SET; }

  SUPPRESS_WARNINGS_KEY_UNCHECKED
  @Override
  public SORTED_SET KEY_GENERIC keySet() { return SORTED_SETS.EMPTY_SET; }

  SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
  @Override
  public SORTED_MAP KEY_VALUE_GENERIC subMap(final KEY_GENERIC_TYPE from, final
KEY_GENERIC_TYPE to) { return EMPTY_MAP; }

  SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
  @Override
  public SORTED_MAP KEY_VALUE_GENERIC headMap(final KEY_GENERIC_TYPE to) { return
EMPTY_MAP; }

  SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
  @Override
  public SORTED_MAP KEY_VALUE_GENERIC tailMap(final KEY_GENERIC_TYPE from) { return
EMPTY_MAP; }

  @Override
  public KEY_GENERIC_TYPE FIRST_KEY() { throw new NoSuchElementException(); }

  @Override
  public KEY_GENERIC_TYPE LAST_KEY() { throw new NoSuchElementException(); }
```

```
#if KEYS_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public SORTED_MAP KEY_VALUE_GENERIC headMap(KEY_GENERIC_CLASS oto) { return
headMap(KEY_CLASS2TYPE(oto)); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public SORTED_MAP KEY_VALUE_GENERIC tailMap(KEY_GENERIC_CLASS ofrom) { return
tailMap(KEY_CLASS2TYPE(ofrom)); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public SORTED_MAP KEY_VALUE_GENERIC subMap(KEY_GENERIC_CLASS ofrom,
KEY_GENERIC_CLASS oto) { return subMap(KEY_CLASS2TYPE(ofrom), KEY_CLASS2TYPE(oto)); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public KEY_GENERIC_CLASS firstKey() { return KEY2OBJ(FIRST_KEY()); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public KEY_GENERIC_CLASS lastKey() { return KEY2OBJ(LAST_KEY()); }
#endif
 }



 /** An empty sorted map (immutable). It is serializable and cloneable.
  */
 SUPPRESS_WARNINGS_KEY_VALUE_RAWTYPES
 public static final EmptySortedMap EMPTY_MAP = new EmptySortedMap();

#if KEYS_REFERENCE || VALUES_REFERENCE
 /** Returns an empty sorted map (immutable). It is serializable and cloneable.
  *
  * <p>This method provides a typesafe access to {@link #EMPTY_MAP}.
```

```
 * @return an empty sorted map (immutable).
 */
@SuppressWarnings("unchecked")
public static KEY_VALUE_GENERIC SORTED_MAP KEY_VALUE_GENERIC emptyMap() {
 return EMPTY_MAP;
}
#endif

/** An immutable class representing a type-specific singleton sorted map.
 *
 * <p>This class may be useful to implement your own in case you subclass
 * a type-specific sorted map.
 */

public static class Singleton KEY_VALUE_GENERIC extends MAPS.Singleton KEY_VALUE_GENERIC
implements SORTED_MAP KEY_VALUE_GENERIC, java.io.Serializable, Cloneable {

 private static final long serialVersionUID = -7046029254386353129L;

 protected final KEY_COMPARATOR KEY_SUPER_GENERIC comparator;

 protected Singleton(final KEY_GENERIC_TYPE key, final VALUE_GENERIC_TYPE value,
KEY_COMPARATOR KEY_SUPER_GENERIC comparator) {
  super(key, value);
  this.comparator = comparator;
 }

 protected Singleton(final KEY_GENERIC_TYPE key, final VALUE_GENERIC_TYPE value) {
  this(key, value, null);
 }

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 final int compare(final KEY_GENERIC_TYPE k1, final KEY_GENERIC_TYPE k2) {
  return comparator == null ? KEY_CMP(k1, k2) : comparator.compare(k1, k2);
 }

 @Override
 public KEY_COMPARATOR KEY_SUPER_GENERIC comparator() { return comparator; }

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 @Override
 public ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> ENTRYSET() { if (entries == null) entries =
ObjectSortedSets.singleton(new ABSTRACT_MAP.BasicEntry KEY_VALUE_GENERIC_DIAMOND(key,
value), entryComparator(comparator)); return (ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC>)entries; }

#if KEYS_PRIMITIVE || VALUES_PRIMITIVE
 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
```

```
  @Deprecated
#else
  /** {@inheritDoc} */
#endif
  @Override
  @SuppressWarnings({ "rawtypes", "unchecked" })
  public ObjectSortedSet<Map.Entry<KEY_GENERIC_CLASS, VALUE_GENERIC_CLASS>> entrySet() { return
(ObjectSortedSet)ENTRYSET(); }

  @Override
  public SORTED_SET KEY_GENERIC keySet() { if (keys == null) keys = SORTED_SETS.singleton(key,
comparator); return (SORTED_SET KEY_GENERIC)keys; }

  SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
  @Override
  public SORTED_MAP KEY_VALUE_GENERIC subMap(final KEY_GENERIC_TYPE from, final
KEY_GENERIC_TYPE to) { if (compare(from, key) <= 0 && compare(key, to) < 0) return this; return
EMPTY_MAP; }

  SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
  @Override
  public SORTED_MAP KEY_VALUE_GENERIC headMap(final KEY_GENERIC_TYPE to) { if (compare(key,
to) < 0) return this; return EMPTY_MAP; }

  SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
  @Override
  public SORTED_MAP KEY_VALUE_GENERIC tailMap(final KEY_GENERIC_TYPE from) { if (compare(from,
key) <= 0) return this; return EMPTY_MAP; }

  @Override
  public KEY_GENERIC_TYPE FIRST_KEY() { return key; }

  @Override
  public KEY_GENERIC_TYPE LAST_KEY() { return key; }

#if KEYS_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public SORTED_MAP KEY_VALUE_GENERIC headMap(KEY_GENERIC_CLASS oto) { return
headMap(KEY_CLASS2TYPE(oto)); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public SORTED_MAP KEY_VALUE_GENERIC tailMap(KEY_GENERIC_CLASS ofrom) { return
```

```
tailMap(KEY_CLASS2TYPE(ofrom)); }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 public SORTED_MAP KEY_VALUE_GENERIC subMap(KEY_GENERIC_CLASS ofrom,
KEY_GENERIC_CLASS oto) { return subMap(KEY_CLASS2TYPE(ofrom), KEY_CLASS2TYPE(oto)); }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 public KEY_GENERIC_CLASS firstKey() { return KEY2OBJ(FIRST_KEY()); }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 public KEY_GENERIC_CLASS lastKey() { return KEY2OBJ(LAST_KEY()); }
#endif
 }

 /** Returns a type-specific immutable sorted map containing only the specified pair. The returned sorted map is
serializable and cloneable.
 *
 * <p>Note that albeit the returned map is immutable, its default return value may be changed.
 *
 * @param key the only key of the returned sorted map.
 * @param value the only value of the returned sorted map.
 * @return a type-specific immutable sorted map containing just the pair {@code &lt;key,value&gt;}.
 */

 public static KEY_VALUE_GENERIC SORTED_MAP KEY_VALUE_GENERIC singleton(final
KEY_GENERIC_CLASS key, VALUE_GENERIC_CLASS value) { return new Singleton
KEY_VALUE_GENERIC_DIAMOND(KEY_CLASS2TYPE(key), VALUE_CLASS2TYPE(value));}

 /** RETURNS a type-specific immutable sorted map containing only the specified pair. The returned sorted map is
serializable and cloneable.
 *
 * <p>Note that albeit the returned map is immutable, its default return value may be changed.
 *
 * @param key the only key of the returned sorted map.
 * @param value the only value of the returned sorted map.
 * @param comparator the comparator to use in the returned sorted map.
 * @return a type-specific immutable sorted map containing just the pair {@code &lt;key,value&gt;}.
 */
```

```
 public static KEY_VALUE_GENERIC SORTED_MAP KEY_VALUE_GENERIC singleton(final
KEY_GENERIC_CLASS key, VALUE_GENERIC_CLASS value, KEY_COMPARATOR
KEY_SUPER_GENERIC comparator) { return new Singleton
KEY_VALUE_GENERIC_DIAMOND(KEY_CLASS2TYPE(key), VALUE_CLASS2TYPE(value), comparator);
}

#if KEYS_PRIMITIVE || VALUES_PRIMITIVE

 /** Returns a type-specific immutable sorted map containing only the specified pair. The returned sorted map is
serializable and cloneable.
  *
  * <p>Note that albeit the returned map is immutable, its default return value may be changed.
  *
  * @param key the only key of the returned sorted map.
  * @param value the only value of the returned sorted map.
  * @return a type-specific immutable sorted map containing just the pair {@code &lt;key,value&gt;}.
  */

 public static KEY_VALUE_GENERIC SORTED_MAP KEY_VALUE_GENERIC singleton(final
KEY_GENERIC_TYPE key, final VALUE_GENERIC_TYPE value) {
  return new Singleton KEY_VALUE_GENERIC_DIAMOND(key, value);
 }

 /** Returns a type-specific immutable sorted map containing only the specified pair. The returned sorted map is
serializable and cloneable.
  *
  * <p>Note that albeit the returned map is immutable, its default return value may be changed.
  *
  * @param key the only key of the returned sorted map.
  * @param value the only value of the returned sorted map.
  * @param comparator the comparator to use in the returned sorted map.
  * @return a type-specific immutable sorted map containing just the pair {@code &lt;key,value&gt;}.
  */

 public static KEY_VALUE_GENERIC SORTED_MAP KEY_VALUE_GENERIC singleton(final
KEY_GENERIC_TYPE key, final VALUE_GENERIC_TYPE value, KEY_COMPARATOR
KEY_SUPER_GENERIC comparator) {
  return new Singleton KEY_VALUE_GENERIC_DIAMOND(key, value, comparator);
 }

#endif


  /** A synchronized wrapper class for sorted maps. */

 public static class SynchronizedSortedMap KEY_VALUE_GENERIC extends MAPS.SynchronizedMap
KEY_VALUE_GENERIC implements SORTED_MAP KEY_VALUE_GENERIC, java.io.Serializable {
```

```
  private static final long serialVersionUID = -7046029254386353129L;

  protected final SORTED_MAP KEY_VALUE_GENERIC sortedMap;

  protected SynchronizedSortedMap(final SORTED_MAP KEY_VALUE_GENERIC m, final Object sync) {
   super(m, sync);
   sortedMap = m;
  }

  protected SynchronizedSortedMap(final SORTED_MAP KEY_VALUE_GENERIC m) {
   super(m);
   sortedMap = m;
  }

  @Override
  public KEY_COMPARATOR KEY_SUPER_GENERIC comparator() { synchronized(sync) { return
sortedMap.comparator(); } }

  @Override
  public ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> ENTRYSET() { if (entries == null) entries =
ObjectSortedSets.synchronize(sortedMap.ENTRYSET(), sync); return (ObjectSortedSet<MAP.Entry
KEY_VALUE_GENERIC>)entries; }

#if KEYS_PRIMITIVE || VALUES_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
#else
  /** {@inheritDoc} */
#endif
  @Override
  @SuppressWarnings({ "rawtypes", "unchecked" })
  public ObjectSortedSet<Map.Entry<KEY_GENERIC_CLASS, VALUE_GENERIC_CLASS>> entrySet() { return
(ObjectSortedSet)ENTRYSET(); }

  @Override
  public SORTED_SET KEY_GENERIC keySet() { if (keys == null) keys =
SORTED_SETS.synchronize(sortedMap.keySet(), sync); return (SORTED_SET KEY_GENERIC)keys; }

  @Override
  public SORTED_MAP KEY_VALUE_GENERIC subMap(final KEY_GENERIC_TYPE from, final
KEY_GENERIC_TYPE to) { return new SynchronizedSortedMap
KEY_VALUE_GENERIC_DIAMOND(sortedMap.subMap(from, to), sync); }

  @Override
  public SORTED_MAP KEY_VALUE_GENERIC headMap(final KEY_GENERIC_TYPE to) { return new
SynchronizedSortedMap KEY_VALUE_GENERIC_DIAMOND(sortedMap.headMap(to), sync); }
```

```
  @Override
  public SORTED_MAP KEY_VALUE_GENERIC tailMap(final KEY_GENERIC_TYPE from) { return new
SynchronizedSortedMap KEY_VALUE_GENERIC_DIAMOND(sortedMap.tailMap(from), sync); }

  @Override
  public KEY_GENERIC_TYPE FIRST_KEY() {  synchronized(sync) { return sortedMap.FIRST_KEY(); } }

  @Override
  public KEY_GENERIC_TYPE LAST_KEY() {  synchronized(sync) { return sortedMap.LAST_KEY(); } }

#if KEYS_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public KEY_GENERIC_CLASS firstKey() {  synchronized(sync) { return sortedMap.firstKey(); } }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public KEY_GENERIC_CLASS lastKey() {  synchronized(sync) { return sortedMap.lastKey(); } }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public SORTED_MAP KEY_VALUE_GENERIC subMap(final KEY_GENERIC_CLASS from, final
KEY_GENERIC_CLASS to) { return new SynchronizedSortedMap
KEY_VALUE_GENERIC_DIAMOND(sortedMap.subMap(from, to), sync); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public SORTED_MAP KEY_VALUE_GENERIC headMap(final KEY_GENERIC_CLASS to) { return new
SynchronizedSortedMap KEY_VALUE_GENERIC_DIAMOND(sortedMap.headMap(to), sync); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public SORTED_MAP KEY_VALUE_GENERIC tailMap(final KEY_GENERIC_CLASS from) { return new
SynchronizedSortedMap KEY_VALUE_GENERIC_DIAMOND(sortedMap.tailMap(from), sync); }
#endif


 }
```

```
 /** Returns a synchronized type-specific sorted map backed by the given type-specific sorted map.
  *
  * @param m the sorted map to be wrapped in a synchronized sorted map.
  * @return a synchronized view of the specified sorted map.
  * @see java.util.Collections#synchronizedSortedMap(SortedMap)
  */
 public static KEY_VALUE_GENERIC SORTED_MAP KEY_VALUE_GENERIC synchronize(final
SORTED_MAP KEY_VALUE_GENERIC m) { return new SynchronizedSortedMap
KEY_VALUE_GENERIC_DIAMOND(m); }

 /** Returns a synchronized type-specific sorted map backed by the given type-specific sorted map, using an
assigned object to synchronize.
  *
  * @param m the sorted map to be wrapped in a synchronized sorted map.
  * @param sync an object that will be used to synchronize the access to the sorted sorted map.
  * @return a synchronized view of the specified sorted map.
  * @see java.util.Collections#synchronizedSortedMap(SortedMap)
  */

 public static KEY_VALUE_GENERIC SORTED_MAP KEY_VALUE_GENERIC synchronize(final
SORTED_MAP KEY_VALUE_GENERIC m, final Object sync) { return new SynchronizedSortedMap
KEY_VALUE_GENERIC_DIAMOND(m, sync); }




 /** An unmodifiable wrapper class for sorted maps. */

 public static class UnmodifiableSortedMap KEY_VALUE_GENERIC extends MAPS.UnmodifiableMap
KEY_VALUE_GENERIC implements SORTED_MAP KEY_VALUE_GENERIC, java.io.Serializable {

  private static final long serialVersionUID = -7046029254386353129L;

  protected final SORTED_MAP KEY_VALUE_GENERIC sortedMap;

  protected UnmodifiableSortedMap(final SORTED_MAP KEY_VALUE_GENERIC m) {
   super(m);
   sortedMap = m;
  }

  @Override
  public KEY_COMPARATOR KEY_SUPER_GENERIC comparator() { return sortedMap.comparator(); }

  @Override
  public ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> ENTRYSET() { if (entries == null) entries =
ObjectSortedSets.unmodifiable(sortedMap.ENTRYSET()); return (ObjectSortedSet<MAP.Entry
KEY_VALUE_GENERIC>)entries; }
```

```
#if KEYS_PRIMITIVE || VALUES_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
#else
  /** {@inheritDoc} */
#endif
  @Override
  @SuppressWarnings({ "rawtypes", "unchecked" })
  public ObjectSortedSet<Map.Entry<KEY_GENERIC_CLASS, VALUE_GENERIC_CLASS>> entrySet() { return
(ObjectSortedSet)ENTRYSET(); }

  @Override
  public SORTED_SET KEY_GENERIC keySet() { if (keys == null) keys =
SORTED_SETS.unmodifiable(sortedMap.keySet()); return (SORTED_SET KEY_GENERIC)keys; }

  @Override
  public SORTED_MAP KEY_VALUE_GENERIC subMap(final KEY_GENERIC_TYPE from, final
KEY_GENERIC_TYPE to) { return new UnmodifiableSortedMap
KEY_VALUE_GENERIC_DIAMOND(sortedMap.subMap(from, to)); }

  @Override
  public SORTED_MAP KEY_VALUE_GENERIC headMap(final KEY_GENERIC_TYPE to) { return new
UnmodifiableSortedMap KEY_VALUE_GENERIC_DIAMOND(sortedMap.headMap(to)); }

  @Override
  public SORTED_MAP KEY_VALUE_GENERIC tailMap(final KEY_GENERIC_TYPE from) { return new
UnmodifiableSortedMap KEY_VALUE_GENERIC_DIAMOND(sortedMap.tailMap(from)); }

  @Override
  public KEY_GENERIC_TYPE FIRST_KEY() {  return sortedMap.FIRST_KEY(); }

  @Override
  public KEY_GENERIC_TYPE LAST_KEY() {  return sortedMap.LAST_KEY(); }

#if KEYS_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public KEY_GENERIC_CLASS firstKey() {  return sortedMap.firstKey(); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public KEY_GENERIC_CLASS lastKey() {  return sortedMap.lastKey(); }
```

```
 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 public SORTED_MAP KEY_VALUE_GENERIC subMap(final KEY_GENERIC_CLASS from, final
KEY_GENERIC_CLASS to) { return new UnmodifiableSortedMap
KEY_VALUE_GENERIC_DIAMOND(sortedMap.subMap(from, to)); }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 public SORTED_MAP KEY_VALUE_GENERIC headMap(final KEY_GENERIC_CLASS to) { return new
UnmodifiableSortedMap KEY_VALUE_GENERIC_DIAMOND(sortedMap.headMap(to)); }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 public SORTED_MAP KEY_VALUE_GENERIC tailMap(final KEY_GENERIC_CLASS from) { return new
UnmodifiableSortedMap KEY_VALUE_GENERIC_DIAMOND(sortedMap.tailMap(from)); }
#endif


 }

 /** Returns an unmodifiable type-specific sorted map backed by the given type-specific sorted map.
  *
  * @param m the sorted map to be wrapped in an unmodifiable sorted map.
  * @return an unmodifiable view of the specified sorted map.
  * @see java.util.Collections#unmodifiableSortedMap(SortedMap)
  */
 public static KEY_VALUE_GENERIC SORTED_MAP KEY_VALUE_GENERIC unmodifiable(final
SORTED_MAP KEY_VALUE_GENERIC m) { return new UnmodifiableSortedMap
KEY_VALUE_GENERIC_DIAMOND(m); }



#if defined(TEST) && ! KEY_CLASS_Reference

 private static long seed = System.currentTimeMillis();
 private static java.util.Random r = new java.util.Random(seed);

 private static KEY_TYPE genKey() {
#if KEY_CLASS_Byte || KEY_CLASS_Short || KEY_CLASS_Character
  return (KEY_TYPE)(r.nextInt());
#elif KEYS_PRIMITIVE
```

```
  return r.NEXT_KEY();
#else
  return Integer.toBinaryString(r.nextInt());
#endif
 }

 private static VALUE_TYPE genValue() {
#if VALUE_CLASS_Byte || VALUE_CLASS_Short || VALUE_CLASS_Character
  return (VALUE_TYPE)(r.nextInt());
#elif VALUES_PRIMITIVE
  return r.NEXT_VALUE();
#elif !VALUE_CLASS_Reference || KEY_CLASS_Reference
  return Integer.toBinaryString(r.nextInt());
#else
  return new java.io.Serializable() {};
#endif
 }


 private static java.text.NumberFormat format = new java.text.DecimalFormat("#,###.00");
 private static java.text.FieldPosition p = new java.text.FieldPosition(0);

 private static String format(double d) {
  StringBuffer s = new StringBuffer();
  return format.format(d, s, p).toString();
 }

 private static void speedTest(int n, boolean comp) {
  System.out.println("There are presently no speed tests for this class.");
 }


 private static boolean valEquals(Object o1, Object o2) {
  return o1 == null ? o2 == null : o1.equals(o2);
 }

 private static void fatal(String msg) {
  System.out.println(msg);
  System.exit(1);
 }

 private static void ensure(boolean cond, String msg) {
  if (cond) return;
  fatal(msg);
 }

 private static Object[] k, v, nk;
 private static KEY_TYPE kt[];
```

```
 private static KEY_TYPE nkt[];
 private static VALUE_TYPE vt[];
 private static SORTED_MAP topMap;

 protected static void testMaps(SORTED_MAP m, SortedMap t, int n, int level) {
  long ms;
  boolean mThrowsIllegal, tThrowsIllegal, mThrowsNoElement, tThrowsNoElement, mThrowsUnsupp,
tThrowsUnsupp;
  Object rt = null, rm = null;

  if (level > 1) return;


  /* Now we check that both maps agree on first/last keys. */

  mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

  try {
   m.firstKey();
  }
  catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
  try {
   t.firstKey();
  }
  catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): firstKey() divergence at
start in java.util.NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
  if (! mThrowsNoElement) ensure(t.firstKey().equals(m.firstKey()), "Error (" + level + ", " + seed + "): m and t differ
at start on their first key (" + m.firstKey() + ", " + t.firstKey() +")");

  mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

  try {
   m.lastKey();
  }
  catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
  try {
   t.lastKey();
  }
  catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): lastKey() divergence at start
in java.util.NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");


  if (! mThrowsNoElement) ensure(t.lastKey().equals(m.lastKey()), "Error (" + level + ", " + seed + "): m and t differ
at start on their last key (" + m.lastKey() + ", " + t.lastKey() +")");
```

```
  /* Now we check that m and t are equal. */
  if (!m.equals(t) || ! t.equals(m)) System.err.println("m: " + m + " t: " + t);

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) at start");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) at start");



  /* Now we check that m actually holds that data. */
  for(java.util.Iterator i=t.entrySet().iterator(); i.hasNext();) {
   java.util.Map.Entry e = (java.util.Map.Entry)i.next();
   ensure(valEquals(e.getValue(), m.get(e.getKey())), "Error (" + level + ", " + seed + "): m and t differ on an entry
("+e+") after insertion (iterating on t)");
  }

  /* Now we check that m actually holds that data, but iterating on m. */
  for(java.util.Iterator i=m.entrySet().iterator(); i.hasNext();) {
   java.util.Map.Entry e = (java.util.Map.Entry)i.next();
   ensure(valEquals(e.getValue(), t.get(e.getKey())), "Error (" + level + ", " + seed + "): m and t differ on an entry
("+e+") after insertion (iterating on m)");
  }

  /* Now we check that m actually holds the same keys. */
  for(java.util.Iterator i=t.keySet().iterator(); i.hasNext();) {
   Object o = i.next();
   ensure(m.containsKey(o), "Error (" + level + ", " + seed + "): m and t differ on a key ("+o+") after insertion
(iterating on t)");
   ensure(m.keySet().contains(o), "Error (" + level + ", " + seed + "): m and t differ on a key ("+o+", in keySet()) after
insertion (iterating on t)");
  }

  /* Now we check that m actually holds the same keys, but iterating on m. */
  for(java.util.Iterator i=m.keySet().iterator(); i.hasNext();) {
   Object o = i.next();
   ensure(t.containsKey(o), "Error (" + level + ", " + seed + "): m and t differ on a key after insertion (iterating on m)");
   ensure(t.keySet().contains(o), "Error (" + level + ", " + seed + "): m and t differ on a key (in keySet()) after insertion
(iterating on m)");
  }



  /* Now we check that m actually hold the same values. */
  for(java.util.Iterator i=t.values().iterator(); i.hasNext();) {
   Object o = i.next();
   ensure(m.containsValue(o), "Error (" + level + ", " + seed + "): m and t differ on a value after insertion (iterating on
t)");
   ensure(m.values().contains(o), "Error (" + level + ", " + seed + "): m and t differ on a value (in values()) after
```

```
insertion (iterating on t)");
  }

  /* Now we check that m actually hold the same values, but iterating on m. */
  for(java.util.Iterator i=m.values().iterator(); i.hasNext();) {
   Object o = i.next();
   ensure(t.containsValue(o), "Error (" + level + ", " + seed + "): m and t differ on a value after insertion (iterating on
m)");
   ensure(t.values().contains(o), "Error (" + level + ", " + seed + "): m and t differ on a value (in values()) after
insertion (iterating on m)");
  }

  /* Now we check that inquiries about random data give the same answer in m and t. For
     m we use the polymorphic method. */

  for(int i=0; i<n;  i++) {
   KEY_TYPE T = genKey();

   mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

   try {
    m.containsKey(KEY2OBJ(T));
   }
   catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
   catch (IllegalArgumentException e) { mThrowsIllegal = true; }

   try {
    t.containsKey(KEY2OBJ(T));
   }
   catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
   catch (IllegalArgumentException e) { tThrowsIllegal = true; }

   ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): containsKey() divergence in
java.util.NoSuchElementException (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
   ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): containsKey() divergence in
IllegalArgumentException (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
   if (!mThrowsNoElement && !mThrowsIllegal) {
    ensure(m.containsKey(KEY2OBJ(T)) == t.containsKey(KEY2OBJ(T)), "Error (" + level + ", " + seed + "):
divergence in keys between t and m (polymorphic method)");

#if KEY_CLASS_Object && ! (VALUES_REFERENCE)
    if ((m.GET_VALUE(T) != VALUE_NULL) != ((t.get(KEY2OBJ(T)) == null ? VALUE_NULL :
VALUE_OBJ2TYPE(t.get(KEY2OBJ(T)))) != VALUE_NULL) ||
     t.get(KEY2OBJ(T)) != null &&
     ! VALUE2OBJ(m.GET_VALUE(T)).equals(t.get(KEY2OBJ(T))))
#else
     if ((m.get(T) != VALUE_NULL) != ((t.get(KEY2OBJ(T)) == null ? VALUE_NULL :
VALUE_OBJ2TYPE(t.get(KEY2OBJ(T)))) != VALUE_NULL) ||
```

```
    t.get(KEY2OBJ(T)) != null &&
    ! m.get(KEY2OBJ(T)).equals(t.get(KEY2OBJ(T))))
#endif
    {
     System.out.println("Error (" + level + ", " + seed + "): divergence between t and m (polymorphic method)");
     System.exit(1);
    }
  }
  }

  /* Again, we check that inquiries about random data give the same answer in m and t, but
    for m we use the standard method. */

  for(int i=0; i<n;  i++) {
   KEY_TYPE T = genKey();

   mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

   try {
    m.get(KEY2OBJ(T));
   }
   catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
   catch (IllegalArgumentException e) { mThrowsIllegal = true; }

   try {
    t.get(KEY2OBJ(T));
   }
   catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
   catch (IllegalArgumentException e) { tThrowsIllegal = true; }

   ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): get() divergence in
java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
   ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): get() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
   if (!mThrowsNoElement && !mThrowsIllegal) ensure(valEquals(m.get(KEY2OBJ(T)), t.get(KEY2OBJ(T))),
"Error (" + level + ", " + seed + "): divergence between t and m (standard method)");
  }

  /* Now we put and remove random data in m and t, checking that the result is the same. */

  for(int i=0; i<20*n;  i++) {
   KEY_TYPE T = genKey();
   VALUE_TYPE U = genValue();

   mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = mThrowsUnsupp =
tThrowsUnsupp = false;

   try {
```

```
   rm = m.put(KEY2OBJ(T), VALUE2OBJ(U));
   }
   catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
   catch (IllegalArgumentException e) { mThrowsIllegal = true; }
   catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }

   try {
    rt = t.put(KEY2OBJ(T), VALUE2OBJ(U));
   }
   catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
   catch (IllegalArgumentException e) { tThrowsIllegal = true; }
   catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }

   ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): put() divergence in
java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
   ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): put() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
   //ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + level + ", " + seed + "): put() divergence in
UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);
   if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsUnsupp) ensure(valEquals(rm, rt), "Error (" + level +
", " + seed + "): divergence in put() between t and m (" + rt + ", " + rm + ")");

   T = genKey();

   mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = mThrowsUnsupp =
tThrowsUnsupp = false;

   try {
    rm = m.remove(KEY2OBJ(T));
   }
   catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
   catch (IllegalArgumentException e) { mThrowsIllegal = true; }
   catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }

   try {
    rt = t.remove(KEY2OBJ(T));
   }
   catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
   catch (IllegalArgumentException e) { tThrowsIllegal = true; }
   catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }


   ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): remove() divergence in
java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
   ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): remove() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
   //ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + level + ", " + seed + "): remove() divergence in
UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);
```

```
  if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsUnsupp) ensure(valEquals(rm, rt), "Error (" + level +
", " + seed + "): divergence in remove() between t and m (" + rt + ", " + rm + ")");
  }

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after removal");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after removal");

  /* Now we check that m actually holds the same data. */

  for(java.util.Iterator i=t.entrySet().iterator(); i.hasNext();) {
   java.util.Map.Entry e = (java.util.Map.Entry)i.next();
   ensure(valEquals(e.getValue(), m.get(e.getKey())), "Error (" + level + ", " + seed + "): m and t differ on an entry
("+e+") after removal (iterating on t)");
  }

  /* Now we check that m actually holds that data, but iterating on m. */

  for(java.util.Iterator i=m.entrySet().iterator(); i.hasNext();) {
   java.util.Map.Entry e = (java.util.Map.Entry)i.next();
   ensure(valEquals(e.getValue(), t.get(e.getKey())), "Error (" + level + ", " + seed + "): m and t differ on an entry
("+e+") after removal (iterating on m)");
  }

  /* Now we check that m actually holds the same keys. */

  for(java.util.Iterator i=t.keySet().iterator(); i.hasNext();) {
   Object o = i.next();
   ensure(m.containsKey(o), "Error (" + level + ", " + seed + "): m and t differ on a key ("+o+") after removal (iterating
on t)");
   ensure(m.keySet().contains(o), "Error (" + level + ", " + seed + "): m and t differ on a key ("+o+", in keySet()) after
removal (iterating on t)");
  }

  /* Now we check that m actually holds the same keys, but iterating on m. */

  for(java.util.Iterator i=m.keySet().iterator(); i.hasNext();) {
   Object o = i.next();
   ensure(t.containsKey(o), "Error (" + level + ", " + seed + "): m and t differ on a key after removal (iterating on m)");
   ensure(t.keySet().contains(o), "Error (" + level + ", " + seed + "): m and t differ on a key (in keySet()) after removal
(iterating on m)");
  }


  /* Now we check that m actually hold the same values. */

  for(java.util.Iterator i=t.values().iterator(); i.hasNext();) {
   Object o = i.next();
   ensure(m.containsValue(o), "Error (" + level + ", " + seed + "): m and t differ on a value after removal (iterating on
```

```
t)");
  ensure(m.values().contains(o), "Error (" + level + ", " + seed + "): m and t differ on a value (in values()) after removal (iterating on t)");
  }

  /* Now we check that m actually hold the same values, but iterating on m. */

  for(java.util.Iterator i=m.values().iterator(); i.hasNext();) {
   Object o = i.next();
   ensure(t.containsValue(o), "Error (" + level + ", " + seed + "): m and t differ on a value after removal (iterating on m)");
   ensure(t.values().contains(o), "Error (" + level + ", " + seed + "): m and t differ on a value (in values()) after removal (iterating on m)");
  }

  /* Now we check that both maps agree on first/last keys. */

  mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

  try {
   m.firstKey();
  }
  catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
  try {
   t.firstKey();
  }
  catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): firstKey() divergence in java.util.NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
  if (! mThrowsNoElement) ensure(t.firstKey().equals(m.firstKey()), "Error (" + level + ", " + seed + "): m and t differ on their first key (" + m.firstKey() + ", " + t.firstKey() +")");

  mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

  try {
   m.lastKey();
  }
  catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
  try {
   t.lastKey();
  }
  catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): lastKey() divergence in java.util.NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");

  if (! mThrowsNoElement) ensure(t.lastKey().equals(m.lastKey()), "Error (" + level + ", " + seed + "): m and t differ
```

```
on their last key (" + m.lastKey() + ", " + t.lastKey() +")");

  /* Now we check cloning. */

  if (level == 0) {
   ensure(m.equals(((Singleton)m).clone()), "Error (" + level + ", " + seed + "): m does not equal m.clone()");
   ensure(((Singleton)m).clone().equals(m), "Error (" + level + ", " + seed + "): m.clone() does not equal m");
   m = (SORTED_MAP)((Singleton)m).clone();
  }

  int h = m.hashCode();


  /* Now we save and read m. */

  SORTED_MAP m2 = null;

  try {
   java.io.File ff = new java.io.File("it.unimi.dsi.fastutil.test");
   java.io.OutputStream os = new java.io.FileOutputStream(ff);
   java.io.ObjectOutputStream oos = new java.io.ObjectOutputStream(os);

   oos.writeObject(m);
   oos.close();

   java.io.InputStream is = new java.io.FileInputStream(ff);
   java.io.ObjectInputStream ois = new java.io.ObjectInputStream(is);

   m2 = (SORTED_MAP)ois.readObject();
   ois.close();
   ff.delete();
  }
  catch(Exception e) {
   e.printStackTrace();
   System.exit(1);
  }

#if !VALUE_CLASS_Reference
  ensure(m2.hashCode() == h, "Error (" + level + ", " + seed + "): hashCode() changed after save/read");

  /* Now we check that m2 actually holds that data. */

  ensure(m2.equals(t), "Error (" + level + ", " + seed + "): ! m2.equals(t) after save/read");
  ensure(t.equals(m2), "Error (" + level + ", " + seed + "): ! t.equals(m2) after save/read");
  /* Now we take out of m everything, and check that it is empty. */
#endif

  /* Now we play with iterators. */
```

```
 {
  java.util.ListIterator i, j;
  Object J;
  i = (java.util.ListIterator)m.entrySet().iterator();
  j = new java.util.LinkedList(t.entrySet()).listIterator();

  for(int k = 0; k < 2*n; k++) {
   ensure(i.hasNext() == j.hasNext(), "Error (" + level + ", " + seed + "): divergence in hasNext()");
   ensure(i.hasPrevious() == j.hasPrevious(), "Error (" + level + ", " + seed + "): divergence in hasPrevious()");

   if (r.nextFloat() < .8 && i.hasNext()) {
    ensure(((java.util.Map.Entry)i.next()).getKey().equals(J = ((Map.Entry)j.next()).getKey()), "Error (" + level + ", " +
seed + "): divergence in next()");

   }
   else if (r.nextFloat() < .2 && i.hasPrevious()) {
    ensure(((java.util.Map.Entry)i.previous()).getKey().equals(J = ((Map.Entry)j.previous()).getKey()), "Error (" + level
+ ", " + seed + "): divergence in previous()");

   }

   ensure(i.nextIndex() == j.nextIndex(), "Error (" + level + ", " + seed + "): divergence in nextIndex()");
   ensure(i.previousIndex() == j.previousIndex(), "Error (" + level + ", " + seed + "): divergence in previousIndex()");

  }
 }

 {
  boolean badPrevious = false;
  Object previous = null;
  it.unimi.dsi.fastutil.BidirectionalIterator i;
  java.util.ListIterator j;
  Object I, J;
  KEY_TYPE from = genKey();
  j = new java.util.LinkedList(t.keySet()).listIterator();
  while(j.hasNext()) {
   Object k = j.next();
   if (((Comparable)k).compareTo(KEY2OBJ(from)) > 0) {
    badPrevious = true;
    j.previous();
    break;
   }
   previous = k;
  }

  i = (it.unimi.dsi.fastutil.BidirectionalIterator)((SORTED_SET)m.keySet()).iterator(from);
```

```
  for(int k = 0; k < 2*n; k++) {
   ensure(i.hasNext() == j.hasNext(), "Error (" + level + ", " + seed + "): divergence in hasNext() (iterator with starting point " + from + ")");
   ensure(i.hasPrevious() == j.hasPrevious() || badPrevious && (i.hasPrevious() == (previous != null)), "Error (" + level + ", " + seed + "): divergence in hasPrevious() (iterator with starting point " + from + ")" + badPrevious);

   if (r.nextFloat() < .8 && i.hasNext()) {
    ensure((I = i.next()).equals(J = j.next()), "Error (" + level + ", " + seed + "): divergence in next() (" + I + ", " + J + ", iterator with starting point " + from + ")");
    //System.err.println("Done next " + I + " " + J + "  " + badPrevious);

    badPrevious = false;

    if (r.nextFloat() < 0.5) {
    }
   }
   else if (!badPrevious && r.nextFloat() < .2 && i.hasPrevious()) {
    ensure((I = i.previous()).equals(J = j.previous()), "Error (" + level + ", " + seed + "): divergence in previous() (" + I + ", " + J + ", iterator with starting point " + from + ")");

    if (r.nextFloat() < 0.5) {
    }
   }
  }

 }

 /* Now we check that m actually holds that data. */

 ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after iteration");
 ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after iteration");

 /* Now we select a pair of keys and create a submap. */

 if (! m.isEmpty()) {
  java.util.ListIterator i;
  Object start = m.firstKey(), end = m.firstKey();
  for(i = (java.util.ListIterator)m.keySet().iterator(); i.hasNext() && r.nextFloat() < .3; start = end = i.next());
  for(; i.hasNext() && r.nextFloat() < .95; end = i.next());

  //System.err.println("Checking subMap from " + start + " to " + end + " (level=" + (level+1) + ")...");
  testMaps((SORTED_MAP)m.subMap((KEY_CLASS)start, (KEY_CLASS)end), t.subMap(start, end), n, level + 1);

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after subMap");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after subMap");

  //System.err.println("Checking headMap to " + end + " (level=" + (level+1) + ")...");
  testMaps((SORTED_MAP)m.headMap((KEY_CLASS)end), t.headMap(end), n, level + 1);
```

```
  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after headMap");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after headMap");

  //System.err.println("Checking tailMap from " + start + " (level=" + (level+1) + ")...");
  testMaps((SORTED_MAP)m.tailMap((KEY_CLASS)start), t.tailMap(start), n, level + 1);

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after tailMap");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after tailMap");
  }


 }


 private static void test() {
  int n = 1;
  k = new Object[n];
  v = new Object[n];
  nk = new Object[n];
  kt = new KEY_TYPE[n];
  nkt = new KEY_TYPE[n];
  vt = new VALUE_TYPE[n];


  for(int i = 0; i < n; i++) {
#if KEY_CLASS_Object
   k[i] = kt[i] = genKey();
   nk[i] = nkt[i] = genKey();
#else
   k[i] = new KEY_CLASS(kt[i] = genKey());
   nk[i] = new KEY_CLASS(nkt[i] = genKey());
#endif
#if VALUES_REFERENCE
   v[i] = vt[i] = genValue();
#else
   v[i] = new VALUE_CLASS(vt[i] = genValue());
#endif
  }

  SORTED_MAP m = new Singleton(kt[0], vt[0]);
  topMap = m;
  SortedMap t1 = new java.util.TreeMap();
  t1.put(k[0], v[0]);
  SortedMap t = java.util.Collections.unmodifiableSortedMap(t1);

  testMaps(m, t, n, 0);
```

```
 System.out.println("Test OK");
 return;
 }


 public static void main(String args[]) {
  if (args.length > 1) r = new java.util.Random(seed = Long.parseLong(args[1]));

  try {
   test();
  } catch(Throwable e) {
   e.printStackTrace(System.err);
   System.err.println("seed: " + seed);
  }
 }

#endif


}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/SortedMaps.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import java.util.Comparator;

/** A type-specific {@link Comparator}; provides methods to compare two primitive types both as objects
```

```
* and as primitive types.
*
* <p>Note that {@code fastutil} provides a corresponding abstract class that
* can be used to implement this interface just by specifying the type-specific
* comparator.
*
* @see Comparator
*/

@FunctionalInterface
public interface KEY_COMPARATOR KEY_GENERIC extends Comparator<KEY_GENERIC_CLASS> {

 /** Compares its two primitive-type arguments for order. Returns a negative integer,
  * zero, or a positive integer as the first argument is less than, equal
  * to, or greater than the second.
  *
  * @see java.util.Comparator
  * @return a negative integer, zero, or a positive integer as the first
  * argument is less than, equal to, or greater than the second.
  */

 int compare(KEY_TYPE k1, KEY_TYPE k2);

#if KEYS_PRIMITIVE
 /** {@inheritDoc}
  * <p>This implementation delegates to the corresponding type-specific method.
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 default int compare(KEY_GENERIC_CLASS ok1, KEY_GENERIC_CLASS ok2) {
  return compare(ok1.KEY_VALUE(), ok2.KEY_VALUE());
 }
#endif

}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/Comparator.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
```

```
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

/**  An abstract class facilitating the creation of type-specific {@linkplain it.unimi.dsi.fastutil.BidirectionalIterator bidirectional iterators}.
*
* @deprecated As of fastutil 8 this class is no longer necessary, as its previous abstract
* methods are now default methods of the type-specific interface.
*/

@Deprecated
public abstract class KEY_ABSTRACT_BIDI_ITERATOR KEY_GENERIC extends KEY_ABSTRACT_ITERATOR KEY_GENERIC implements KEY_BIDI_ITERATOR KEY_GENERIC {
 protected KEY_ABSTRACT_BIDI_ITERATOR() {}
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/AbstractBidirectionalIterator.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;
```

```
import java.util.Collection;
import java.util.Comparator;
import java.util.Iterator;
import java.util.SortedSet;
import java.util.NoSuchElementException;

/** A type-specific AVL tree set with a fast, small-footprint implementation.
*
* <p>The iterators provided by this class are type-specific {@link
* it.unimi.dsi.fastutil.BidirectionalIterator bidirectional iterators}.
* Moreover, the iterator returned by {@code iterator()} can be safely cast
* to a type-specific {@linkplain java.util.ListIterator list iterator}.
*/

public class AVL_TREE_SET KEY_GENERIC extends ABSTRACT_SORTED_SET KEY_GENERIC
implements java.io.Serializable, Cloneable, SORTED_SET KEY_GENERIC {

 /** A reference to the root entry. */
 protected transient Entry KEY_GENERIC tree;

 /** Number of elements in this set. */
 protected int count;

 /** The entry of the first element of this set. */
 protected transient Entry KEY_GENERIC firstEntry;

 /** The entry of the last element of this set. */
 protected transient Entry KEY_GENERIC lastEntry;

 /** This set's comparator, as provided in the constructor. */
 protected Comparator<? super KEY_GENERIC_CLASS> storedComparator;

 /** This set's actual comparator; it may differ from {@link #storedComparator} because it is
  always a type-specific comparator, so it could be derived from the former by wrapping. */
 protected transient KEY_COMPARATOR KEY_SUPER_GENERIC actualComparator;

 private static final long serialVersionUID = -7046029254386353130L;

 {
  allocatePaths();
 }

 /** Creates a new empty tree set.
  */

 public AVL_TREE_SET() {
  tree = null;
```

```
 count = 0;
 }

 /** Generates the comparator that will be actually used.
  *
  * <p>When a given {@link Comparator} is specified and stored in {@link
  * #storedComparator}, we must check whether it is type-specific.  If it is
  * so, we can used directly, and we store it in {@link #actualComparator}. Otherwise,
  * we adapt it using a helper static method.
  */
 private void setActualComparator() {
#if KEY_CLASS_Object
  actualComparator = storedComparator;
#else
  actualComparator = COMPARATORS.AS_KEY_COMPARATOR(storedComparator);
#endif
 }


 /** Creates a new empty tree set with the given comparator.
  *
  * @param c a {@link Comparator} (even better, a type-specific comparator).
  */

 public AVL_TREE_SET(final Comparator<? super KEY_GENERIC_CLASS> c) {
  this();
  storedComparator = c;
  setActualComparator();
 }


 /** Creates a new tree set copying a given set.
  *
  * @param c a collection to be copied into the new tree set.
  */

 public AVL_TREE_SET(final Collection<? extends KEY_GENERIC_CLASS> c) {
  this();
  addAll(c);
 }

 /** Creates a new tree set copying a given sorted set (and its {@link Comparator}).
  *
  * @param s a {@link SortedSet} to be copied into the new tree set.
  */

 public AVL_TREE_SET(final SortedSet <KEY_GENERIC_CLASS> s) {
  this(s.comparator());
```

```
 addAll(s);
}

/** Creates a new tree set copying a given type-specific collection.
 *
 * @param c a type-specific collection to be copied into the new tree set.
 */

public AVL_TREE_SET(final COLLECTION KEY_EXTENDS_GENERIC c) {
 this();
 addAll(c);
}

/** Creates a new tree set copying a given type-specific sorted set (and its {@link Comparator}).
 *
 * @param s a type-specific sorted set to be copied into the new tree set.
 */

public AVL_TREE_SET(final SORTED_SET KEY_GENERIC s) {
 this(s.comparator());
 addAll(s);
}


/** Creates a new tree set using elements provided by a type-specific iterator.
 *
 * @param i a type-specific iterator whose elements will fill the set.
 */

public AVL_TREE_SET(final STD_KEY_ITERATOR KEY_EXTENDS_GENERIC i) {
 while(i.hasNext()) add(i.NEXT_KEY());
}

#if KEYS_PRIMITIVE

/** Creates a new tree set using elements provided by an iterator.
 *
 * @param i an iterator whose elements will fill the set.
 */

SUPPRESS_WARNINGS_KEY_UNCHECKED
public AVL_TREE_SET(final Iterator<?> i) {
 this(ITERATORS.AS_KEY_ITERATOR(i));
}

#endif

/** Creates a new tree set and fills it with the elements of a given array using a given {@link Comparator}.
```

```
 *
 * @param a an array whose elements will be used to fill the set.
 * @param offset the first element to use.
 * @param length the number of elements to use.
 * @param c a {@link Comparator} (even better, a type-specific comparator).
 */

 public AVL_TREE_SET(final KEY_GENERIC_TYPE[] a, final int offset, final int length, final Comparator<?
super KEY_GENERIC_CLASS> c) {
 this(c);
 ARRAYS.ensureOffsetLength(a, offset, length);
 for(int i = 0; i < length; i++) add(a[offset + i]);
 }


 /** Creates a new tree set and fills it with the elements of a given array.
 *
 * @param a an array whose elements will be used to fill the set.
 * @param offset the first element to use.
 * @param length the number of elements to use.
 */

 public AVL_TREE_SET(final KEY_GENERIC_TYPE[] a, final int offset, final int length) {
 this(a, offset, length, null);
 }


 /** Creates a new tree set copying the elements of an array.
 *
 * @param a an array to be copied into the new tree set.
 */

 public AVL_TREE_SET(final KEY_GENERIC_TYPE[] a) {
 this();
 int i = a.length;
 while(i-- != 0) add(a[i]);
 }


 /** Creates a new tree set copying the elements of an array using a given {@link Comparator}.
 *
 * @param a an array to be copied into the new tree set.
 * @param c a {@link Comparator} (even better, a type-specific comparator).
 */

 public AVL_TREE_SET(final KEY_GENERIC_TYPE[] a, final Comparator<? super KEY_GENERIC_CLASS>
c) {
 this(c);
```

```
 int i = a.length;
 while(i-- != 0) add(a[i]);
}



/*
 * The following methods implements some basic building blocks used by
 * all accessors.  They are (and should be maintained) identical to those used in AVLTreeMap.drv.
 *
 * The add()/remove() code is derived from Ben Pfaff's GNU libavl
 * (http://www.msu.edu/~pfaffben/avl/). If you want to understand what's
 * going on, you should have a look at the literate code contained therein
 * first.
 */



/** Compares two keys in the right way.
 *
 * <p>This method uses the {@link #actualComparator} if it is non-{@code null}.
 * Otherwise, it resorts to primitive type comparisons or to {@link Comparable#compareTo(Object) compareTo()}.
 *
 * @param k1 the first key.
 * @param k2 the second key.
 * @return a number smaller than, equal to or greater than 0, as usual
 * (i.e., when k1 &lt; k2, k1 = k2 or k1 &gt; k2, respectively).
 */

SUPPRESS_WARNINGS_KEY_UNCHECKED
final int compare(final KEY_GENERIC_TYPE k1, final KEY_GENERIC_TYPE k2) {
 return actualComparator == null ? KEY_CMP(k1, k2) : actualComparator.compare(k1, k2);
}



/** Returns the entry corresponding to the given key, if it is in the tree; {@code null}, otherwise.
 *
 * @param k the key to search for.
 * @return the corresponding entry, or {@code null} if no entry with the given key exists.
 */

private Entry KEY_GENERIC findKey(final KEY_GENERIC_TYPE k) {
 Entry KEY_GENERIC e = tree;
 int cmp;

 while (e != null && (cmp = compare(k, e.key)) != 0)
  e = cmp < 0 ? e.left() : e.right();
```

```
 return e;
}

/** Locates a key.
 *
 * @param k a key.
 * @return the last entry on a search for the given key; this will be
 * the given key, if it present; otherwise, it will be either the smallest greater key or the greatest smaller key.
 */

final Entry KEY_GENERIC locateKey(final KEY_GENERIC_TYPE k) {
 Entry KEY_GENERIC e = tree, last = tree;
 int cmp = 0;

 while (e != null && (cmp = compare(k, e.key)) != 0) {
  last = e;
  e = cmp < 0 ? e.left() : e.right();
 }

 return cmp == 0 ? e : last;
}


/** This vector remembers the path followed during the current insertion. It suffices for
 about 2<sup>32</sup> entries. */
private transient boolean dirPath[];

private void allocatePaths() {
 dirPath = new boolean[48];
}


@Override
public boolean add(final KEY_GENERIC_TYPE k) {

 if (tree == null) { // The case of the empty tree is treated separately.
  count++;
  tree = lastEntry = firstEntry = new Entry KEY_GENERIC_DIAMOND(k);
 }
 else {
  Entry KEY_GENERIC p = tree, q = null, y = tree, z = null, e = null, w = null;
  int cmp, i = 0;

  while(true) {
   if ((cmp = compare(k, p.key)) == 0) return false;

   if (p.balance() != 0) {
```

```
 i = 0;
 z = q;
 y = p;
}

if (dirPath[i++] = cmp > 0) {
 if (p.succ()) {
  count++;
  e = new Entry KEY_GENERIC_DIAMOND(k);

  if (p.right == null) lastEntry = e;

  e.left = p;
  e.right = p.right;

  p.right(e);

  break;
 }

 q = p;
 p = p.right;
}
else {
 if (p.pred()) {
  count++;
  e = new Entry KEY_GENERIC_DIAMOND(k);

  if (p.left == null) firstEntry = e;

  e.right = p;
  e.left = p.left;

  p.left(e);

  break;
 }

 q = p;
 p = p.left;
 }
}

p = y;
i = 0;

while(p != e) {
 if (dirPath[i]) p.incBalance();
```

```
 else p.decBalance();

 p = dirPath[i++] ? p.right : p.left;
}

if (y.balance() == -2) {
 Entry KEY_GENERIC x = y.left;

 if (x.balance() == -1) {
  w = x;
  if (x.succ()) {
   x.succ(false);
   y.pred(x);
  }
  else y.left = x.right;

  x.right = y;
  x.balance(0);
  y.balance(0);
 }
 else {
  assert x.balance() == 1;

  w = x.right;
  x.right = w.left;
  w.left = x;
  y.left = w.right;
  w.right = y;
  if (w.balance() == -1) {
   x.balance(0);
   y.balance(1);
  }
  else if (w.balance() == 0) {
   x.balance(0);
   y.balance(0);
  }
  else {
   x.balance(-1);
   y.balance(0);
  }
  w.balance(0);


  if (w.pred()) {
   x.succ(w);
   w.pred(false);
  }
  if (w.succ()) {
```

```
  y.pred(w);
  w.succ(false);
  }

 }
}
else if (y.balance() == +2) {
 Entry KEY_GENERIC x = y.right;

 if (x.balance() == 1) {
  w = x;
  if (x.pred()) {
   x.pred(false);
   y.succ(x);
  }
  else y.right = x.left;

  x.left = y;
  x.balance(0);
  y.balance(0);
 }
 else {
  assert x.balance() == -1;

  w = x.left;
  x.left = w.right;
  w.right = x;
  y.right = w.left;
  w.left = y;
  if (w.balance() == 1) {
   x.balance(0);
   y.balance(-1);
  }
  else if (w.balance() == 0) {
   x.balance(0);
   y.balance(0);
  }
  else {
   x.balance(1);
   y.balance(0);
  }
  w.balance(0);


  if (w.pred()) {
   y.succ(w);
   w.pred(false);
  }
```

```
   if (w.succ()) {
    x.pred(w);
    w.succ(false);
   }

  }
 }
 else return true;

 if (z == null) tree = w;
 else {
  if (z.left == y) z.left = w;
  else z.right = w;
 }
}

 return true;
}




/** Finds the parent of an entry.
 *
 * @param e a node of the tree.
 * @return the parent of the given node, or {@code null} for the root.
 */

private Entry KEY_GENERIC parent(final Entry KEY_GENERIC e) {
 if (e == tree) return null;

 Entry KEY_GENERIC x, y, p;
 x = y = e;

 while(true) {
  if (y.succ()) {
   p = y.right;
   if (p == null || p.left != e) {
    while(! x.pred()) x = x.left;
    p = x.left;
   }
   return p;
  }
  else if (x.pred()) {
   p = x.left;
   if (p == null || p.right != e) {
    while(! y.succ()) y = y.right;
    p = y.right;
```

```
  }
  return p;
  }

  x = x.left;
  y = y.right;
 }
}


SUPPRESS_WARNINGS_KEY_UNCHECKED
@Override
public boolean remove(final KEY_TYPE k) {
 if (tree == null) return false;

 int cmp;
 Entry KEY_GENERIC p = tree, q = null;
 boolean dir = false;
 final KEY_GENERIC_TYPE kk = KEY_GENERIC_CAST k;

 while(true) {
  if ((cmp = compare(kk, p.key)) == 0) break;
  else if (dir = cmp > 0) {
   q = p;
   if ((p = p.right()) == null) return false;
  }
  else {
   q = p;
   if ((p = p.left()) == null) return false;
  }
 }

 if (p.left == null) firstEntry = p.next();
 if (p.right == null) lastEntry = p.prev();

 if (p.succ()) {
  if (p.pred()) {
   if (q != null) {
    if (dir) q.succ(p.right);
    else q.pred(p.left);
   }
   else tree = dir ? p.right : p.left;
  }
  else {
   p.prev().right = p.right;

   if (q != null) {
    if (dir) q.right = p.left;
```

```
  else q.left = p.left;
 }
 else tree = p.left;
}
}
else {
 Entry KEY_GENERIC r = p.right;

 if (r.pred()) {
  r.left = p.left;
  r.pred(p.pred());
  if (! r.pred()) r.prev().right = r;
  if (q != null) {
   if (dir) q.right = r;
   else q.left = r;
  }
  else tree = r;

  r.balance(p.balance());
  q = r;
  dir = true;

 }
 else {
  Entry KEY_GENERIC s;

  while(true) {
   s = r.left;
   if (s.pred()) break;
   r = s;
  }

  if (s.succ()) r.pred(s);
  else r.left = s.right;

  s.left = p.left;

  if (! p.pred()) {
   p.prev().right = s;
   s.pred(false);
  }

  s.right = p.right;
  s.succ(false);

  if (q != null) {
   if (dir) q.right = s;
   else q.left = s;
```

```
 }
 else tree = s;

 s.balance(p.balance());
 q = r;
 dir = false;
 }
}

Entry KEY_GENERIC y;

while(q != null) {
 y = q;
 q = parent(y);

 if (! dir) {
  dir = q != null && q.left != y;
  y.incBalance();

  if (y.balance() == 1) break;
  else if (y.balance() == 2) {

   Entry KEY_GENERIC x = y.right;
   assert x != null;

   if (x.balance() == -1) {
    Entry KEY_GENERIC w;

    assert x.balance() == -1;

    w = x.left;
    x.left = w.right;
    w.right = x;
    y.right = w.left;
    w.left = y;

    if (w.balance() == 1) {
     x.balance(0);
     y.balance(-1);
    }
    else if (w.balance() == 0) {
     x.balance(0);
     y.balance(0);
    }
    else {
     assert w.balance() == -1;

     x.balance(1);
```

```
 y.balance(0);
 }

 w.balance(0);

 if (w.pred()) {
  y.succ(w);
  w.pred(false);
 }
 if (w.succ()) {
  x.pred(w);
  w.succ(false);
 }

 if (q != null) {
  if (dir) q.right = w;
  else q.left = w;
 }
 else tree = w;
}
else {
 if (q != null) {
  if (dir) q.right = x;
  else q.left = x;
 }
 else tree = x;

 if (x.balance() == 0) {
  y.right = x.left;
  x.left = y;
  x.balance(-1);
  y.balance(+1);
  break;
 }

 assert x.balance() == 1;

 if (x.pred()) {
  y.succ(true);
  x.pred(false);
 }
 else y.right = x.left;

 x.left = y;
 y.balance(0);
 x.balance(0);
 }
}
```

```
}
else {
 dir = q != null && q.left != y;
 y.decBalance();

 if (y.balance() == -1) break;
 else if (y.balance() == -2) {

  Entry KEY_GENERIC x = y.left;
  assert x != null;

  if (x.balance() == 1) {
   Entry KEY_GENERIC w;

   assert x.balance() == 1;

   w = x.right;
   x.right = w.left;
   w.left = x;
   y.left = w.right;
   w.right = y;

   if (w.balance() == -1) {
    x.balance(0);
    y.balance(1);
   }
   else if (w.balance() == 0) {
    x.balance(0);
    y.balance(0);
   }
   else {
    assert w.balance() == 1;

    x.balance(-1);
    y.balance(0);
   }

   w.balance(0);

   if (w.pred()) {
    x.succ(w);
    w.pred(false);
   }
   if (w.succ()) {
    y.pred(w);
    w.succ(false);
   }
```

```
    if (q != null) {
     if (dir) q.right = w;
     else q.left = w;
    }
    else tree = w;
   }
   else {
    if (q != null) {
     if (dir) q.right = x;
     else q.left = x;
    }
    else tree = x;

    if (x.balance() == 0) {
     y.left = x.right;
     x.right = y;
     x.balance(+1);
     y.balance(-1);
     break;
    }

    assert x.balance() == -1;

    if (x.succ()) {
     y.pred(true);
     x.succ(false);
    }
    else y.left = x.right;

    x.right = y;
    y.balance(0);
    x.balance(0);
   }
  }
 }

 count--;
 return true;
}

SUPPRESS_WARNINGS_KEY_UNCHECKED
@Override
public boolean contains(final KEY_TYPE k) {
 return findKey(KEY_GENERIC_CAST k) != null;
}

#if KEY_CLASS_Object
```

```
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public K get(final KEY_TYPE k) {
  final Entry KEY_GENERIC entry = findKey(KEY_GENERIC_CAST k);
  return entry == null ? null : entry.key;
 }
#endif

 @Override
 public void clear() {
  count = 0;
  tree = null;
  firstEntry = lastEntry = null;
 }


 /** This class represent an entry in a tree set.
  *
  * <p>We use the only "metadata", i.e., {@link Entry#info}, to store
  * information about balance, predecessor status and successor status.
  *
  * <p>Note that since the class is recursive, it can be
  * considered equivalently a tree.
  */

 private static final class Entry KEY_GENERIC implements Cloneable {
  /** If the bit in this mask is true, {@link #right} points to a successor. */
  private static final int SUCC_MASK = 1 << 31;
  /** If the bit in this mask is true, {@link #left} points to a predecessor. */
  private static final int PRED_MASK = 1 << 30;
  /** The bits in this mask hold the node balance info. You can get it just by casting to byte. */
  private static final int BALANCE_MASK = 0xFF;
  /** The key of this entry. */
  KEY_GENERIC_TYPE key;
  /** The pointers to the left and right subtrees. */
  Entry KEY_GENERIC left, right;
  /** This integers holds different information in different bits (see {@link #SUCC_MASK}, {@link
#PRED_MASK} and {@link #BALANCE_MASK}). */
  int info;

  Entry() {}

  /** Creates a new entry with the given key.
   *
   * @param k a key.
   */
  Entry(final KEY_GENERIC_TYPE k) {
   this.key = k;
   info = SUCC_MASK | PRED_MASK;
```

```
}

/** Returns the left subtree.
 *
 * @return the left subtree ({@code null} if the left
 * subtree is empty).
 */
Entry KEY_GENERIC left() {
 return (info & PRED_MASK) != 0 ? null : left;
}

/** Returns the right subtree.
 *
 * @return the right subtree ({@code null} if the right
 * subtree is empty).
 */
Entry KEY_GENERIC right() {
 return (info & SUCC_MASK) != 0 ? null : right;
}

/** Checks whether the left pointer is really a predecessor.
 * @return true if the left pointer is a predecessor.
 */
boolean pred() {
 return (info & PRED_MASK) != 0;
}

/** Checks whether the right pointer is really a successor.
 * @return true if the right pointer is a successor.
 */
boolean succ() {
 return (info & SUCC_MASK) != 0;
}

/** Sets whether the left pointer is really a predecessor.
 * @param pred if true then the left pointer will be considered a predecessor.
 */
void pred(final boolean pred) {
 if (pred) info |= PRED_MASK;
 else info &= ~PRED_MASK;
}

/** Sets whether the right pointer is really a successor.
 * @param succ if true then the right pointer will be considered a successor.
 */
void succ(final boolean succ) {
 if (succ) info |= SUCC_MASK;
 else info &= ~SUCC_MASK;
```

```
}

/** Sets the left pointer to a predecessor.
 * @param pred the predecessr.
 */
void pred(final Entry KEY_GENERIC pred) {
 info |= PRED_MASK;
 left = pred;
}

/** Sets the right pointer to a successor.
 * @param succ the successor.
 */
void succ(final Entry KEY_GENERIC succ) {
 info |= SUCC_MASK;
 right = succ;
}

/** Sets the left pointer to the given subtree.
 * @param left the new left subtree.
 */
void left(final Entry KEY_GENERIC left) {
 info &= ~PRED_MASK;
 this.left = left;
}

/** Sets the right pointer to the given subtree.
 * @param right the new right subtree.
 */
void right(final Entry KEY_GENERIC right) {
 info &= ~SUCC_MASK;
 this.right = right;
}

/** Returns the current level of the node.
 * @return the current level of this node.
 */
int balance() {
 return (byte)info;
}

/** Sets the level of this node.
 * @param level the new level of this node.
 */
void balance(int level) {
 info &= ~BALANCE_MASK;
 info |= (level & BALANCE_MASK);
}
```

```
/** Increments the level of this node. */
void incBalance() {
 info = info & ~BALANCE_MASK | ((byte)info + 1) & 0xFF;
}

/** Decrements the level of this node. */
protected void decBalance() {
 info = info & ~BALANCE_MASK | ((byte)info - 1) & 0xFF;
}

/** Computes the next entry in the set order.
 *
 * @return the next entry ({@code null}) if this is the last entry).
 */

Entry KEY_GENERIC next() {
 Entry KEY_GENERIC next = this.right;
 if ((info & SUCC_MASK) == 0) while ((next.info & PRED_MASK) == 0) next = next.left;
 return next;
}

/** Computes the previous entry in the set order.
 *
 * @return the previous entry ({@code null}) if this is the first entry).
 */

Entry KEY_GENERIC prev() {
 Entry KEY_GENERIC prev = this.left;
 if ((info & PRED_MASK) == 0) while ((prev.info & SUCC_MASK) == 0) prev = prev.right;
 return prev;
}

@Override
SUPPRESS_WARNINGS_KEY_UNCHECKED
public Entry KEY_GENERIC clone() {
 Entry KEY_GENERIC c;
 try {
  c = (Entry KEY_GENERIC)super.clone();
 }
 catch(CloneNotSupportedException cantHappen) {
  throw new InternalError();
 }

 c.key = key;
 c.info = info;

 return c;
```

```
 }

 public boolean equals(final Object o) {
  if (!(o instanceof Entry)) return false;
  Entry KEY_GENERIC_WILDCARD e = (Entry KEY_GENERIC_WILDCARD)o;

  return KEY_EQUALS(key, e.key);
 }

 public int hashCode() {
  return KEY2JAVAHASH_NOT_NULL(key);
 }


 public String toString() {
  return String.valueOf(key);
 }

 /*
   public void prettyPrint() {
   prettyPrint(0);
   }

   public void prettyPrint(int level) {
   if (pred()) {
   for (int i = 0; i < level; i++)
   System.err.print("  ");
   System.err.println("pred: " + left);
   }
   else if (left != null)
   left.prettyPrint(level +1);
   for (int i = 0; i < level; i++)
   System.err.print("  ");
   System.err.println(key + " (" + level() + ")");
   if (succ()) {
   for (int i = 0; i < level; i++)
   System.err.print("  ");
   System.err.println("succ: " + right);
   }
   else if (right != null)
   right.prettyPrint(level + 1);
   }
 */
}

/*
  public void prettyPrint() {
  System.err.println("size: " + count);
```

```
  if (tree != null) tree.prettyPrint();
  }
*/

@Override
public int size() {
 return count;
}

@Override
public boolean isEmpty() {
 return count == 0;
}

@Override
public KEY_GENERIC_TYPE FIRST() {
 if (tree == null) throw new NoSuchElementException();
 return firstEntry.key;
}

@Override
public KEY_GENERIC_TYPE LAST() {
 if (tree == null) throw new NoSuchElementException();
 return lastEntry.key;
}


/** An iterator on the whole range.
 *
 * <p>This class can iterate in both directions on a threaded tree.
 */

private class SetIterator implements KEY_LIST_ITERATOR KEY_GENERIC {
 /** The entry that will be returned by the next call to {@link java.util.ListIterator#previous()} (or {@code null} if
no previous entry exists). */
 Entry KEY_GENERIC prev;
 /** The entry that will be returned by the next call to {@link java.util.ListIterator#next()} (or {@code null} if no
next entry exists). */
 Entry KEY_GENERIC next;
 /** The last entry that was returned (or {@code null} if we did not iterate or used {@link #remove()}). */
 Entry KEY_GENERIC curr;
 /** The current index (in the sense of a {@link java.util.ListIterator}). Note that this value is not meaningful when
this {@link SetIterator} has been created using the nonempty constructor.*/
 int index = 0;

 SetIterator() {
  next = firstEntry;
 }
```

```
SetIterator(final KEY_GENERIC_TYPE k) {
 if ((next = locateKey(k)) != null) {
  if (compare(next.key, k) <= 0) {
   prev = next;
   next = next.next();
  }
  else prev = next.prev();
 }
}

@Override
public boolean hasNext() { return next != null; }
@Override
public boolean hasPrevious() { return prev != null; }

void updateNext() { next = next.next(); }

Entry KEY_GENERIC nextEntry() {
 if (! hasNext()) throw new NoSuchElementException();
 curr = prev = next;
 index++;
 updateNext();
 return curr;
}

@Override
public KEY_GENERIC_TYPE NEXT_KEY() { return nextEntry().key; }
@Override
public KEY_GENERIC_TYPE PREV_KEY() { return previousEntry().key; }

void updatePrevious() { prev = prev.prev(); }

Entry KEY_GENERIC previousEntry() {
 if (! hasPrevious()) throw new NoSuchElementException();
 curr = next = prev;
 index--;
 updatePrevious();
 return curr;
}

@Override
public int nextIndex() { return index; }

@Override
public int previousIndex() { return index - 1; }

@Override
```

```
 public void remove() {
  if (curr == null) throw new IllegalStateException();
  /* If the last operation was a next(), we are removing an entry that preceeds
     the current index, and thus we must decrement it. */
  if (curr == prev) index--;
  next = prev = curr;
  updatePrevious();
  updateNext();
  AVL_TREE_SET.this.remove(curr.key);
  curr = null;
 }
}

@Override
public KEY_BIDI_ITERATOR KEY_GENERIC iterator() { return new SetIterator(); }

@Override
public KEY_BIDI_ITERATOR KEY_GENERIC iterator(final KEY_GENERIC_TYPE from) { return new
SetIterator(from); }

@Override
public KEY_COMPARATOR KEY_SUPER_GENERIC comparator() { return actualComparator; }

@Override
public SORTED_SET KEY_GENERIC headSet(final KEY_GENERIC_TYPE to) { return new
Subset(KEY_NULL, true, to, false); }

@Override
public SORTED_SET KEY_GENERIC tailSet(final KEY_GENERIC_TYPE from) { return new Subset(from,
false, KEY_NULL, true); }

@Override
public SORTED_SET KEY_GENERIC subSet(final KEY_GENERIC_TYPE from, final KEY_GENERIC_TYPE
to) { return new Subset(from, false, to, false); }

/** A subset with given range.
 *
 * <p>This class represents a subset. One has to specify the left/right
 * limits (which can be set to -&infin; or &infin;). Since the subset is a
 * view on the set, at a given moment it could happen that the limits of
 * the range are not any longer in the main set. Thus, things such as
 * {@link java.util.SortedSet#first()} or {@link java.util.SortedSet#size()} must be always computed
 * on-the-fly.
 */
private final class Subset extends ABSTRACT_SORTED_SET KEY_GENERIC implements java.io.Serializable,
SORTED_SET KEY_GENERIC {
 private static final long serialVersionUID = -7046029254386353129L;
```

```
 /** The start of the subset range, unless {@link #bottom} is true. */
 KEY_GENERIC_TYPE from;
 /** The end of the subset range, unless {@link #top} is true. */
 KEY_GENERIC_TYPE to;
 /** If true, the subset range starts from -&infin;. */
 boolean bottom;
 /** If true, the subset range goes to &infin;. */
 boolean top;

 /** Creates a new subset with given key range.
  *
  * @param from the start of the subset range.
  * @param bottom if true, the first parameter is ignored and the range starts from -&infin;.
  * @param to the end of the subset range.
  * @param top if true, the third parameter is ignored and the range goes to &infin;.
  */
 public Subset(final KEY_GENERIC_TYPE from, final boolean bottom, final KEY_GENERIC_TYPE to, final
boolean top) {
  if (! bottom && ! top && AVL_TREE_SET.this.compare(from, to) > 0) throw new
IllegalArgumentException("Start element (" + from  + ") is larger than end element (" + to + ")");

  this.from = from;
  this.bottom = bottom;
  this.to = to;
  this.top = top;
 }

 @Override
 public void clear() {
  final SubsetIterator i = new SubsetIterator();
  while(i.hasNext()) {
   i.NEXT_KEY();
   i.remove();
  }
 }

 /** Checks whether a key is in the subset range.
  * @param k a key.
  * @return true if is the key is in the subset range.
  */
 final boolean in(final KEY_GENERIC_TYPE k) {
  return (bottom || AVL_TREE_SET.this.compare(k, from) >= 0) &&
   (top || AVL_TREE_SET.this.compare(k, to) < 0);
 }

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 @Override
 public boolean contains(final KEY_TYPE k) {
```

```
  return in(KEY_GENERIC_CAST k) && AVL_TREE_SET.this.contains(k);
  }

  @Override
  public boolean add(final KEY_GENERIC_TYPE  k) {
  if (! in(k)) throw new IllegalArgumentException("Element (" + k + ") out of range [" + (bottom ? "-" :
String.valueOf(from)) + ", " + (top ? "-" : String.valueOf(to)) + ")");
  return AVL_TREE_SET.this.add(k);
  }

  SUPPRESS_WARNINGS_KEY_UNCHECKED
  @Override
  public boolean remove(final KEY_TYPE k) {
  if (! in(KEY_GENERIC_CAST k)) return false;
  return AVL_TREE_SET.this.remove(k);
  }

  @Override
  public int size() {
  final SubsetIterator i = new SubsetIterator();
  int n = 0;

  while(i.hasNext()) {
   n++;
   i.NEXT_KEY();
  }

  return n;
  }

  @Override
  public boolean isEmpty() { return ! new SubsetIterator().hasNext(); }

  @Override
  public KEY_COMPARATOR KEY_SUPER_GENERIC comparator() { return actualComparator; }

  @Override
  public KEY_BIDI_ITERATOR KEY_GENERIC iterator() { return new SubsetIterator(); }

  @Override
  public KEY_BIDI_ITERATOR KEY_GENERIC iterator(final KEY_GENERIC_TYPE from) { return new
SubsetIterator(from); }

  @Override
  public SORTED_SET KEY_GENERIC headSet(final KEY_GENERIC_TYPE to) {
  if (top) return new Subset(from, bottom, to, false);
  return compare(to, this.to) < 0 ? new Subset(from, bottom, to, false) : this;
  }
```

```
@Override
public SORTED_SET KEY_GENERIC tailSet(final KEY_GENERIC_TYPE from) {
 if (bottom) return new Subset(from, false, to, top);
 return compare(from, this.from) > 0 ? new Subset(from, false, to, top) : this;
}

@Override
public SORTED_SET KEY_GENERIC subSet(KEY_GENERIC_TYPE from, KEY_GENERIC_TYPE to) {
 if (top && bottom) return new Subset(from, false, to, false);
 if (! top) to = compare(to, this.to) < 0 ? to : this.to;
 if (! bottom) from = compare(from, this.from) > 0 ? from : this.from;
   if (! top && ! bottom && from == this.from && to == this.to) return this;
 return new Subset(from, false, to, false);
}

/** Locates the first entry.
 *
 * @return the first entry of this subset, or {@code null} if the subset is empty.
 */
public AVL_TREE_SET.Entry KEY_GENERIC firstEntry() {
 if (tree == null) return null;
 // If this subset goes to -infinity, we return the main set first entry; otherwise, we locate the start of the set.
 AVL_TREE_SET.Entry KEY_GENERIC e;
 if (bottom) e = firstEntry;
 else {
  e = locateKey(from);
  // If we find either the start or something greater we're OK.
  if (compare(e.key, from) < 0) e = e.next();
 }
 // Finally, if this subset doesn't go to infinity, we check that the resulting key isn't greater than the end.
 if (e == null || ! top && compare(e.key, to) >= 0) return null;
 return e;
}

/** Locates the last entry.
 *
 * @return the last entry of this subset, or {@code null} if the subset is empty.
 */
public AVL_TREE_SET.Entry KEY_GENERIC lastEntry() {
 if (tree == null) return null;
 // If this subset goes to infinity, we return the main set last entry; otherwise, we locate the end of the set.
 AVL_TREE_SET.Entry KEY_GENERIC e;
 if (top) e = lastEntry;
 else {
  e = locateKey(to);
  // If we find something smaller than the end we're OK.
  if (compare(e.key, to) >= 0) e = e.prev();
```

```
 }
 // Finally, if this subset doesn't go to -infinity, we check that the resulting key isn't smaller than the start.
 if (e == null || ! bottom && compare(e.key, from) < 0) return null;
 return e;
 }

 @Override
 public KEY_GENERIC_TYPE FIRST() {
  AVL_TREE_SET.Entry KEY_GENERIC e = firstEntry();
  if (e == null) throw new NoSuchElementException();
  return e.key;
 }

 @Override
 public KEY_GENERIC_TYPE LAST() {
  AVL_TREE_SET.Entry KEY_GENERIC e = lastEntry();
  if (e == null) throw new NoSuchElementException();
  return e.key;
 }

 /** An iterator for subranges.
  *
  * <p>This class inherits from {@link SetIterator}, but overrides the methods that
  * update the pointer after a {@link java.util.ListIterator#next()} or {@link java.util.ListIterator#previous()}. If we
would
  * move out of the range of the subset we just overwrite the next or previous
  * entry with {@code null}.
  */
 private final class SubsetIterator extends SetIterator {
  SubsetIterator() {
   next = firstEntry();
  }

  SubsetIterator(final KEY_GENERIC_TYPE k) {
   this();

   if (next != null) {
    if (! bottom && compare(k, next.key) < 0) prev = null;
    else if (! top && compare(k, (prev = lastEntry()).key) >= 0) next = null;
    else {
     next = locateKey(k);

     if (compare(next.key, k) <= 0) {
      prev = next;
      next = next.next();
     }
     else prev = next.prev();
    }
```

```
   }
  }

  @Override
  void updatePrevious() {
   prev = prev.prev();
   if (! bottom && prev != null && AVL_TREE_SET.this.compare(prev.key, from) < 0) prev = null;
  }

  @Override
  void updateNext() {
   next = next.next();
   if (! top && next != null && AVL_TREE_SET.this.compare(next.key, to) >= 0) next = null;
  }
 }
}

/** Returns a deep copy of this tree set.
 *
 * <p>This method performs a deep copy of this tree set; the data stored in the
 * set, however, is not cloned. Note that this makes a difference only for object keys.
 *
 * @return a deep copy of this tree set.
 */

@Override
SUPPRESS_WARNINGS_KEY_UNCHECKED
public Object clone() {
 AVL_TREE_SET KEY_GENERIC c;
 try {
  c = (AVL_TREE_SET KEY_GENERIC)super.clone();
 }
 catch(CloneNotSupportedException cantHappen) {
  throw new InternalError();
 }

 c.allocatePaths();

 if (count != 0) {
  // Also this apparently unfathomable code is derived from GNU libavl.
  Entry KEY_GENERIC e, p, q, rp = new Entry KEY_GENERIC_DIAMOND(), rq = new Entry
KEY_GENERIC_DIAMOND();

  p = rp;
  rp.left(tree);

  q = rq;
  rq.pred(null);
```

```
 while(true) {
  if (! p.pred()) {
   e = p.left.clone();
   e.pred(q.left);
   e.succ(q);
   q.left(e);

   p = p.left;
   q = q.left;
  }
  else {
   while(p.succ()) {
    p = p.right;

    if (p == null) {
     q.right = null;
     c.tree = rq.left;

     c.firstEntry = c.tree;
     while(c.firstEntry.left != null) c.firstEntry = c.firstEntry.left;
     c.lastEntry = c.tree;
     while(c.lastEntry.right != null) c.lastEntry = c.lastEntry.right;

     return c;
    }
    q = q.right;
   }

   p = p.right;
   q = q.right;
  }

  if (! p.succ()) {
   e = p.right.clone();
   e.succ(q.right);
   e.pred(q);
   q.right(e);
  }
 }
}

 return c;
}


private void writeObject(java.io.ObjectOutputStream s) throws java.io.IOException {
 int n = count;
```

```
 SetIterator i = new SetIterator();

 s.defaultWriteObject();
 while(n-- != 0) s.WRITE_KEY(i.NEXT_KEY());
 }


 /** Reads the given number of entries from the input stream, returning the corresponding tree.
  *
  * @param s the input stream.
  * @param n the (positive) number of entries to read.
  * @param pred the entry containing the key that preceeds the first key in the tree.
  * @param succ the entry containing the key that follows the last key in the tree.
  */
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 private Entry KEY_GENERIC readTree(final java.io.ObjectInputStream s, final int n, final Entry KEY_GENERIC
pred, final Entry KEY_GENERIC succ) throws java.io.IOException, ClassNotFoundException {
  if (n == 1) {
   final Entry KEY_GENERIC top = new Entry KEY_GENERIC_DIAMOND(KEY_GENERIC_CAST
s.READ_KEY());
   top.pred(pred);
   top.succ(succ);

   return top;
  }

  if (n == 2) {
   /* We handle separately this case so that recursion will
    *always* be on nonempty subtrees. */
   final Entry KEY_GENERIC top = new Entry KEY_GENERIC_DIAMOND(KEY_GENERIC_CAST
s.READ_KEY());
   top.right(new Entry KEY_GENERIC_DIAMOND(KEY_GENERIC_CAST s.READ_KEY()));
   top.right.pred(top);
   top.balance(1);
   top.pred(pred);
   top.right.succ(succ);

   return top;
  }

  // The right subtree is the largest one.
  final int rightN = n / 2, leftN = n - rightN - 1;

  final Entry KEY_GENERIC top = new Entry KEY_GENERIC_DIAMOND();

  top.left(readTree(s, leftN, pred, top));

  top.key = KEY_GENERIC_CAST s.READ_KEY();
```

```
 top.right(readTree(s, rightN, top, succ));

 if (n == (n & -n)) top.balance(1); // Quick test for determining whether n is a power of 2.

 return top;
}


private void readObject(java.io.ObjectInputStream s) throws java.io.IOException, ClassNotFoundException {
 s.defaultReadObject();
 /* The storedComparator is now correctly set, but we must restore
    on-the-fly the actualComparator. */
 setActualComparator();
 allocatePaths();

 if (count != 0) {
  tree = readTree(s, count, null, null);
  Entry KEY_GENERIC e;

  e = tree;
  while(e.left() != null) e = e.left();
  firstEntry = e;

  e = tree;
  while(e.right() != null) e = e.right();
  lastEntry = e;
 }
}

#ifdef ASSERTS_CODE
 private static KEY_GENERIC int checkTree(Entry KEY_GENERIC e) {
  if (e == null) return 0;

  final int leftN = checkTree(e.left()), rightN = checkTree(e.right());
  if (leftN + e.balance() != rightN)
   throw new AssertionError("Mismatch between left tree size (" + leftN + "), right tree size (" + rightN + ") and
balance (" + e.balance() + ")");

  return Math.max(leftN , rightN) + 1;
 }
#endif

#ifdef TEST

 private static long seed = System.currentTimeMillis();
 private static java.util.Random r = new java.util.Random(seed);
```

```
 private static KEY_TYPE genKey() {
#if KEY_CLASS_Byte || KEY_CLASS_Short || KEY_CLASS_Character
  return (KEY_TYPE)(r.nextInt());
#elif KEYS_PRIMITIVE
  return r.NEXT_KEY();
#else
  return Integer.toBinaryString(r.nextInt());
#endif
 }

 private static java.text.NumberFormat format = new java.text.DecimalFormat("#,###.00");
 private static java.text.FieldPosition p = new java.text.FieldPosition(0);

 private static String format(double d) {
  StringBuffer s = new StringBuffer();
  return format.format(d, s, p).toString();
 }

 private static void speedTest(int n, boolean comp) {
  int i, j;
  AVL_TREE_SET m;
  java.util.TreeSet t;
  KEY_TYPE k[] = new KEY_TYPE[n];
  KEY_TYPE nk[] = new KEY_TYPE[n];
  long ms;

  for(i = 0; i < n; i++) {
   k[i] = genKey();
   nk[i] = genKey();
  }

  double totAdd = 0, totYes = 0, totNo = 0, totIterFor = 0, totIterBack = 0, totRemYes = 0, d, dd;

  if (comp) {
   for(j = 0; j < 20; j++) {

    t = new java.util.TreeSet();

    /* We first add all pairs to t. */
    for(i = 0; i < n;  i++) t.add(KEY2OBJ(k[i]));

    /* Then we remove the first half and put it back. */
    for(i = 0; i < n/2;  i++) t.remove(KEY2OBJ(k[i]));

    ms = System.currentTimeMillis();
    for(i = 0; i < n/2;  i++) t.add(KEY2OBJ(k[i]));
    d = System.currentTimeMillis() - ms;
```

```
  /* Then we remove the other half and put it back again. */
  ms = System.currentTimeMillis();
  for(i = n/2; i < n;  i++) t.remove(KEY2OBJ(k[i]));
  dd = System.currentTimeMillis() - ms ;

  ms = System.currentTimeMillis();
  for(i = n/2; i < n;  i++) t.add(KEY2OBJ(k[i]));
  d += System.currentTimeMillis() - ms;
  if (j > 2) totAdd += n/d;
  System.out.print("Add: " + format(n/d) +" K/s ");

  /* Then we remove again the first half. */
  ms = System.currentTimeMillis();
  for(i = 0; i < n/2;  i++) t.remove(KEY2OBJ(k[i]));
  dd += System.currentTimeMillis() - ms ;
  if (j > 2) totRemYes += n/dd;
  System.out.print("RemYes: " + format(n/dd) +" K/s ");

  /* And then we put it back. */
  for(i = 0; i < n/2;  i++) t.add(KEY2OBJ(k[i]));

  /* We check for pairs in t. */
  ms = System.currentTimeMillis();
  for(i = 0; i < n;  i++) t.contains(KEY2OBJ(k[i]));
  d = 1.0 * n / (System.currentTimeMillis() - ms);
  if (j > 2) totYes += d;
  System.out.print("Yes: " + format(d) +" K/s ");

  /* We check for pairs not in t. */
  ms = System.currentTimeMillis();
  for(i = 0; i < n;  i++) t.contains(KEY2OBJ(nk[i]));
  d = 1.0 * n / (System.currentTimeMillis() - ms);
  if (j > 2) totNo += d;
  System.out.print("No: " + format(d) +" K/s ");

  /* We iterate on t. */
  ms = System.currentTimeMillis();
  for(Iterator it = t.iterator(); it.hasNext(); it.next());
  d = 1.0 * n / (System.currentTimeMillis() - ms);
  if (j > 2) totIterFor += d;
  System.out.print("IterFor: " + format(d) +" K/s ");

  System.out.println();
  }

 System.out.println();
  System.out.println("java.util Add: " + format(totAdd/(j-3)) + " K/s RemYes: " + format(totRemYes/(j-3)) + " K/s
Yes: " + format(totYes/(j-3)) + " K/s No: " + format(totNo/(j-3)) + " K/s IterFor: " + format(totIterFor/(j-3))  + "
```

```
K/s");

  System.out.println();

  totAdd = totYes = totNo = totIterFor = totIterBack = totRemYes = 0;

  }

  for(j = 0; j < 20; j++) {

  m = new AVL_TREE_SET();

  /* We first add all pairs to m. */
  for(i = 0; i < n;  i++) m.add(k[i]);

  /* Then we remove the first half and put it back. */
  for(i = 0; i < n/2;  i++) m.remove(k[i]);

  ms = System.currentTimeMillis();
  for(i = 0; i < n/2;  i++) m.add(k[i]);
  d = System.currentTimeMillis() - ms;

  /* Then we remove the other half and put it back again. */
  ms = System.currentTimeMillis();
  for(i = n/2; i < n;  i++) m.remove(k[i]);
  dd = System.currentTimeMillis() - ms ;

  ms = System.currentTimeMillis();
  for(i = n/2; i < n;  i++) m.add(k[i]);
  d += System.currentTimeMillis() - ms;
  if (j > 2) totAdd += n/d;
  System.out.print("Add: " + format(n/d) +" K/s ");

  /* Then we remove again the first half. */
  ms = System.currentTimeMillis();
  for(i = 0; i < n/2;  i++) m.remove(k[i]);
  dd += System.currentTimeMillis() - ms ;
  if (j > 2) totRemYes += n/dd;
  System.out.print("RemYes: " + format(n/dd) +" K/s ");

  /* And then we put it back. */
  for(i = 0; i < n/2;  i++) m.add(k[i]);

  /* We check for pairs in m. */
  ms = System.currentTimeMillis();
  for(i = 0; i < n;  i++) m.contains(k[i]);
  d = 1.0 * n / (System.currentTimeMillis() - ms);
  if (j > 2) totYes += d;
```

```
  System.out.print("Yes: " + format(d) +" K/s ");

  /* We check for pairs not in m. */
  ms = System.currentTimeMillis();
  for(i = 0; i < n;  i++) m.contains(nk[i]);
  d = 1.0 * n / (System.currentTimeMillis() - ms);
  if (j > 2) totNo += d;
  System.out.print("No: " + format(d) +" K/s ");

  /* We iterate on m. */
  KEY_LIST_ITERATOR it = (KEY_LIST_ITERATOR)m.iterator();
  ms = System.currentTimeMillis();
  for(; it.hasNext(); it.NEXT_KEY());
  d = 1.0 * n / (System.currentTimeMillis() - ms);
  if (j > 2) totIterFor += d;
  System.out.print("IterFor: " + format(d) +" K/s ");

  /* We iterate back on m. */
  ms = System.currentTimeMillis();
  for(; it.hasPrevious(); it.PREV_KEY());
  d = 1.0 * n / (System.currentTimeMillis() - ms);
  if (j > 2) totIterBack += d;
  System.out.print("IterBack: " + format(d) +" K/s ");

  System.out.println();
 }


 System.out.println();
 System.out.println("fastutil  Add: " + format(totAdd/(j-3)) + " K/s RemYes: " + format(totRemYes/(j-3)) + " K/s
Yes: " + format(totYes/(j-3)) + " K/s No: " + format(totNo/(j-3)) + " K/s IterFor: " + format(totIterFor/(j-3))  + " K/s
IterBack: " + format(totIterBack/(j-3)) + "K/s");

 System.out.println();
}


private static boolean valEquals(Object o1, Object o2) {
 return o1 == null ? o2 == null : o1.equals(o2);
}

private static void fatal(String msg) {
 System.out.println(msg);
 System.exit(1);
}

private static void ensure(boolean cond, String msg) {
 if (cond) return;
```

```
 fatal(msg);
}

private static Object[] k, v, nk;
private static KEY_TYPE kt[];
private static KEY_TYPE nkt[];
private static AVL_TREE_SET topSet;

protected static void testSets(SORTED_SET m, SortedSet t, int n, int level) {
 long ms;
 boolean mThrowsIllegal, tThrowsIllegal, mThrowsNoElement, tThrowsNoElement;
 boolean rt = false, rm = false;

 if (level > 4) return;


 /* Now we check that both sets agree on first/last keys. */

 mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

 try {
  m.first();
 }
 catch (NoSuchElementException e) { mThrowsNoElement = true; }
 try {
  t.first();
 }
 catch (NoSuchElementException e) { tThrowsNoElement = true; }

 ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): first() divergence at start in
NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
 if (! mThrowsNoElement) ensure(t.first().equals(m.first()), "Error (" + level + ", " + seed + "): m and t differ at start
on their first key (" + m.first() + ", " + t.first() +")");

 mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

 try {
  m.last();
 }
 catch (NoSuchElementException e) { mThrowsNoElement = true; }
 try {
  t.last();
 }
 catch (NoSuchElementException e) { tThrowsNoElement = true; }

 ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): last() divergence at start in
NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
```

```
  if (! mThrowsNoElement) ensure(t.last().equals(m.last()), "Error (" + level + ", " + seed + "): m and t differ at start
on their last key (" + m.last() + ", " + t.last() +")");


  /* Now we check that m and t are equal. */
  if (!m.equals(t) || ! t.equals(m)) System.err.println("m: " + m + " t: " + t);

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) at start");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) at start");



  /* Now we check that m actually holds that data. */
  for(Iterator i=t.iterator(); i.hasNext();) {
   ensure(m.contains(i.next()), "Error (" + level + ", " + seed + "): m and t differ on an entry after insertion (iterating on
t)");
  }

  /* Now we check that m actually holds that data, but iterating on m. */
  for(Iterator i=m.iterator(); i.hasNext();) {
   ensure(t.contains(i.next()), "Error (" + level + ", " + seed + "): m and t differ on an entry after insertion (iterating on
m)");
  }

  /* Now we check that inquiries about random data give the same answer in m and t. For
     m we use the polymorphic method. */

  for(int i=0; i<n;  i++) {
   KEY_TYPE T = genKey();

   mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

   try {
    m.contains(T);
   }
   catch (NoSuchElementException e) { mThrowsNoElement = true; }
   catch (IllegalArgumentException e) { mThrowsIllegal = true; }

   try {
    t.contains(KEY2OBJ(T));
   }
   catch (NoSuchElementException e) { tThrowsNoElement = true; }
   catch (IllegalArgumentException e) { tThrowsIllegal = true; }

   ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): contains() divergence in
NoSuchElementException (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
   ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): contains() divergence in
```

```
IllegalArgumentException (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
  if (!mThrowsNoElement && !mThrowsIllegal) ensure(m.contains(KEY2OBJ(T)) == t.contains(KEY2OBJ(T)),
"Error (" + level + ", " + seed + "): divergence in keys between t and m (polymorphic method)");
  }

  /* Again, we check that inquiries about random data give the same answer in m and t, but
    for m we use the standard method. */

  for(int i=0; i<n;  i++) {
  KEY_TYPE T = genKey();

  mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

  try {
   m.contains(KEY2OBJ(T));
  }
  catch (NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }

  try {
   t.contains(KEY2OBJ(T));
  }
  catch (NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): contains() divergence in
NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): contains() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
  if (!mThrowsNoElement && !mThrowsIllegal) ensure(m.contains(KEY2OBJ(T)) ==  t.contains(KEY2OBJ(T)),
"Error (" + level + ", " + seed + "): divergence between t and m (standard method)");
  }

  /* Now we add and remove random data in m and t, checking that the result is the same. */

  for(int i=0; i<20*n;  i++) {
  KEY_TYPE T = genKey();

  mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

  try {
   rm = m.add(KEY2OBJ(T));
  }
  catch (NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }

  try {
   rt = t.add(KEY2OBJ(T));
```

```
  }
  catch (NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }


  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): add() divergence in
NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): add() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
  if (!mThrowsNoElement && !mThrowsIllegal) ensure(rm == rt, "Error (" + level + ", " + seed + "): divergence in
add() between t and m");

  T = genKey();

  mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

  try {
   rm = m.remove(KEY2OBJ(T));
  }
  catch (NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }

  try {
   rt = t.remove(KEY2OBJ(T));
  }
  catch (NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }


  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): remove() divergence in
NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): remove() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
  if (!mThrowsNoElement && !mThrowsIllegal) ensure(rm == rt, "Error (" + level + ", " + seed + "): divergence in
remove() between t and m");
  }

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after removal");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after removal");

  /* Now we check that m actually holds the same data. */

  for(Iterator i=t.iterator(); i.hasNext();) {
  ensure(m.contains(i.next()), "Error (" + level + ", " + seed + "): m and t differ on an entry after removal (iterating on
t)");
  }

  /* Now we check that m actually holds that data, but iterating on m. */
```

```
  for(Iterator i=m.iterator(); i.hasNext();) {
   ensure(t.contains(i.next()), "Error (" + level + ", " + seed + "): m and t differ on an entry after removal (iterating on m)");
  }

  /* Now we check that both sets agree on first/last keys. */

  mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

  try {
   m.first();
  }
  catch (NoSuchElementException e) { mThrowsNoElement = true; }
  try {
   t.first();
  }
  catch (NoSuchElementException e) { tThrowsNoElement = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): first() divergence in NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
  if (! mThrowsNoElement) ensure(t.first().equals(m.first()), "Error (" + level + ", " + seed + "): m and t differ on their first key (" + m.first() + ", " + t.first() +")");

  mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

  try {
   m.last();
  }
  catch (NoSuchElementException e) { mThrowsNoElement = true; }
  try {
   t.last();
  }
  catch (NoSuchElementException e) { tThrowsNoElement = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): last() divergence in NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");

  if (! mThrowsNoElement) ensure(t.last().equals(m.last()), "Error (" + level + ", " + seed + "): m and t differ on their last key (" + m.last() + ", " + t.last() +")");

  /* Now we check cloning. */

  if (level == 0) {
   ensure(m.equals(((AVL_TREE_SET)m).clone()), "Error (" + level + ", " + seed + "): m does not equal m.clone()");
   ensure(((AVL_TREE_SET)m).clone().equals(m), "Error (" + level + ", " + seed + "): m.clone() does not equal m");
  }
```

```
int h = m.hashCode();


/* Now we save and read m. */

SORTED_SET m2 = null;

try {
 java.io.File ff = new java.io.File("it.unimi.dsi.fastutil.test");
 java.io.OutputStream os = new java.io.FileOutputStream(ff);
 java.io.ObjectOutputStream oos = new java.io.ObjectOutputStream(os);

 oos.writeObject(m);
 oos.close();

 java.io.InputStream is = new java.io.FileInputStream(ff);
 java.io.ObjectInputStream ois = new java.io.ObjectInputStream(is);

 m2 = (SORTED_SET)ois.readObject();
 ois.close();
 ff.delete();
}
catch(Exception e) {
 e.printStackTrace();
 System.exit(1);
}

ensure(m2.hashCode() == h, "Error (" + level + ", " + seed + "): hashCode() changed after save/read");

/* Now we check that m2 actually holds that data. */

ensure(m2.equals(t), "Error (" + level + ", " + seed + "): ! m2.equals(t) after save/read");
ensure(t.equals(m2), "Error (" + level + ", " + seed + "): ! t.equals(m2) after save/read");
/* Now we take out of m everything, and check that it is empty. */

for(Iterator i=t.iterator(); i.hasNext();) m2.remove(i.next());

ensure(m2.isEmpty(), "Error (" + level + ", " + seed + "): m2 is not empty (as it should be)");

/* Now we play with iterators. */

{
 java.util.ListIterator i, j;
 Object J;
 i = (java.util.ListIterator)m.iterator();
 j = new java.util.LinkedList(t).listIterator();

 for(int k = 0; k < 2*n; k++) {
```

```
 ensure(i.hasNext() == j.hasNext(), "Error (" + level + ", " + seed + "): divergence in hasNext()");
 ensure(i.hasPrevious() == j.hasPrevious(), "Error (" + level + ", " + seed + "): divergence in hasPrevious()");

 if (r.nextFloat() < .8 && i.hasNext()) {
  ensure(i.next().equals(J = j.next()), "Error (" + level + ", " + seed + "): divergence in next()");

  if (r.nextFloat() < 0.5) {
   i.remove();
   j.remove();
   t.remove(J);
  }
 }
 else if (r.nextFloat() < .2 && i.hasPrevious()) {
  ensure(i.previous().equals(J = j.previous()), "Error (" + level + ", " + seed + "): divergence in previous()");

  if (r.nextFloat() < 0.5) {
   i.remove();
   j.remove();
   t.remove(J);
  }
 }

 ensure(i.nextIndex() == j.nextIndex(), "Error (" + level + ", " + seed + "): divergence in nextIndex()");
 ensure(i.previousIndex() == j.previousIndex(), "Error (" + level + ", " + seed + "): divergence in previousIndex()");

 }

}

{
 boolean badPrevious = false;
 Object previous = null;
 it.unimi.dsi.fastutil.BidirectionalIterator i;
 java.util.ListIterator j;
 Object I, J;
 KEY_TYPE from = genKey();
 j = new java.util.LinkedList(t).listIterator();
 while(j.hasNext()) {
  Object k = j.next();
  if (((Comparable)k).compareTo(KEY2OBJ(from)) > 0) {
   badPrevious = true;
   j.previous();
   break;
  }
  previous = k;
 }

 i = (it.unimi.dsi.fastutil.BidirectionalIterator)m.iterator(from);
```

```
  for(int k = 0; k < 2*n; k++) {
   ensure(i.hasNext() == j.hasNext(), "Error (" + level + ", " + seed + "): divergence in hasNext() (iterator with starting point " + from + ")");
   ensure(i.hasPrevious() == j.hasPrevious() || badPrevious && (i.hasPrevious() == (previous != null)), "Error (" + level + ", " + seed + "): divergence in hasPrevious() (iterator with starting point " + from + ")");

   if (r.nextFloat() < .8 && i.hasNext()) {
    ensure((I = i.next()).equals(J = j.next()), "Error (" + level + ", " + seed + "): divergence in next() (" + I + ", " + J + ", iterator with starting point " + from + ")");
    //System.err.println("Done next " + I + " " + J + "  " + badPrevious);

    badPrevious = false;

    if (r.nextFloat() < 0.5) {
     //System.err.println("Removing in next");
     i.remove();
     j.remove();
     t.remove(J);
    }
   }
   else if (!badPrevious && r.nextFloat() < .2 && i.hasPrevious()) {
    ensure((I = i.previous()).equals(J = j.previous()), "Error (" + level + ", " + seed + "): divergence in previous() (" + I + ", " + J + ", iterator with starting point " + from + ")");

    if (r.nextFloat() < 0.5) {
     //System.err.println("Removing in prev");
     i.remove();
     j.remove();
     t.remove(J);
    }
   }
  }

 }

 /* Now we check that m actually holds that data. */

 ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after iteration");
 ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after iteration");

 /* Now we select a pair of keys and create a subset. */

 if (! m.isEmpty()) {
  java.util.ListIterator i;
  Object start = m.first(), end = m.first();
  for(i = (java.util.ListIterator)m.iterator(); i.hasNext() && r.nextFloat() < .3; start = end = i.next());
  for(; i.hasNext() && r.nextFloat() < .95; end = i.next());
```

```
  //System.err.println("Checking subSet from " + start + " to " + end + " (level=" + (level+1) + ")...");
  testSets((SORTED_SET)m.subSet((KEY_CLASS)start, (KEY_CLASS)end), t.subSet(start, end), n, level + 1);

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after subSet");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after subSet");

  //System.err.println("Checking headSet to " + end + " (level=" + (level+1) + ")...");
  testSets((SORTED_SET)m.headSet((KEY_CLASS)end), t.headSet(end), n, level + 1);

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after headSet");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after headSet");

  //System.err.println("Checking tailSet from " + start + " (level=" + (level+1) + ")...");
  testSets((SORTED_SET)m.tailSet((KEY_CLASS)start), t.tailSet(start), n, level + 1);

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after tailSet");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after tailSet");
  }


 }


 private static void runTest(int n) {
  AVL_TREE_SET m = new AVL_TREE_SET();
  SortedSet t = new java.util.TreeSet();
  topSet = m;
  k = new Object[n];
  nk = new Object[n];
  kt = new KEY_TYPE[n];
  nkt = new KEY_TYPE[n];

  for(int i = 0; i < n; i++) {
#if KEY_CLASS_Object
   k[i] = kt[i] = genKey();
   nk[i] = nkt[i] = genKey();
#else
   k[i] = new KEY_CLASS(kt[i] = genKey());
   nk[i] = new KEY_CLASS(nkt[i] = genKey());
#endif
  }

  /* We add pairs to t. */
  for(int i = 0; i < n;  i++) t.add(k[i]);

  /* We add to m the same data */
  m.addAll(t);
```

```
 testSets(m, t, n, 0);

 System.out.println("Test OK");
 return;
}


public static void main(String args[]) {
 int n  = Integer.parseInt(args[1]);
 if (args.length > 2) r = new java.util.Random(seed = Long.parseLong(args[2]));


 try {
  if ("speedTest".equals(args[0]) || "speedComp".equals(args[0])) speedTest(n, "speedComp".equals(args[0]));
  else if ("test".equals(args[0])) runTest(n);
 } catch(Throwable e) {
  e.printStackTrace(System.err);
  System.err.println("seed: " + seed);
 }
}

#endif


}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/AVLTreeSet.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
```

```
package PACKAGE;

import it.unimi.dsi.fastutil.Function;

/** A class providing static methods and objects that do useful things with type-specific functions.
*
* @see it.unimi.dsi.fastutil.Function
* @see java.util.Collections
*/

public final class FUNCTIONS {

 private FUNCTIONS() {}

 /** An immutable class representing an empty type-specific function.
  *
  * <p>This class may be useful to implement your own in case you subclass
  * a type-specific function.
  */

 public static class EmptyFunction KEY_VALUE_GENERIC extends ABSTRACT_FUNCTION
KEY_VALUE_GENERIC implements java.io.Serializable, Cloneable {

  private static final long serialVersionUID = -7046029254386353129L;

  protected EmptyFunction() {}

  @Override
  public VALUE_GENERIC_TYPE GET_VALUE(final KEY_TYPE k) { return VALUE_NULL; }

  @Override
  public boolean containsKey(final KEY_TYPE k) { return false; }

  @Override
  public VALUE_GENERIC_TYPE defaultReturnValue()  { return VALUE_NULL; }

  @Override
  public void defaultReturnValue(final VALUE_GENERIC_TYPE defRetValue)  { throw new
UnsupportedOperationException(); }

  @Override
  public int size() { return 0; }

  @Override
  public void clear() {}

  @Override
```

```
 public Object clone() { return EMPTY_FUNCTION; }

 @Override
 public int hashCode() { return 0; }

 @Override
 public boolean equals(final Object o) {
  if (! (o instanceof Function)) return false;
  return ((Function<?,?>)o).size() == 0;
 }

 @Override
 public String toString() { return "{}"; }

 private Object readResolve() { return EMPTY_FUNCTION; }
}


/** An empty type-specific function (immutable). It is serializable and cloneable. */

 SUPPRESS_WARNINGS_KEY_VALUE_RAWTYPES
 public static final EmptyFunction EMPTY_FUNCTION = new EmptyFunction();


/** An immutable class representing a type-specific singleton function. Note
 *  that the default return value is still settable.
 *
 * <p>Note that albeit the function is immutable, its default return value may be changed.
 *
 * <p>This class may be useful to implement your own in case you subclass
 * a type-specific function.
 */

 public static class Singleton KEY_VALUE_GENERIC extends ABSTRACT_FUNCTION
KEY_VALUE_GENERIC implements java.io.Serializable, Cloneable {

 private static final long serialVersionUID = -7046029254386353129L;

 protected final KEY_GENERIC_TYPE key;
 protected final VALUE_GENERIC_TYPE value;

 protected Singleton(final KEY_GENERIC_TYPE key, final VALUE_GENERIC_TYPE value) {
  this.key = key;
  this.value = value;
 }

 @Override
 public boolean containsKey(final KEY_TYPE k) { return KEY_EQUALS(key, k); }
```

```
 @Override
 public VALUE_GENERIC_TYPE GET_VALUE(final KEY_TYPE k) { return KEY_EQUALS(key, k) ? value :
defRetValue; }

 @Override
 public int size() { return 1; }

 @Override
 public Object clone() { return this; }
 }

 /** Returns a type-specific immutable function containing only the specified pair.
  * The returned function is serializable and cloneable.
  *
  * <p>Note that albeit the returned function is immutable, its default return value may be changed.
  *
  * @param key the only key of the returned function.
  * @param value the only value of the returned function.
  * @return a type-specific immutable function containing just the pair {@code &lt;key,value&gt;}.
  */

 public static KEY_VALUE_GENERIC FUNCTION KEY_VALUE_GENERIC singleton(final
KEY_GENERIC_TYPE key, VALUE_GENERIC_TYPE value) {
  return new Singleton KEY_VALUE_GENERIC_DIAMOND(key, value);
 }

#if KEYS_PRIMITIVE || VALUES_PRIMITIVE

 /** Returns a type-specific immutable function containing only the specified pair. The returned function is
serializable and cloneable.
  *
  * <p>Note that albeit the returned function is immutable, its default return value may be changed.
  *
  * @param key the only key of the returned function.
  * @param value the only value of the returned function.
  * @return a type-specific immutable function containing just the pair {@code &lt;key,value&gt;}.
  */

 public static KEY_VALUE_GENERIC FUNCTION KEY_VALUE_GENERIC singleton(final
KEY_GENERIC_CLASS key, final VALUE_GENERIC_CLASS value) {
  return new Singleton KEY_VALUE_GENERIC_DIAMOND(KEY_CLASS2TYPE(key),
VALUE_CLASS2TYPE(value));
 }

#endif
```

```
 /** A synchronized wrapper class for functions. */

 public static class SynchronizedFunction KEY_VALUE_GENERIC implements FUNCTION
KEY_VALUE_GENERIC, java.io.Serializable {

  private static final long serialVersionUID = -7046029254386353129L;

  protected final FUNCTION KEY_VALUE_GENERIC function;
  protected final Object sync;

  protected SynchronizedFunction(final FUNCTION KEY_VALUE_GENERIC f, final Object sync) {
   if (f == null) throw new NullPointerException();
   this.function = f;
   this.sync = sync;
  }

  protected SynchronizedFunction(final FUNCTION KEY_VALUE_GENERIC f) {
   if (f == null) throw new NullPointerException();
   this.function = f;
   this.sync = this;
  }

#ifdef JDK_PRIMITIVE_FUNCTION

#if KEY_WIDENED
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
#endif
  @Override
  public VALUE_GENERIC_TYPE_WIDENED
JDK_PRIMITIVE_FUNCTION_APPLY(KEY_GENERIC_TYPE_WIDENED operand) { synchronized(sync) {
return function.JDK_PRIMITIVE_FUNCTION_APPLY(operand); } }

#endif

#if KEYS_PRIMITIVE || VALUES_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
#endif
  @Override
  public VALUE_GENERIC_CLASS apply(final KEY_GENERIC_CLASS key) { synchronized (sync) { return
function.apply(key); } }

  @Override
  public int size() { synchronized(sync) { return function.size(); } }
```

```
  @Override
  public VALUE_GENERIC_TYPE defaultReturnValue()  { synchronized(sync) { return
function.defaultReturnValue(); } }

  @Override
  public void defaultReturnValue(final VALUE_GENERIC_TYPE defRetValue)  { synchronized(sync) {
function.defaultReturnValue(defRetValue); } }

  @Override
  public boolean containsKey(final KEY_TYPE k) { synchronized(sync) { return function.containsKey(k); } }

#if KEYS_PRIMITIVE

  @Deprecated
  @Override
  public boolean containsKey(final Object k) { synchronized(sync) { return function.containsKey(k); } }

#endif

  @Override
  public VALUE_GENERIC_TYPE put(final KEY_GENERIC_TYPE k, final VALUE_GENERIC_TYPE v) {
synchronized(sync) { return function.put(k, v); } }

  @Override
  public VALUE_GENERIC_TYPE GET_VALUE(final KEY_TYPE k) { synchronized(sync) { return
function.GET_VALUE(k); } }

  @Override
  public VALUE_GENERIC_TYPE REMOVE_VALUE(final KEY_TYPE k) { synchronized(sync) { return
function.REMOVE_VALUE(k); } }

  @Override
  public void clear() { synchronized(sync) { function.clear(); } }

#if KEYS_PRIMITIVE || VALUES_PRIMITIVE

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public VALUE_GENERIC_CLASS put(final KEY_GENERIC_CLASS k, final VALUE_GENERIC_CLASS v) {
synchronized(sync) { return function.put(k, v); } }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public VALUE_GENERIC_CLASS get(final Object k) { synchronized(sync) { return function.get(k); } }
```

```
 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 public VALUE_GENERIC_CLASS remove(final Object k) { synchronized(sync) { return function.remove(k); } }

#endif

 @Override
 public int hashCode() { synchronized(sync) { return function.hashCode(); } }

 @Override
 public boolean equals(final Object o) {
  if (o == this) return true;
  synchronized(sync) { return function.equals(o); }
 }

 @Override
 public String toString() { synchronized(sync) { return function.toString(); } }

 private void writeObject(java.io.ObjectOutputStream s) throws java.io.IOException {
  synchronized(sync) { s.defaultWriteObject(); }
 }
}

 /** Returns a synchronized type-specific function backed by the given type-specific function.
  *
  * @param f the function to be wrapped in a synchronized function.
  * @return a synchronized view of the specified function.
  * @see java.util.Collections#synchronizedMap(java.util.Map)
  */
 public static KEY_VALUE_GENERIC FUNCTION KEY_VALUE_GENERIC synchronize(final FUNCTION
KEY_VALUE_GENERIC f) { return new SynchronizedFunction KEY_VALUE_GENERIC_DIAMOND(f); }

 /** Returns a synchronized type-specific function backed by the given type-specific function, using an assigned
object to synchronize.
  *
  * @param f the function to be wrapped in a synchronized function.
  * @param sync an object that will be used to synchronize the access to the function.
  * @return a synchronized view of the specified function.
  * @see java.util.Collections#synchronizedMap(java.util.Map)
  */

 public static KEY_VALUE_GENERIC FUNCTION KEY_VALUE_GENERIC synchronize(final FUNCTION
KEY_VALUE_GENERIC f, final Object sync) { return new SynchronizedFunction
KEY_VALUE_GENERIC_DIAMOND(f, sync); }
```

```
 /** An unmodifiable wrapper class for functions. */

 public static class UnmodifiableFunction KEY_VALUE_GENERIC extends ABSTRACT_FUNCTION
KEY_VALUE_GENERIC implements java.io.Serializable {

  private static final long serialVersionUID = -7046029254386353129L;

  protected final FUNCTION KEY_VALUE_GENERIC function;

  protected UnmodifiableFunction(final FUNCTION KEY_VALUE_GENERIC f) {
   if (f == null) throw new NullPointerException();
   this.function = f;
  }

  @Override
  public int size() { return function.size(); }

  @Override
  public VALUE_GENERIC_TYPE defaultReturnValue()  { return function.defaultReturnValue(); }

  @Override
  public void defaultReturnValue(final VALUE_GENERIC_TYPE defRetValue)  { throw new
UnsupportedOperationException(); }

  @Override
  public boolean containsKey(final KEY_TYPE k) { return function.containsKey(k); }

  @Override
  public VALUE_GENERIC_TYPE put(final KEY_GENERIC_TYPE k, final VALUE_GENERIC_TYPE v) { throw
new UnsupportedOperationException(); }

  @Override
  public VALUE_GENERIC_TYPE GET_VALUE(final KEY_TYPE k) { return function.GET_VALUE(k); }

  @Override
  public VALUE_GENERIC_TYPE REMOVE_VALUE(final KEY_TYPE k) { throw new
UnsupportedOperationException(); }

  @Override
  public void clear() { throw new UnsupportedOperationException(); }

#if KEYS_PRIMITIVE || VALUES_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
```

```
  public VALUE_GENERIC_CLASS put(final KEY_GENERIC_CLASS k, final VALUE_GENERIC_CLASS v) {
throw new UnsupportedOperationException(); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public VALUE_GENERIC_CLASS get(final Object k) { return function.get(k); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public VALUE_GENERIC_CLASS remove(final Object k) { throw new UnsupportedOperationException(); }
#endif

  @Override
  public int hashCode() { return function.hashCode(); }

  @Override
  public boolean equals(Object o) { return o == this || function.equals(o); }

  @Override
  public String toString() { return function.toString(); }
 }

 /** Returns an unmodifiable type-specific function backed by the given type-specific function.
  *
  * @param f the function to be wrapped in an unmodifiable function.
  * @return an unmodifiable view of the specified function.
  * @see java.util.Collections#unmodifiableMap(java.util.Map)
  */
 public static KEY_VALUE_GENERIC FUNCTION KEY_VALUE_GENERIC unmodifiable(final FUNCTION
KEY_VALUE_GENERIC f) { return new UnmodifiableFunction KEY_VALUE_GENERIC_DIAMOND(f); }


#if KEYS_PRIMITIVE || VALUES_PRIMITIVE
 /** An adapter for mapping generic total functions to partial primitive functions. */

 public static class PrimitiveFunction KEY_VALUE_GENERIC implements FUNCTION
KEY_VALUE_GENERIC {
  protected final java.util.function.Function<? super KEY_GENERIC_CLASS, ? extends
VALUE_GENERIC_CLASS> function;

  protected PrimitiveFunction(java.util.function.Function<? super KEY_GENERIC_CLASS, ? extends
VALUE_GENERIC_CLASS> function) {
   this.function = function;
  }
```

```
#if KEYS_PRIMITIVE
  @Override
  public boolean containsKey(KEY_GENERIC_TYPE key) { return function.apply(KEY2OBJ(key)) != null; }
#endif

  SUPPRESS_WARNINGS_KEY_UNCHECKED
#if KEYS_PRIMITIVE
  @Deprecated
#endif
  @Override
  public boolean containsKey(Object key) {
#if KEYS_PRIMITIVE
   if (key == null) return false;
#endif
   return function.apply(KEY_CLASS_CAST (key)) != null;
  }

  SUPPRESS_WARNINGS_KEY_UNCHECKED
  @Override
  public VALUE_GENERIC_TYPE GET_VALUE(KEY_TYPE key) {
   VALUE_GENERIC_CLASS v = function.apply(KEY_GENERIC_CAST KEY2OBJ(key));
#if VALUES_PRIMITIVE
   if (v == null) return defaultReturnValue();
#else
   if (v == null) return null;
#endif
   return VALUE_CLASS2TYPE(v);
  }

  SUPPRESS_WARNINGS_KEY_UNCHECKED
  @Deprecated
  @Override
  public VALUE_GENERIC_CLASS get(Object key) {
#if KEYS_PRIMITIVE
   if (key == null) return null;
#endif
   return function.apply(KEY_CLASS_CAST key);
  }

  @Deprecated
  @Override
  public VALUE_GENERIC_CLASS put(final KEY_GENERIC_CLASS key, final VALUE_GENERIC_CLASS
value) { throw new UnsupportedOperationException(); }
 }

 /** Returns a (partial) type-specific function based on the given total generic function.
  * <p>The returned function contains all keys which are not mapped to {@code null}. If the function already
```

```
 * is a primitive function, it is returned without changes.
 * <p><strong>Warning</strong>: If the given function is a &ldquo;widened&rdquo; primitive function (e.g. an
 * {@code Int2IntFunction} given to {@code Short2ShortFunctions}), it still is wrapped into a proxy,
 * decreasing performance.
 *
 * @param f the function to be converted to a type-specific function.
 * @return a primitive view of the specified function.
 * @throws NullPointerException if {@code f} is null.
 * @see PrimitiveFunction
 * @since 8.1.0
 */
 SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
 public static KEY_VALUE_GENERIC FUNCTION KEY_VALUE_GENERIC primitive(final
java.util.function.Function<? super KEY_GENERIC_CLASS, ? extends VALUE_GENERIC_CLASS> f) {
  java.util.Objects.requireNonNull(f);

  if (f instanceof FUNCTION) return (FUNCTION KEY_VALUE_GENERIC) f;
#if defined JDK_PRIMITIVE_FUNCTION
  if (f instanceof JDK_PRIMITIVE_FUNCTION)
#if KEYS_PRIMITIVE
#if VALUE_WIDENED
   return key -> VALUE_NARROWING(((JDK_PRIMITIVE_FUNCTION KEY_VALUE_GENERIC)
f).JDK_PRIMITIVE_FUNCTION_APPLY(key));
#else
   return ((JDK_PRIMITIVE_FUNCTION KEY_VALUE_GENERIC) f)::JDK_PRIMITIVE_FUNCTION_APPLY;
#endif
#else
   return key -> VALUE_NARROWING(((JDK_PRIMITIVE_FUNCTION KEY_VALUE_GENERIC)
f).JDK_PRIMITIVE_FUNCTION_APPLY((K)(key)));
#endif
#endif
  return new PrimitiveFunction KEY_VALUE_GENERIC_DIAMOND(f);
 }
#endif
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/Functions.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
```

```
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

/**  An abstract class facilitating the creation of type-specific {@linkplain java.util.ListIterator list iterators}.
*
* @deprecated As of fastutil 8 this class is no longer necessary, as its previous abstract
* methods are now default methods of the type-specific interface.
*/

@Deprecated
public abstract class KEY_ABSTRACT_LIST_ITERATOR KEY_GENERIC extends
KEY_ABSTRACT_BIDI_ITERATOR KEY_GENERIC implements KEY_LIST_ITERATOR KEY_GENERIC {
 protected KEY_ABSTRACT_LIST_ITERATOR() {}
}

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-
a775574/drv/AbstractListIterator.drv
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2007-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;
```

```
import java.util.Collection;
import java.util.NoSuchElementException;

/** A simple, brute-force implementation of a set based on a backing array.
*
* <p>The main purpose of this
* implementation is that of wrapping cleanly the brute-force approach to the storage of a very
* small number of items: just put them into an array and scan linearly to find an item.
*/

public class ARRAY_SET KEY_GENERIC extends ABSTRACT_SET KEY_GENERIC implements
java.io.Serializable, Cloneable {

 private static final long serialVersionUID = 1L;
 /** The backing array (valid up to {@link #size}, excluded). */
 private transient KEY_TYPE[] a;
 /** The number of valid entries in {@link #a}. */
 private int size;

 /** Creates a new array set using the given backing array. The resulting set will have as many elements as the array.
  *
  * <p>It is responsibility of the caller that the elements of {@code a} are distinct.
  *
  * @param a the backing array.
  */
 public ARRAY_SET(final KEY_TYPE[] a) {
  this.a = a;
  size = a.length;
 }

 /** Creates a new empty array set.
  */
 public ARRAY_SET() { this.a = ARRAYS.EMPTY_ARRAY; }

 /** Creates a new empty array set of given initial capacity.
  *
  * @param capacity the initial capacity.
  */
 public ARRAY_SET(final int capacity) { this.a = new KEY_TYPE[capacity]; }

 /** Creates a new array set copying the contents of a given collection.
  * @param c a collection.
  */
 public ARRAY_SET(COLLECTION KEY_GENERIC c) {
  this(c.size ());
  addAll(c);
 }
```

```
 /** Creates a new array set copying the contents of a given set.
  * @param c a collection.
  */
 public ARRAY_SET(final Collection<? extends KEY_GENERIC_CLASS> c) {
  this(c.size());
  addAll(c);
 }


 /** Creates a new array set using the given backing array and the given number of elements of the array.
  *
  * <p>It is responsibility of the caller that the first {@code size} elements of {@code a} are distinct.
  *
  * @param a the backing array.
  * @param size the number of valid elements in {@code a}.
  */
 public ARRAY_SET(final KEY_TYPE[] a, final int size) {
  this.a = a;
  this.size = size;
  if (size > a.length) throw new IllegalArgumentException("The provided size (" + size + ") is larger than or equal to
the array size (" + a.length + ")");
 }

 private int findKey(final KEY_TYPE o) {
  for(int i = size; i-- != 0;) if (KEY_EQUALS(a[i], o)) return i;
  return -1;
 }

 @Override
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public KEY_ITERATOR KEY_GENERIC iterator() {
  return new KEY_ITERATOR KEY_GENERIC () {
   int next = 0;

   @Override
   public boolean hasNext() { return next < size; }

   @Override
   public KEY_GENERIC_TYPE NEXT_KEY() {
    if (! hasNext()) throw new NoSuchElementException();
    return KEY_GENERIC_CAST a[next++];
   }

   @Override
   public void remove() {
    final int tail = size-- - next--;
    System.arraycopy(a, next + 1, a, next, tail);
 #if KEYS_REFERENCE
```

```
    a[size] = null;
#endif
   }
  };
 }

 @Override
 public boolean contains(final KEY_TYPE k) { return findKey(k) != -1; }

 @Override
 public int size() { return size; }

 @Override
 public boolean remove(final KEY_TYPE k) {
  final int pos = findKey(k);
  if (pos == -1) return false;
  final int tail = size - pos - 1;
  for(int i = 0; i < tail; i++) a[pos + i] = a[pos + i + 1];
  size--;
#if KEYS_REFERENCE
  a[size] = null;
#endif
  return true;
 }

 @Override
 public boolean add(final KEY_GENERIC_TYPE k) {
  final int pos = findKey(k);
  if (pos != -1) return false;
  if (size == a.length) {
   final KEY_TYPE[] b = new KEY_TYPE[size == 0 ? 2 : size * 2];
   for(int i = size; i-- != 0;) b[i] = a[i];
   a = b;
  }
  a[size++] = k;
  return true;
 }

 @Override
 public void clear() {
#if KEYS_REFERENCE
  java.util.Arrays.fill(a, 0, size, null);
#endif
  size = 0;
 }

 @Override
 public boolean isEmpty() { return size == 0; }
```

```
 /** Returns a deep copy of this set.
  *
  * <p>This method performs a deep copy of this array set; the data stored in the
  * set, however, is not cloned. Note that this makes a difference only for object keys.
  *
  *  @return a deep copy of this set.
  */
 @Override
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public ARRAY_SET KEY_GENERIC clone() {
  ARRAY_SET KEY_GENERIC c;
  try {
   c = (ARRAY_SET KEY_GENERIC)super.clone();
  }
  catch(CloneNotSupportedException cantHappen) {
   throw new InternalError();
  }
  c.a = a.clone();
  return c;
 }

 private void writeObject(java.io.ObjectOutputStream s) throws java.io.IOException {
  s.defaultWriteObject();
  for(int i = 0; i < size; i++) s.WRITE_KEY(a[i]);
 }

 private void readObject(java.io.ObjectInputStream s) throws java.io.IOException, ClassNotFoundException {
  s.defaultReadObject();
  a = new KEY_TYPE[size];
  for(int i = 0; i < size; i++) a[i] = s.READ_KEY();
 }
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/ArraySet.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
```

```
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import it.unimi.dsi.fastutil.Hash;
import it.unimi.dsi.fastutil.HashCommon;
import static it.unimi.dsi.fastutil.HashCommon.arraySize;
import static it.unimi.dsi.fastutil.HashCommon.maxFill;

import java.util.Map;
import java.util.Arrays;
import java.util.NoSuchElementException;
import java.util.function.Consumer;

import VALUE_PACKAGE.VALUE_COLLECTION;
import VALUE_PACKAGE.VALUE_ABSTRACT_COLLECTION;

#if VALUES_PRIMITIVE
import VALUE_PACKAGE.VALUE_ITERATOR;
#endif

#if VALUE_CLASS_Boolean
import it.unimi.dsi.fastutil.booleans.BooleanConsumer;
#endif


#ifdef Linked

import java.util.Comparator;

#if VALUES_PRIMITIVE
import VALUE_PACKAGE.VALUE_LIST_ITERATOR;
#else
import it.unimi.dsi.fastutil.objects.ObjectIterator;
#endif

import it.unimi.dsi.fastutil.objects.AbstractObjectSortedSet;
import it.unimi.dsi.fastutil.objects.ObjectListIterator;
import it.unimi.dsi.fastutil.objects.ObjectBidirectionalIterator;
import it.unimi.dsi.fastutil.objects.ObjectSortedSet;

/**  A type-specific linked hash map with with a fast, small-footprint implementation.
```

```
*
* <p>Instances of this class use a hash table to represent a map. The table is
* filled up to a specified <em>load factor</em>, and then doubled in size to
* accommodate new entries. If the table is emptied below <em>one fourth</em>
* of the load factor, it is halved in size; however, the table is never reduced to a
* size smaller than that at creation time: this approach makes it
* possible to create maps with a large capacity in which insertions and
* deletions do not cause immediately rehashing. Moreover, halving is
* not performed when deleting entries from an iterator, as it would interfere
* with the iteration process.
*
* <p>Note that {@link #clear()} does not modify the hash table size.
* Rather, a family of {@linkplain #trim() trimming
* methods} lets you control the size of the table; this is particularly useful
* if you reuse instances of this class.
*
* <p>Iterators generated by this map will enumerate pairs in the same order in which they
* have been added to the map (addition of pairs whose key is already present
* in the map does not change the iteration order). Note that this order has nothing in common with the natural
* order of the keys. The order is kept by means of a doubly linked list, represented
* <i>via</i> an array of longs parallel to the table.
*
* <p>This class implements the interface of a sorted map, so to allow easy
* access of the iteration order: for instance, you can get the first key
* in iteration order with {@code firstKey()} without having to create an
* iterator; however, this class partially violates the {@link java.util.SortedMap}
* contract because all submap methods throw an exception and {@link
* #comparator()} returns always {@code null}.
*
* <p>Additional methods, such as {@code getAndMoveToFirst()}, make it easy
* to use instances of this class as a cache (e.g., with LRU policy).
*
* <p>The iterators provided by the views of this class using are type-specific
* {@linkplain java.util.ListIterator list iterators}, and can be started at any
* element <em>which is a key of the map</em>, or
* a {@link NoSuchElementException} exception will be thrown.
* If, however, the provided element is not the first or last key in the
* map, the first access to the list index will require linear time, as in the worst case
* the entire key set must be scanned in iteration order to retrieve the positional
* index of the starting key. If you use just the methods of a type-specific {@link
it.unimi.dsi.fastutil.BidirectionalIterator},
* however, all operations will be performed in constant time.
*
* @see Hash
* @see HashCommon
*/

public class OPEN_HASH_MAP KEY_VALUE_GENERIC extends ABSTRACT_SORTED_MAP
```

KEY_VALUE_GENERIC implements java.io.Serializable, Cloneable, Hash {

#else

import it.unimi.dsi.fastutil.objects.AbstractObjectSet;
import it.unimi.dsi.fastutil.objects.ObjectIterator;

#ifdef Custom

/** A type-specific hash map with a fast, small-footprint implementation whose {@linkplain
it.unimi.dsi.fastutil.Hash.Strategy hashing strategy}
* is specified at creation time.
*
* <p>Instances of this class use a hash table to represent a map. The table is
* filled up to a specified load factor, and then doubled in size to
* accommodate new entries. If the table is emptied below one fourth
* of the load factor, it is halved in size; however, the table is never reduced to a
* size smaller than that at creation time: this approach makes it
* possible to create maps with a large capacity in which insertions and
* deletions do not cause immediately rehashing. Moreover, halving is
* not performed when deleting entries from an iterator, as it would interfere
* with the iteration process.
*
* <p>Note that {@link #clear()} does not modify the hash table size.
* Rather, a family of {@linkplain #trim() trimming
* methods} lets you control the size of the table; this is particularly useful
* if you reuse instances of this class.
*
* @see Hash
* @see HashCommon
*/

public class OPEN_HASH_MAP KEY_VALUE_GENERIC extends ABSTRACT_MAP
KEY_VALUE_GENERIC implements java.io.Serializable, Cloneable, Hash {

#else

/** A type-specific hash map with a fast, small-footprint implementation.
*
* <p>Instances of this class use a hash table to represent a map. The table is
* filled up to a specified load factor, and then doubled in size to
* accommodate new entries. If the table is emptied below one fourth
* of the load factor, it is halved in size; however, the table is never reduced to a
* size smaller than that at creation time: this approach makes it
* possible to create maps with a large capacity in which insertions and
* deletions do not cause immediately rehashing. Moreover, halving is
* not performed when deleting entries from an iterator, as it would interfere
* with the iteration process.

```
*
* <p>Note that {@link #clear()} does not modify the hash table size.
* Rather, a family of {@linkplain #trim() trimming
* methods} lets you control the size of the table; this is particularly useful
* if you reuse instances of this class.
*
* @see Hash
* @see HashCommon
*/

public class OPEN_HASH_MAP KEY_VALUE_GENERIC extends ABSTRACT_MAP
KEY_VALUE_GENERIC implements java.io.Serializable, Cloneable, Hash {

#endif

#endif

 private static final long serialVersionUID = 0L;
 private static final boolean ASSERTS = ASSERTS_VALUE;

 /** The array of keys. */
 protected transient KEY_GENERIC_TYPE[] key;

 /** The array of values. */
 protected transient VALUE_GENERIC_TYPE[] value;

 /** The mask for wrapping a position counter. */
 protected transient int mask;

 /** Whether this map contains the key zero. */
 protected transient boolean containsNullKey;

#ifdef Custom
 /** The hash strategy of this custom map. */
 protected STRATEGY KEY_SUPER_GENERIC strategy;
#endif

#ifdef Linked
 /** The index of the first entry in iteration order. It is valid iff {@link #size} is nonzero; otherwise, it contains -1. */
 protected transient int first = -1;
 /** The index of the last entry in iteration order. It is valid iff {@link #size} is nonzero; otherwise, it contains -1. */
 protected transient int last = -1;
 /** For each entry, the next and the previous entry in iteration order,
  * stored as {@code ((prev & 0xFFFFFFFFL) << 32) | (next & 0xFFFFFFFFL)}.
  * The first entry contains predecessor -1, and the last entry
  * contains successor -1. */
  protected transient long[] link;
#endif
```

```
 /** The current table size. */
 protected transient int n;

 /** Threshold after which we rehash. It must be the table size times {@link #f}. */
 protected transient int maxFill;

 /** We never resize below this threshold, which is the construction-time {#n}. */
 protected final transient int minN;

 /** Number of entries in the set (including the key zero, if present). */
 protected int size;

 /** The acceptable load factor. */
 protected final float f;

#ifdef Linked
 /** Cached set of entries. */
 protected transient FastSortedEntrySet KEY_VALUE_GENERIC entries;

 /** Cached set of keys. */
 protected transient SORTED_SET KEY_GENERIC keys;
#else
 /** Cached set of entries. */
 protected transient FastEntrySet KEY_VALUE_GENERIC entries;

 /** Cached set of keys. */
 protected transient SET KEY_GENERIC keys;
#endif

 /** Cached collection of values. */
 protected transient VALUE_COLLECTION VALUE_GENERIC values;


#ifdef Custom
 /** Creates a new hash map.
  *
  * <p>The actual table size will be the least power of two greater than {@code expected}/{@code f}.
  *
  * @param expected the expected number of elements in the hash map.
  * @param f the load factor.
  * @param strategy the strategy.
  */
 SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
 public OPEN_HASH_MAP(final int expected, final float f, final STRATEGY KEY_SUPER_GENERIC strategy) {
  this.strategy = strategy;
#else
 /** Creates a new hash map.
```

```
 *
 * <p>The actual table size will be the least power of two greater than {@code expected}/{@code f}.
 *
 * @param expected the expected number of elements in the hash map.
 * @param f the load factor.
 */
 SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
 public OPEN_HASH_MAP(final int expected, final float f) {
#endif
  if (f <= 0 || f > 1) throw new IllegalArgumentException("Load factor must be greater than 0 and smaller than or
equal to 1");
  if (expected < 0) throw new IllegalArgumentException("The expected number of elements must be nonnegative");

  this.f = f;

  minN = n = arraySize(expected, f);
  mask = n - 1;
  maxFill = maxFill(n, f);
  key = KEY_GENERIC_ARRAY_CAST new KEY_TYPE[n + 1];
  value = VALUE_GENERIC_ARRAY_CAST new VALUE_TYPE[n + 1];
#ifdef Linked
  link = new long[n + 1];
#endif
 }


#ifdef Custom
 /** Creates a new hash map with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor.
  *
  * @param expected the expected number of elements in the hash map.
  * @param strategy the strategy.
  */

 public OPEN_HASH_MAP(final int expected, final STRATEGY KEY_SUPER_GENERIC strategy) {
  this(expected, DEFAULT_LOAD_FACTOR, strategy);
 }
#else
 /** Creates a new hash map with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor.
  *
  * @param expected the expected number of elements in the hash map.
  */

 public OPEN_HASH_MAP(final int expected) {
  this(expected, DEFAULT_LOAD_FACTOR);
 }
#endif
```

```
#ifdef Custom
 /** Creates a new hash map with initial expected {@link Hash#DEFAULT_INITIAL_SIZE} entries
  * and {@link Hash#DEFAULT_LOAD_FACTOR} as load factor.
  * @param strategy the strategy.
  */

 public OPEN_HASH_MAP(final STRATEGY KEY_SUPER_GENERIC strategy) {
  this(DEFAULT_INITIAL_SIZE, DEFAULT_LOAD_FACTOR, strategy);
 }
#else
 /** Creates a new hash map with initial expected {@link Hash#DEFAULT_INITIAL_SIZE} entries
  * and {@link Hash#DEFAULT_LOAD_FACTOR} as load factor.
  */

 public OPEN_HASH_MAP() {
  this(DEFAULT_INITIAL_SIZE, DEFAULT_LOAD_FACTOR);
 }
#endif


#ifdef Custom
 /** Creates a new hash map copying a given one.
  *
  * @param m a {@link Map} to be copied into the new hash map.
  * @param f the load factor.
  * @param strategy the strategy.
  */

 public OPEN_HASH_MAP(final Map<? extends KEY_GENERIC_CLASS, ? extends
VALUE_GENERIC_CLASS> m, final float f, final STRATEGY KEY_SUPER_GENERIC strategy) {
  this(m.size(), f, strategy);
  putAll(m);
 }
#else
 /** Creates a new hash map copying a given one.
  *
  * @param m a {@link Map} to be copied into the new hash map.
  * @param f the load factor.
  */

 public OPEN_HASH_MAP(final Map<? extends KEY_GENERIC_CLASS, ? extends
VALUE_GENERIC_CLASS> m, final float f) {
  this(m.size(), f);
  putAll(m);
 }
#endif

#ifdef Custom
```

```
 /** Creates a new hash map with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor copying a given one.
  *
  * @param m a {@link Map} to be copied into the new hash map.
  * @param strategy the strategy.
  */

 public OPEN_HASH_MAP(final Map<? extends KEY_GENERIC_CLASS, ? extends
VALUE_GENERIC_CLASS> m, final STRATEGY KEY_SUPER_GENERIC strategy) {
  this(m, DEFAULT_LOAD_FACTOR, strategy);
 }
#else
 /** Creates a new hash map with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor copying a given one.
  *
  * @param m a {@link Map} to be copied into the new hash map.
  */

 public OPEN_HASH_MAP(final Map<? extends KEY_GENERIC_CLASS, ? extends
VALUE_GENERIC_CLASS> m) {
  this(m, DEFAULT_LOAD_FACTOR);
 }
#endif

#ifdef Custom
 /** Creates a new hash map copying a given type-specific one.
  *
  * @param m a type-specific map to be copied into the new hash map.
  * @param f the load factor.
  * @param strategy the strategy.
  */

 public OPEN_HASH_MAP(final MAP KEY_VALUE_GENERIC m, final float f, final STRATEGY
KEY_SUPER_GENERIC strategy) {
  this(m.size(), f, strategy);
  putAll(m);
 }
#else
 /** Creates a new hash map copying a given type-specific one.
  *
  * @param m a type-specific map to be copied into the new hash map.
  * @param f the load factor.
  */

 public OPEN_HASH_MAP(final MAP KEY_VALUE_GENERIC m, final float f) {
  this(m.size(), f);
  putAll(m);
 }
#endif
```

```
#ifdef Custom
 /** Creates a new hash map with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor copying a given type-specific one.
  *
  * @param m a type-specific map to be copied into the new hash map.
  * @param strategy the strategy.
  */

 public OPEN_HASH_MAP(final MAP KEY_VALUE_GENERIC m, final STRATEGY KEY_SUPER_GENERIC strategy) {
  this(m, DEFAULT_LOAD_FACTOR, strategy);
 }
#else
 /** Creates a new hash map with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor copying a given type-specific one.
  *
  * @param m a type-specific map to be copied into the new hash map.
  */

 public OPEN_HASH_MAP(final MAP KEY_VALUE_GENERIC m) {
  this(m, DEFAULT_LOAD_FACTOR);
 }
#endif


#ifdef Custom
 /** Creates a new hash map using the elements of two parallel arrays.
  *
  * @param k the array of keys of the new hash map.
  * @param v the array of corresponding values in the new hash map.
  * @param f the load factor.
  * @param strategy the strategy.
  * @throws IllegalArgumentException if {@code k} and {@code v} have different lengths.
  */

 public OPEN_HASH_MAP(final KEY_GENERIC_TYPE[] k, final VALUE_GENERIC_TYPE[] v, final float f, final STRATEGY KEY_SUPER_GENERIC strategy) {
  this(k.length, f, strategy);
  if (k.length != v.length) throw new IllegalArgumentException("The key array and the value array have different lengths (" + k.length + " and " + v.length + ")");
  for(int i = 0; i < k.length; i++) this.put(k[i], v[i]);
 }
#else
 /** Creates a new hash map using the elements of two parallel arrays.
  *
  * @param k the array of keys of the new hash map.
  * @param v the array of corresponding values in the new hash map.
  * @param f the load factor.
```

```
 * @throws IllegalArgumentException if {@code k} and {@code v} have different lengths.
 */

 public OPEN_HASH_MAP(final KEY_GENERIC_TYPE[] k, final VALUE_GENERIC_TYPE[] v, final float f) {
  this(k.length, f);
  if (k.length != v.length) throw new IllegalArgumentException("The key array and the value array have different lengths (" + k.length + " and " + v.length + ")");
  for(int i = 0; i < k.length; i++) this.put(k[i], v[i]);
 }
#endif

#ifdef Custom
 /** Creates a new hash map with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor using the elements of two parallel arrays.
  *
  * @param k the array of keys of the new hash map.
  * @param v the array of corresponding values in the new hash map.
  * @param strategy the strategy.
  * @throws IllegalArgumentException if {@code k} and {@code v} have different lengths.
  */

 public OPEN_HASH_MAP(final KEY_GENERIC_TYPE[] k, final VALUE_GENERIC_TYPE[] v, final STRATEGY KEY_SUPER_GENERIC strategy) {
  this(k, v, DEFAULT_LOAD_FACTOR, strategy);
 }
#else
 /** Creates a new hash map with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor using the elements of two parallel arrays.
  *
  * @param k the array of keys of the new hash map.
  * @param v the array of corresponding values in the new hash map.
  * @throws IllegalArgumentException if {@code k} and {@code v} have different lengths.
  */

 public OPEN_HASH_MAP(final KEY_GENERIC_TYPE[] k, final VALUE_GENERIC_TYPE[] v) {
  this(k, v, DEFAULT_LOAD_FACTOR);
 }
#endif



#ifdef Custom
 /** Returns the hashing strategy.
  *
  * @return the hashing strategy of this custom hash map.
  */

 public STRATEGY KEY_SUPER_GENERIC strategy() {
```

```
 return strategy;
 }
#endif

 private int realSize() {
  return containsNullKey ? size - 1 : size;
 }

 private void ensureCapacity(final int capacity) {
  final int needed = arraySize(capacity, f);
  if (needed > n) rehash(needed);
 }

 private void tryCapacity(final long capacity) {
  final int needed = (int)Math.min(1 << 30, Math.max(2, HashCommon.nextPowerOfTwo((long)Math.ceil(capacity / f))));
  if (needed > n) rehash(needed);
 }

 private VALUE_GENERIC_TYPE removeEntry(final int pos) {
  final VALUE_GENERIC_TYPE oldValue = value[pos];
#if VALUES_REFERENCE
  value[pos] = null;
#endif
  size--;
#ifdef Linked
  fixPointers(pos);
#endif

  shiftKeys(pos);
  if (n > minN && size < maxFill / 4 && n > DEFAULT_INITIAL_SIZE) rehash(n / 2);
  return oldValue;
 }

 private VALUE_GENERIC_TYPE removeNullEntry() {
  containsNullKey = false;
#if KEYS_REFERENCE
  key[n] = null;
#endif
  final VALUE_GENERIC_TYPE oldValue = value[n];
#if VALUES_REFERENCE
  value[n] = null;
#endif
  size--;
#ifdef Linked
  fixPointers(n);
#endif
  if (n > minN && size < maxFill / 4 && n > DEFAULT_INITIAL_SIZE) rehash(n / 2);
```

```
 return oldValue;
 }


 @Override
 public void putAll(Map<? extends KEY_GENERIC_CLASS,? extends VALUE_GENERIC_CLASS> m) {
  if (f <= .5) ensureCapacity(m.size()); // The resulting map will be sized for m.size() elements
  else tryCapacity(size() + m.size()); // The resulting map will be tentatively sized for size() + m.size() elements
  super.putAll(m);
 }

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 private int find(final KEY_GENERIC_TYPE k) {
  if (KEY_EQUALS_NULL(k)) return containsNullKey ? n : -(n + 1);

  KEY_GENERIC_TYPE curr;
  final KEY_GENERIC_TYPE[] key = this.key;
  int pos;

  // The starting point.
  if (KEY_IS_NULL(curr = key[pos = KEY2INTHASH(k) & mask])) return -(pos + 1);
  if (KEY_EQUALS_NOT_NULL(k, curr)) return pos;
  // There's always an unused entry.
  while(true) {
   if (KEY_IS_NULL(curr = key[pos = (pos + 1) & mask])) return -(pos + 1);
   if (KEY_EQUALS_NOT_NULL(k, curr)) return pos;
  }
 }


 private void insert(final int pos, final KEY_GENERIC_TYPE k, final VALUE_GENERIC_TYPE v) {
  if (pos == n) containsNullKey = true;
  key[pos] = k;
  value[pos] = v;
 #ifdef Linked
  if (size == 0) {
   first = last = pos;
   // Special case of SET_UPPER_LOWER(link[pos], -1, -1);
   link[pos] = -1L;
  }
  else {
   SET_NEXT(link[last], pos);
   SET_UPPER_LOWER(link[pos], last, -1);
   last = pos;
  }
 #endif

  if (size++ >= maxFill) rehash(arraySize(size + 1, f));
```

```
 if (ASSERTS) checkTable();
}


@Override
public VALUE_GENERIC_TYPE put(final KEY_GENERIC_TYPE k, final VALUE_GENERIC_TYPE v) {
 final int pos = find(k);
 if (pos < 0) {
  insert(-pos - 1, k, v);
  return defRetValue;
 }
 final VALUE_GENERIC_TYPE oldValue = value[pos];
 value[pos] = v;
 return oldValue;
}

#if VALUE_CLASS_Byte || VALUE_CLASS_Short || VALUE_CLASS_Char || VALUE_CLASS_Integer ||
VALUE_CLASS_Long || VALUE_CLASS_Float || VALUE_CLASS_Double

 private VALUE_GENERIC_TYPE addToValue(final int pos, final VALUE_GENERIC_TYPE incr) {
  final VALUE_GENERIC_TYPE oldValue = value[pos];
#if VALUE_CLASS_Byte || VALUE_CLASS_Short || VALUE_CLASS_Char
  value[pos] = (VALUE_TYPE)(oldValue + incr);
#else
  value[pos] = oldValue + incr;
#endif
  return oldValue;
 }

 /** Adds an increment to value currently associated with a key.
  *
  * <p>Note that this method respects the {@linkplain #defaultReturnValue() default return value} semantics: when
  * called with a key that does not currently appears in the map, the key
  * will be associated with the default return value plus
  * the given increment.
  *
  * @param k the key.
  * @param incr the increment.
  * @return the old value, or the {@linkplain #defaultReturnValue() default return value} if no value was present for
the given key.
  */
 public VALUE_GENERIC_TYPE addTo(final KEY_GENERIC_TYPE k, final VALUE_GENERIC_TYPE incr) {
  int pos;

  if (KEY_EQUALS_NULL(k)) {
   if (containsNullKey) return addToValue(n, incr);
   pos = n;
   containsNullKey = true;
```

```
  }
  else {
   KEY_GENERIC_TYPE curr;
   final KEY_GENERIC_TYPE[] key = this.key;

   // The starting point.
   if (! KEY_IS_NULL(curr = key[pos = KEY2INTHASH(k) & mask])) {
    if (KEY_EQUALS_NOT_NULL(curr, k)) return addToValue(pos, incr);
    while(! KEY_IS_NULL(curr = key[pos = (pos + 1) & mask]))
     if (KEY_EQUALS_NOT_NULL(curr, k)) return addToValue(pos, incr);
   }
  }

  key[pos] = k;

#if VALUE_CLASS_Byte || VALUE_CLASS_Short || VALUE_CLASS_Char
  value[pos] = (VALUE_TYPE)(defRetValue + incr);
#else
  value[pos] = defRetValue + incr;
#endif

#ifdef Linked
  if (size == 0) {
   first = last = pos;
   // Special case of SET_UPPER_LOWER(link[pos], -1, -1);
   link[pos] = -1L;
  }
  else {
   SET_NEXT(link[last], pos);
   SET_UPPER_LOWER(link[pos], last, -1);
   last = pos;
  }
#endif

  if (size++ >= maxFill) rehash(arraySize(size + 1, f));
  if (ASSERTS) checkTable();
  return defRetValue;
 }

#endif

 /** Shifts left entries with the specified hash code, starting at the specified position,
  * and empties the resulting free entry.
  *
  * @param pos a starting position.
  */
 protected final void shiftKeys(int pos) {
  // Shift entries with the same hash.
```

```
 int last, slot;
 KEY_GENERIC_TYPE curr;
 final KEY_GENERIC_TYPE[] key = this.key;

 for(;;) {
  pos = ((last = pos) + 1) & mask;

  for(;;) {
   if (KEY_IS_NULL(curr = key[pos])) {
    key[last] = KEY_NULL;
#if VALUES_REFERENCE
    value[last] = null;
#endif
    return;
   }
   slot = KEY2INTHASH(curr) & mask;
   if (last <= pos ? last >= slot || slot > pos : last >= slot && slot > pos) break;
   pos = (pos + 1) & mask;
  }

  key[last] = curr;
  value[last] = value[pos];
#ifdef Linked
  fixPointers(pos, last);
#endif
 }
}

@Override
SUPPRESS_WARNINGS_KEY_UNCHECKED
public VALUE_GENERIC_TYPE REMOVE_VALUE(final KEY_TYPE k) {
 if (KEY_EQUALS_NULL(KEY_GENERIC_CAST k)) {
  if (containsNullKey) return removeNullEntry();
  return defRetValue;
 }

 KEY_GENERIC_TYPE curr;
 final KEY_GENERIC_TYPE[] key = this.key;
 int pos;

 // The starting point.
 if (KEY_IS_NULL(curr = key[pos = KEY2INTHASH_CAST(k) & mask])) return defRetValue;
 if (KEY_EQUALS_NOT_NULL_CAST(k, curr)) return removeEntry(pos);
 while(true) {
  if (KEY_IS_NULL(curr = key[pos = (pos + 1) & mask])) return defRetValue;
  if (KEY_EQUALS_NOT_NULL_CAST(k, curr)) return removeEntry(pos);
 }
}
```

```
#ifdef Linked

 private VALUE_GENERIC_TYPE setValue(final int pos, final VALUE_GENERIC_TYPE v) {
  final VALUE_GENERIC_TYPE oldValue = value[pos];
  value[pos] = v;
  return oldValue;
 }

 /** Removes the mapping associated with the first key in iteration order.
  * @return the value previously associated with the first key in iteration order.
  * @throws NoSuchElementException is this map is empty.
  */
 public VALUE_GENERIC_TYPE REMOVE_FIRST_VALUE() {
  if (size == 0) throw new NoSuchElementException();
  final int pos = first;
  // Abbreviated version of fixPointers(pos)
  first = GET_NEXT(link[pos]);
  if (0 <= first) {
   // Special case of SET_PREV(link[first], -1)
   link[first] |= (-1 & 0xFFFFFFFFL) << 32;
  }
  size--;
  final VALUE_GENERIC_TYPE v = value[pos];
  if (pos == n) {
   containsNullKey = false;
#if KEYS_REFERENCE
   key[n] = null;
#endif
#if VALUES_REFERENCE
   value[n] = null;
#endif
  }
  else shiftKeys(pos);
  if (n > minN && size < maxFill / 4 && n > DEFAULT_INITIAL_SIZE) rehash(n / 2);
  return v;
 }

 /** Removes the mapping associated with the last key in iteration order.
  * @return the value previously associated with the last key in iteration order.
  * @throws NoSuchElementException is this map is empty.
  */
 public VALUE_GENERIC_TYPE REMOVE_LAST_VALUE() {
  if (size == 0) throw new NoSuchElementException();
  final int pos = last;
  // Abbreviated version of fixPointers(pos)
  last = GET_PREV(link[pos]);
```

```
 if (0 <= last) {
  // Special case of SET_NEXT(link[last], -1)
  link[last] |= -1 & 0xFFFFFFFFL;
 }
 size--;
 final VALUE_GENERIC_TYPE v = value[pos];
 if (pos == n) {
  containsNullKey = false;
#if KEYS_REFERENCE
  key[n] = null;
#endif
#if VALUES_REFERENCE
  value[n] = null;
#endif
 }
 else shiftKeys(pos);
 if (n > minN && size < maxFill / 4 && n > DEFAULT_INITIAL_SIZE) rehash(n / 2);
 return v;
}

private void moveIndexToFirst(final int i) {
 if (size == 1 || first == i) return;
 if (last == i) {
  last = GET_PREV(link[i]);
  // Special case of SET_NEXT(link[last], -1);
  link[last] |= -1 & 0xFFFFFFFFL;
 }
 else {
  final long linki = link[i];
  final int prev = GET_PREV(linki);
  final int next = GET_NEXT(linki);
  COPY_NEXT(link[prev], linki);
  COPY_PREV(link[next], linki);
 }
 SET_PREV(link[first], i);
 SET_UPPER_LOWER(link[i], -1, first);
 first = i;
}

private void moveIndexToLast(final int i) {
 if (size == 1 ||  last == i) return;
 if (first == i) {
  first = GET_NEXT(link[i]);
  // Special case of SET_PREV(link[first], -1);
  link[first] |= (-1 & 0xFFFFFFFFL) << 32;
 }
 else {
  final long linki = link[i];
```

```
  final int prev = GET_PREV(linki);
  final int next = GET_NEXT(linki);
  COPY_NEXT(link[prev], linki);
  COPY_PREV(link[next], linki);
  }
  SET_NEXT(link[last], i);
  SET_UPPER_LOWER(link[i], last, -1);
  last = i;
  }

  /** Returns the value to which the given key is mapped; if the key is present, it is moved to the first position of the iteration order.
  *
  * @param k the key.
  * @return the corresponding value, or the {@linkplain #defaultReturnValue() default return value} if no value was present for the given key.
  */
  public VALUE_GENERIC_TYPE getAndMoveToFirst(final KEY_GENERIC_TYPE k) {
  if (KEY_EQUALS_NULL(k)) {
   if (containsNullKey) {
    moveIndexToFirst(n);
    return value[n];
   }

   return defRetValue;
  }

  KEY_GENERIC_TYPE curr;
  final KEY_GENERIC_TYPE[] key = this.key;
  int pos;

  // The starting point.
  if (KEY_IS_NULL(curr = key[pos = KEY2INTHASH(k) & mask])) return defRetValue;
  if (KEY_EQUALS_NOT_NULL(k, curr)) {
   moveIndexToFirst(pos);
   return value[pos];
  }

  // There's always an unused entry.
  while(true) {
   if (KEY_IS_NULL(curr = key[pos = (pos + 1) & mask])) return defRetValue;
   if (KEY_EQUALS_NOT_NULL(k, curr))  {
    moveIndexToFirst(pos);
    return value[pos];
   }
  }
 }
```

```
 /** Returns the value to which the given key is mapped; if the key is present, it is moved to the last position of the iteration order.
  *
  * @param k the key.
  * @return the corresponding value, or the {@linkplain #defaultReturnValue() default return value} if no value was present for the given key.
  */
 public VALUE_GENERIC_TYPE getAndMoveToLast(final KEY_GENERIC_TYPE k) {
  if (KEY_EQUALS_NULL(k)) {
   if (containsNullKey) {
    moveIndexToLast(n);
    return value[n];
   }

   return defRetValue;
  }

  KEY_GENERIC_TYPE curr;
  final KEY_GENERIC_TYPE[] key = this.key;
  int pos;

  // The starting point.
  if (KEY_IS_NULL(curr = key[pos = KEY2INTHASH(k) & mask])) return defRetValue;
  if (KEY_EQUALS_NOT_NULL(k, curr)) {
   moveIndexToLast(pos);
   return value[pos];
  }

  // There's always an unused entry.
  while(true) {
   if (KEY_IS_NULL(curr = key[pos = (pos + 1) & mask])) return defRetValue;
   if (KEY_EQUALS_NOT_NULL(k, curr))  {
    moveIndexToLast(pos);
    return value[pos];
   }
  }
 }

 /** Adds a pair to the map; if the key is already present, it is moved to the first position of the iteration order.
  *
  * @param k the key.
  * @param v the value.
  * @return the old value, or the {@linkplain #defaultReturnValue() default return value} if no value was present for the given key.
  */
 public VALUE_GENERIC_TYPE putAndMoveToFirst(final KEY_GENERIC_TYPE k, final VALUE_GENERIC_TYPE v) {
  int pos;
```

```
if (KEY_EQUALS_NULL(k)) {
 if (containsNullKey) {
  moveIndexToFirst(n);
  return setValue(n, v);
 }
 containsNullKey = true;
 pos = n;
}
else {
 KEY_GENERIC_TYPE curr;
 final KEY_GENERIC_TYPE[] key = this.key;

 // The starting point.
 if (! KEY_IS_NULL(curr = key[pos = KEY2INTHASH(k) & mask])) {
  if (KEY_EQUALS_NOT_NULL(curr, k)) {
   moveIndexToFirst(pos);
   return setValue(pos, v);
  }
  while(! KEY_IS_NULL(curr = key[pos = (pos + 1) & mask]))
   if (KEY_EQUALS_NOT_NULL(curr, k)) {
    moveIndexToFirst(pos);
    return setValue(pos, v);
   }
 }
}

key[pos] = k;
value[pos] = v;

if (size == 0) {
 first = last = pos;
 // Special case of SET_UPPER_LOWER(link[pos], -1, -1);
 link[pos] = -1L;
}
else {
 SET_PREV(link[first], pos);
 SET_UPPER_LOWER(link[pos], -1, first);
 first = pos;
}

if (size++ >= maxFill) rehash(arraySize(size, f));
if (ASSERTS) checkTable();
return defRetValue;
}

/** Adds a pair to the map; if the key is already present, it is moved to the last position of the iteration order.
 *
```

```
 * @param k the key.
 * @param v the value.
 * @return the old value, or the {@linkplain #defaultReturnValue() default return value} if no value was present for
the given key.
 */
 public VALUE_GENERIC_TYPE putAndMoveToLast(final KEY_GENERIC_TYPE k, final
VALUE_GENERIC_TYPE v) {
 int pos;

 if (KEY_EQUALS_NULL(k)) {
  if (containsNullKey) {
   moveIndexToLast(n);
   return setValue(n, v);
  }
  containsNullKey = true;
  pos = n;
 }
 else {
  KEY_GENERIC_TYPE curr;
  final KEY_GENERIC_TYPE[] key = this.key;

  // The starting point.
  if (! KEY_IS_NULL(curr = key[pos = KEY2INTHASH(k) & mask])) {
   if (KEY_EQUALS_NOT_NULL(curr, k)) {
    moveIndexToLast(pos);
    return setValue(pos, v);
   }
   while(! KEY_IS_NULL(curr = key[pos = (pos + 1) & mask]))
    if (KEY_EQUALS_NOT_NULL(curr, k)) {
     moveIndexToLast(pos);
     return setValue(pos, v);
    }
  }
 }

 key[pos] = k;
 value[pos] = v;

 if (size == 0) {
  first = last = pos;
  // Special case of SET_UPPER_LOWER(link[pos], -1, -1);
  link[pos] = -1L;
 }
 else {
  SET_NEXT(link[last], pos);
  SET_UPPER_LOWER(link[pos], last, -1);
  last = pos;
 }
```

```
 if (size++ >= maxFill) rehash(arraySize(size, f));
 if (ASSERTS) checkTable();
 return defRetValue;
 }

#endif

 @Override
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public VALUE_GENERIC_TYPE GET_VALUE(final KEY_TYPE k) {
 if (KEY_EQUALS_NULL(KEY_GENERIC_CAST k)) return containsNullKey ? value[n] : defRetValue;

 KEY_GENERIC_TYPE curr;
 final KEY_GENERIC_TYPE[] key = this.key;
 int pos;

 // The starting point.
 if (KEY_IS_NULL(curr = key[pos = KEY2INTHASH_CAST(k) & mask])) return defRetValue;
 if (KEY_EQUALS_NOT_NULL_CAST(k, curr)) return value[pos];
 // There's always an unused entry.
 while(true) {
  if (KEY_IS_NULL(curr = key[pos = (pos + 1) & mask])) return defRetValue;
  if (KEY_EQUALS_NOT_NULL_CAST(k, curr)) return value[pos];
 }
 }

 @Override
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public boolean containsKey(final KEY_TYPE k) {
 if (KEY_EQUALS_NULL(KEY_GENERIC_CAST k)) return containsNullKey;

 KEY_GENERIC_TYPE curr;
 final KEY_GENERIC_TYPE[] key = this.key;
 int pos;

 // The starting point.
 if (KEY_IS_NULL(curr = key[pos = KEY2INTHASH_CAST(k) & mask])) return false;
 if (KEY_EQUALS_NOT_NULL_CAST(k, curr)) return true;
 // There's always an unused entry.
 while(true) {
  if (KEY_IS_NULL(curr = key[pos = (pos + 1) & mask])) return false;
  if (KEY_EQUALS_NOT_NULL_CAST(k, curr)) return true;
 }
 }



 @Override
```

```
 public boolean containsValue(final VALUE_TYPE v) {
  final VALUE_GENERIC_TYPE value[] = this.value;
  final KEY_GENERIC_TYPE key[] = this.key;
  if (containsNullKey && VALUE_EQUALS(value[n], v)) return true;
  for(int i = n; i-- != 0;) if (! KEY_IS_NULL(key[i]) && VALUE_EQUALS(value[i], v)) return true;
  return false;
 }


#if KEYS_PRIMITIVE || VALUES_PRIMITIVE

 /** {@inheritDoc} */
 @Override
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public VALUE_GENERIC_TYPE getOrDefault(final KEY_TYPE k, final VALUE_GENERIC_TYPE
defaultValue) {
  if (KEY_EQUALS_NULL(KEY_GENERIC_CAST k)) return containsNullKey ? value[n] : defaultValue;

  KEY_GENERIC_TYPE curr;
  final KEY_GENERIC_TYPE[] key = this.key;
  int pos;

  // The starting point.
  if (KEY_IS_NULL(curr = key[pos = KEY2INTHASH_CAST(k) & mask])) return defaultValue;
  if (KEY_EQUALS_NOT_NULL_CAST(k, curr)) return value[pos];
  // There's always an unused entry.
  while(true) {
   if (KEY_IS_NULL(curr = key[pos = (pos + 1) & mask])) return defaultValue;
   if (KEY_EQUALS_NOT_NULL_CAST(k, curr)) return value[pos];
  }
 }

 /** {@inheritDoc} */
 @Override
 public VALUE_GENERIC_TYPE putIfAbsent(final KEY_GENERIC_TYPE k, final VALUE_GENERIC_TYPE
v) {
  final int pos = find(k);
  if (pos >= 0) return value[pos];
  insert(-pos - 1, k, v);
  return defRetValue;
 }

 /** {@inheritDoc} */
 @Override
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public boolean remove(final KEY_TYPE k, final VALUE_TYPE v) {
  if (KEY_EQUALS_NULL(KEY_GENERIC_CAST k)) {
   if (containsNullKey && VALUE_EQUALS(v, value[n])) {
```

```
   removeNullEntry();
   return true;
  }
  return false;
 }

 KEY_GENERIC_TYPE curr;
 final KEY_GENERIC_TYPE[] key = this.key;
 int pos;

 // The starting point.
 if (KEY_IS_NULL(curr = key[pos = KEY2INTHASH_CAST(k) & mask])) return false;
 if (KEY_EQUALS_NOT_NULL_CAST(k, curr) && VALUE_EQUALS(v, value[pos])) {
  removeEntry(pos);
  return true;
 }
 while(true) {
  if (KEY_IS_NULL(curr = key[pos = (pos + 1) & mask])) return false;
  if (KEY_EQUALS_NOT_NULL_CAST(k, curr) && VALUE_EQUALS(v, value[pos])) {
   removeEntry(pos);
   return true;
  }
 }
}

/** {@inheritDoc} */
@Override
public boolean replace(final KEY_GENERIC_TYPE k, final VALUE_GENERIC_TYPE oldValue, final
VALUE_GENERIC_TYPE v) {
 final int pos = find(k);
 if (pos < 0 || ! VALUE_EQUALS(oldValue, value[pos])) return false;
 value[pos] = v;
 return true;
}

/** {@inheritDoc} */
@Override
public VALUE_GENERIC_TYPE replace(final KEY_GENERIC_TYPE k, final VALUE_GENERIC_TYPE v) {
 final int pos = find(k);
 if (pos < 0) return defRetValue;
 final VALUE_GENERIC_TYPE oldValue = value[pos];
 value[pos] = v;
 return oldValue;
}

#ifdef JDK_PRIMITIVE_FUNCTION

/** {@inheritDoc} */
```

```
 @Override
 public VALUE_GENERIC_TYPE COMPUTE_IF_ABSENT_JDK(final KEY_GENERIC_TYPE k, final
JDK_PRIMITIVE_FUNCTION KEY_SUPER_GENERIC VALUE_EXTENDS_GENERIC mappingFunction) {
  java.util.Objects.requireNonNull(mappingFunction);
  final int pos = find(k);
  if (pos >= 0) return value[pos];
  final VALUE_GENERIC_TYPE newValue =
VALUE_NARROWING(mappingFunction.JDK_PRIMITIVE_FUNCTION_APPLY(k));
  insert(-pos -1, k, newValue);
  return newValue;
 }

#endif

#if KEYS_PRIMITIVE && VALUES_PRIMITIVE

 /** {@inheritDoc} */
 @Override
 public VALUE_GENERIC_TYPE COMPUTE_IF_ABSENT_NULLABLE(final KEY_GENERIC_TYPE k, final
JDK_KEY_TO_GENERIC_FUNCTION<? extends VALUE_GENERIC_CLASS> mappingFunction) {
  java.util.Objects.requireNonNull(mappingFunction);
  final int pos = find(k);
  if (pos >= 0) return value[pos];
  final VALUE_GENERIC_CLASS newValue = mappingFunction.apply(k);
  if (newValue == null) return defRetValue;
  final VALUE_GENERIC_TYPE v = VALUE_CLASS2TYPE(newValue);
  insert(-pos - 1, k, v);
  return v;
 }

#endif

 /** {@inheritDoc} */
 @Override
 public VALUE_GENERIC_TYPE COMPUTE_IF_PRESENT(final KEY_GENERIC_TYPE k, final
java.util.function.BiFunction<? super KEY_GENERIC_CLASS, ? super VALUE_GENERIC_CLASS, ? extends
VALUE_GENERIC_CLASS> remappingFunction) {
  java.util.Objects.requireNonNull(remappingFunction);
  final int pos = find(k);
  if (pos < 0) return defRetValue;
  final VALUE_GENERIC_CLASS newValue = remappingFunction.apply(KEY2OBJ(k),
VALUE2OBJ(value[pos]));
  if (newValue == null) {
   if (KEY_EQUALS_NULL(k)) removeNullEntry();
   else removeEntry(pos);
   return defRetValue;
  }
  return value[pos] = VALUE_CLASS2TYPE(newValue);
```

```
 }

 /** {@inheritDoc} */
 @Override
 public VALUE_GENERIC_TYPE COMPUTE(final KEY_GENERIC_TYPE k, final
java.util.function.BiFunction<? super KEY_GENERIC_CLASS, ? super VALUE_GENERIC_CLASS, ? extends
VALUE_GENERIC_CLASS> remappingFunction) {
  java.util.Objects.requireNonNull(remappingFunction);
  final int pos = find(k);
  final VALUE_GENERIC_CLASS newValue = remappingFunction.apply(KEY2OBJ(k), pos >= 0 ?
VALUE2OBJ(value[pos]) : null);
  if (newValue == null) {
   if (pos >= 0) {
    if (KEY_EQUALS_NULL(k)) removeNullEntry();
    else removeEntry(pos);
   }
   return defRetValue;
  }

  VALUE_GENERIC_TYPE newVal = VALUE_CLASS2TYPE(newValue);
  if (pos < 0) {
   insert(-pos - 1, k, newVal);
   return newVal;
  }

  return value[pos] = newVal;
 }

 /** {@inheritDoc} */
 @Override
 public VALUE_GENERIC_TYPE MERGE(final KEY_GENERIC_TYPE k, final VALUE_GENERIC_TYPE v,
final java.util.function.BiFunction<? super VALUE_GENERIC_CLASS, ? super VALUE_GENERIC_CLASS, ?
extends VALUE_GENERIC_CLASS> remappingFunction) {
  java.util.Objects.requireNonNull(remappingFunction);

  final int pos = find(k);
#if VALUES_PRIMITIVE
  if (pos < 0) {
#else
  if (pos < 0 || value[pos] == null) {
   if (v == null) return defRetValue;
#endif
   insert(-pos - 1, k, v);
   return v;
  }

  final VALUE_GENERIC_CLASS newValue = remappingFunction.apply(VALUE2OBJ(value[pos]),
VALUE2OBJ(v));
```

```
 if (newValue == null) {
  if (KEY_EQUALS_NULL(k)) removeNullEntry();
  else removeEntry(pos);
  return defRetValue;
  }

  return value[pos] = VALUE_CLASS2TYPE(newValue);
 }


#endif

 /* Removes all elements from this map.
  *
  * <p>To increase object reuse, this method does not change the table size.
  * If you want to reduce the table size, you must use {@link #trim()}.
  *
  */
 @Override
 public void clear() {
  if (size == 0) return;
  size = 0;
  containsNullKey = false;

  Arrays.fill(key, KEY_NULL);
#if VALUES_REFERENCE
  Arrays.fill(value, null);
#endif

#ifdef Linked
  first = last = -1;
#endif
 }

 @Override
 public int size() {
  return size;
 }

 @Override
 public boolean isEmpty() {
  return size == 0;
 }


 /** The entry class for a hash map does not record key and value, but
  * rather the position in the hash table of the corresponding entry. This
  * is necessary so that calls to {@link java.util.Map.Entry#setValue(Object)} are reflected in
```

```
 * the map */

 final class MapEntry implements MAP.Entry KEY_VALUE_GENERIC, Map.Entry<KEY_GENERIC_CLASS,
VALUE_GENERIC_CLASS> {
  // The table index this entry refers to, or -1 if this entry has been deleted.
  int index;

  MapEntry(final int index) {
   this.index = index;
  }

  MapEntry() {}

  @Override
  public KEY_GENERIC_TYPE ENTRY_GET_KEY() {
     return key[index];
  }

  @Override
  public VALUE_GENERIC_TYPE ENTRY_GET_VALUE() {
   return value[index];
  }

  @Override
  public VALUE_GENERIC_TYPE setValue(final VALUE_GENERIC_TYPE v) {
   final VALUE_GENERIC_TYPE oldValue = value[index];
   value[index] = v;
   return oldValue;
  }

#if KEYS_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public KEY_GENERIC_CLASS getKey() {
   return KEY2OBJ(key[index]);
  }
#endif

#if VALUES_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public VALUE_GENERIC_CLASS getValue() {
   return VALUE2OBJ(value[index]);
  }
```

```
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public VALUE_GENERIC_CLASS setValue(final VALUE_GENERIC_CLASS v) {
   return VALUE2OBJ(setValue(VALUE_CLASS2TYPE(v)));
  }
#endif

  @SuppressWarnings("unchecked")
  @Override
  public boolean equals(final Object o) {
   if (!(o instanceof Map.Entry)) return false;
   Map.Entry<KEY_GENERIC_CLASS, VALUE_GENERIC_CLASS> e = (Map.Entry<KEY_GENERIC_CLASS,
VALUE_GENERIC_CLASS>)o;

   return KEY_EQUALS(key[index], KEY_CLASS2TYPE(e.getKey())) && VALUE_EQUALS(value[index],
VALUE_CLASS2TYPE(e.getValue()));
  }

  @Override
  public int hashCode() {
   return KEY2JAVAHASH(key[index]) ^ VALUE2JAVAHASH(value[index]);
  }

  @Override
  public String toString() {
   return key[index] + "=>" + value[index];
  }
 }


#ifdef Linked

 /** Modifies the {@link #link} vector so that the given entry is removed.
  * This method will complete in constant time.
  *
  * @param i the index of an entry.
  */
 protected void fixPointers(final int i) {
  if (size == 0) {
   first = last = -1;
   return;
  }
  if (first == i) {
   first = GET_NEXT(link[i]);
   if (0 <= first) {
```

```
  // Special case of SET_PREV(link[first], -1)
  link[first] |= (-1 & 0xFFFFFFFFL) << 32;
 }
 return;
}
if (last == i) {
 last = GET_PREV(link[i]);
 if (0 <= last) {
  // Special case of SET_NEXT(link[last], -1)
  link[last] |= -1 & 0xFFFFFFFFL;
 }
 return;
}
final long linki = link[i];
final int prev = GET_PREV(linki);
final int next = GET_NEXT(linki);
COPY_NEXT(link[prev], linki);
COPY_PREV(link[next], linki);
}


/** Modifies the {@link #link} vector for a shift from s to d.
 * <p>This method will complete in constant time.
 *
 * @param s the source position.
 * @param d the destination position.
 */
protected void fixPointers(int s, int d) {
 if (size == 1) {
  first = last = d;
  // Special case of SET_UPPER_LOWER(link[d], -1, -1)
  link[d] = -1L;
  return;
 }
 if (first == s) {
  first = d;
  SET_PREV(link[GET_NEXT(link[s])], d);
  link[d] = link[s];
  return;
 }
 if (last == s) {
  last = d;
  SET_NEXT(link[GET_PREV(link[s])], d);
  link[d] = link[s];
  return;
 }
 final long links = link[s];
 final int prev = GET_PREV(links);
```

```
 final int next = GET_NEXT(links);
 SET_NEXT(link[prev], d);
 SET_PREV(link[next], d);
 link[d] = links;
}


/** Returns the first key of this map in iteration order.
 *
 * @return the first key in iteration order.
 */
@Override
public KEY_GENERIC_TYPE FIRST_KEY() {
 if (size == 0) throw new NoSuchElementException();
 return key[first];
}


/** Returns the last key of this map in iteration order.
 *
 * @return the last key in iteration order.
 */
@Override
public KEY_GENERIC_TYPE LAST_KEY() {
 if (size == 0) throw new NoSuchElementException();
 return key[last];
}

/** {@inheritDoc}
 * <p>This implementation just throws an {@link UnsupportedOperationException}.*/
@Override
public SORTED_MAP KEY_VALUE_GENERIC tailMap(KEY_GENERIC_TYPE from) { throw new
UnsupportedOperationException(); }

/** {@inheritDoc}
 * <p>This implementation just throws an {@link UnsupportedOperationException}.*/
@Override
public SORTED_MAP KEY_VALUE_GENERIC headMap(KEY_GENERIC_TYPE to) { throw new
UnsupportedOperationException(); }

/** {@inheritDoc}
 * <p>This implementation just throws an {@link UnsupportedOperationException}.*/
@Override
public SORTED_MAP KEY_VALUE_GENERIC subMap(KEY_GENERIC_TYPE from, KEY_GENERIC_TYPE
to) { throw new UnsupportedOperationException(); }

/** {@inheritDoc}
 * <p>This implementation just returns {@code null}.*/
```

```
 @Override
 public KEY_COMPARATOR KEY_SUPER_GENERIC comparator() { return null; }


 /** A list iterator over a linked map.
  *
  * <p>This class provides a list iterator over a linked hash map. The constructor runs in constant time.
  */
 private class MapIterator {
  /** The entry that will be returned by the next call to {@link java.util.ListIterator#previous()} (or {@code null} if no previous entry exists). */
  int prev = -1;
  /** The entry that will be returned by the next call to {@link java.util.ListIterator#next()} (or {@code null} if no next entry exists). */
  int next = -1;
  /** The last entry that was returned (or -1 if we did not iterate or used {@link java.util.Iterator#remove()}). */
  int curr = -1;
  /** The current index (in the sense of a {@link java.util.ListIterator}). Note that this value is not meaningful when this iterator has been created using the nonempty constructor.*/
  int index = -1;

  protected MapIterator() {
   next = first;
   index = 0;
  }

  private MapIterator(final KEY_GENERIC_TYPE from) {
   if (KEY_EQUALS_NULL(from)) {
    if (OPEN_HASH_MAP.this.containsNullKey) {
     next = GET_NEXT(link[n]);
     prev = n;
     return;
    }
    else throw new NoSuchElementException("The key " + from + " does not belong to this map.");
   }

   if (KEY_EQUALS(key[last], from)) {
    prev = last;
    index = size;
    return;
   }

   // The starting point.
   int pos = KEY2INTHASH(from) & mask;

   // There's always an unused entry.
   while(! KEY_IS_NULL(key[pos])) {
    if (KEY_EQUALS_NOT_NULL(key[pos], from)) {
```

```
   // Note: no valid index known.
   next = GET_NEXT(link[pos]);
   prev = pos;
   return;
  }
  pos = (pos + 1) & mask;
 }
 throw new NoSuchElementException("The key " + from + " does not belong to this map.");
}

public boolean hasNext() { return next != -1; }

public boolean hasPrevious() { return prev != -1; }

private final void ensureIndexKnown() {
 if (index >= 0) return;
 if (prev == -1) {
  index = 0;
  return;
 }
 if (next == -1) {
  index = size;
  return;
 }
 int pos = first;
 index = 1;
 while(pos != prev) {
  pos = GET_NEXT(link[pos]);
  index++;
 }
}

public int nextIndex() {
 ensureIndexKnown();
 return index;
}

public int previousIndex() {
 ensureIndexKnown();
 return index - 1;
}

public int nextEntry() {
 if (! hasNext()) throw new NoSuchElementException();

 curr = next;
 next = GET_NEXT(link[curr]);
 prev = curr;
```

```
 if (index >= 0) index++;

 return curr;
}

public int previousEntry() {
 if (! hasPrevious()) throw new NoSuchElementException();

 curr = prev;
 prev = GET_PREV(link[curr]);
 next = curr;

 if (index >= 0) index--;

 return curr;
}

public void remove() {
 ensureIndexKnown();
 if (curr == -1) throw new IllegalStateException();

 if (curr == prev) {
  /* If the last operation was a next(), we are removing an entry that preceeds
     the current index, and thus we must decrement it. */
  index--;
  prev = GET_PREV(link[curr]);
 }
 else
  next = GET_NEXT(link[curr]);

 size--;
 /* Now we manually fix the pointers. Because of our knowledge of next
    and prev, this is going to be faster than calling fixPointers(). */
 if (prev == -1) first = next;
 else
  SET_NEXT(link[prev], next);
 if (next == -1) last = prev;
 else
  SET_PREV(link[next], prev);

 int last, slot, pos = curr;
 curr = -1;

 if (pos == n) {
  OPEN_HASH_MAP.this.containsNullKey = false;
#if KEYS_REFERENCE
  key[n] = null;
```

```
#endif
#if VALUES_REFERENCE
    value[n] = null;
#endif
   }
   else {
    KEY_GENERIC_TYPE curr;
    final KEY_GENERIC_TYPE[] key = OPEN_HASH_MAP.this.key;
    // We have to horribly duplicate the shiftKeys() code because we need to update next/prev.
    for(;;) {
     pos = ((last = pos) + 1) & mask;
     for(;;) {
      if (KEY_IS_NULL(curr = key[pos])) {
       key[last] = KEY_NULL;
#if VALUES_REFERENCE
       value[last] = null;
#endif
       return;
      }
      slot = KEY2INTHASH(curr) & mask;
      if (last <= pos ? last >= slot || slot > pos : last >= slot && slot > pos) break;
      pos = (pos + 1) & mask;
     }

     key[last] = curr;
     value[last] = value[pos];
     if (next == pos) next = last;
     if (prev == pos) prev = last;
     fixPointers(pos, last);
    }
   }
  }

  public int skip(final int n) {
   int i = n;
   while(i-- != 0 && hasNext()) nextEntry();
   return n - i - 1;
  }

  public int back(final int n) {
   int i = n;
   while(i-- != 0 && hasPrevious()) previousEntry();
   return n - i - 1;
  }

  public void set(@SuppressWarnings("unused") MAP.Entry KEY_VALUE_GENERIC ok) {
   throw new UnsupportedOperationException();
  }
```

```
 public void add(@SuppressWarnings("unused") MAP.Entry KEY_VALUE_GENERIC ok) {
  throw new UnsupportedOperationException();
 }
}

private class EntryIterator extends MapIterator implements ObjectListIterator<MAP.Entry
KEY_VALUE_GENERIC> {
 private MapEntry entry;

 public EntryIterator() {}

 public EntryIterator(KEY_GENERIC_TYPE from) {
  super(from);
 }

 @Override
 public MapEntry next() {
  return entry = new MapEntry(nextEntry());
 }

 @Override
 public MapEntry previous() {
  return entry = new MapEntry(previousEntry());
 }

 @Override
 public void remove() {
  super.remove();
  entry.index = -1; // You cannot use a deleted entry.
 }
}

private class FastEntryIterator extends MapIterator implements ObjectListIterator<MAP.Entry
KEY_VALUE_GENERIC> {
 final MapEntry entry = new MapEntry();

 public FastEntryIterator() {}

 public FastEntryIterator(KEY_GENERIC_TYPE from) {
  super(from);
 }

 @Override
 public MapEntry next() {
  entry.index = nextEntry();
  return entry;
 }
```

```
 @Override
 public MapEntry previous() {
  entry.index = previousEntry();
  return entry;
 }
}

#else

 /** An iterator over a hash map. */

 private class MapIterator {
  /** The index of the last entry returned, if positive or zero; initially, {@link #n}. If negative, the last
   entry returned was that of the key of index {@code - pos - 1} from the {@link #wrapped} list. */
  int pos = n;
  /** The index of the last entry that has been returned (more precisely, the value of {@link #pos} if {@link #pos} is positive,
   or {@link Integer#MIN_VALUE} if {@link #pos} is negative). It is -1 if either
   we did not return an entry yet, or the last returned entry has been removed. */
  int last = -1;
  /** A downward counter measuring how many entries must still be returned. */
  int c = size;
  /** A boolean telling us whether we should return the entry with the null key. */
  boolean mustReturnNullKey = OPEN_HASH_MAP.this.containsNullKey;
  /** A lazily allocated list containing keys of entries that have wrapped around the table because of removals. */
  ARRAY_LIST KEY_GENERIC wrapped;

  public boolean hasNext() {
   return c != 0;
  }

  public int nextEntry() {
   if (! hasNext()) throw new NoSuchElementException();

   c--;
   if (mustReturnNullKey) {
    mustReturnNullKey = false;
    return last = n;
   }

   final KEY_GENERIC_TYPE key[] = OPEN_HASH_MAP.this.key;

   for(;;) {
    if (--pos < 0) {
     // We are just enumerating elements from the wrapped list.
     last = Integer.MIN_VALUE;
     final KEY_GENERIC_TYPE k = wrapped.GET_KEY(- pos - 1);
```

```
    int p = KEY2INTHASH(k) & mask;
    while (! KEY_EQUALS_NOT_NULL(k, key[p])) p = (p + 1) & mask;
    return p;
   }
   if (! KEY_IS_NULL(key[pos])) return last = pos;
  }
 }

 /** Shifts left entries with the specified hash code, starting at the specified position,
  * and empties the resulting free entry.
  *
  * @param pos a starting position.
  */
 private void shiftKeys(int pos) {
  // Shift entries with the same hash.
  int last, slot;
  KEY_GENERIC_TYPE curr;
  final KEY_GENERIC_TYPE[] key = OPEN_HASH_MAP.this.key;

  for(;;) {
   pos = ((last = pos) + 1) & mask;

   for(;;) {
    if (KEY_IS_NULL(curr = key[pos])) {
     key[last] = KEY_NULL;
#if VALUES_REFERENCE
     value[last] = null;
#endif
     return;
    }
    slot = KEY2INTHASH(curr) & mask;
    if (last <= pos ? last >= slot || slot > pos : last >= slot && slot > pos) break;
    pos = (pos + 1) & mask;
   }

   if (pos < last) { // Wrapped entry.
    if (wrapped == null) wrapped = new ARRAY_LIST KEY_GENERIC_DIAMOND(2);
    wrapped.add(key[pos]);
   }

   key[last] = curr;
   value[last] = value[pos];
  }
 }

 public void remove() {
  if (last == -1) throw new IllegalStateException();
  if (last == n) {
```

```
    containsNullKey = false;
#if KEYS_REFERENCE
    key[n] = null;
#endif
#if VALUES_REFERENCE
    value[n] = null;
#endif
   }
   else if (pos >= 0) shiftKeys(last);
   else {
    // We're removing wrapped entries.
#if KEYS_REFERENCE
    OPEN_HASH_MAP.this.REMOVE_VALUE(wrapped.set(- pos - 1, null));
#else
    OPEN_HASH_MAP.this.REMOVE_VALUE(wrapped.GET_KEY(- pos - 1));
#endif
    last = -1; // Note that we must not decrement size
    return;
   }

   size--;
   last = -1; // You can no longer remove this entry.
   if (ASSERTS) checkTable();
  }

  public int skip(final int n) {
   int i = n;
   while(i-- != 0 && hasNext()) nextEntry();
   return n - i - 1;
  }
 }


 private class EntryIterator extends MapIterator implements ObjectIterator<MAP.Entry KEY_VALUE_GENERIC>
{
  private MapEntry entry;

  @Override
  public MapEntry next() {
   return entry = new MapEntry(nextEntry());
  }

  @Override
  public void remove() {
   super.remove();
   entry.index = -1; // You cannot use a deleted entry.
  }
 }
```

```
 private class FastEntryIterator extends MapIterator implements ObjectIterator<MAP.Entry
KEY_VALUE_GENERIC> {
  private final MapEntry entry = new MapEntry();

  @Override
  public MapEntry next() {
   entry.index = nextEntry();
   return entry;
  }
 }

#endif

#ifdef Linked
 private final class MapEntrySet extends AbstractObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC>
implements FastSortedEntrySet KEY_VALUE_GENERIC {

  @Override
  public ObjectBidirectionalIterator<MAP.Entry KEY_VALUE_GENERIC> iterator() { return new EntryIterator(); }

  @Override
  public Comparator<? super MAP.Entry KEY_VALUE_GENERIC> comparator() { return null; }

  @Override
  public ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> subSet(MAP.Entry KEY_VALUE_GENERIC
fromElement, MAP.Entry KEY_VALUE_GENERIC toElement) { throw new UnsupportedOperationException(); }

  @Override
  public ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> headSet(MAP.Entry KEY_VALUE_GENERIC
toElement) { throw new UnsupportedOperationException(); }

  @Override
  public ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> tailSet(MAP.Entry KEY_VALUE_GENERIC
fromElement) { throw new UnsupportedOperationException(); }

  @Override
  public MAP.Entry KEY_VALUE_GENERIC first() {
   if (size == 0) throw new NoSuchElementException();
   return new MapEntry(OPEN_HASH_MAP.this.first);
  }

  @Override
  public MAP.Entry KEY_VALUE_GENERIC last() {
   if (size == 0) throw new NoSuchElementException();
   return new MapEntry(OPEN_HASH_MAP.this.last);
  }
```

```
#else
 private final class MapEntrySet extends AbstractObjectSet<MAP.Entry KEY_VALUE_GENERIC> implements
FastEntrySet KEY_VALUE_GENERIC {

  @Override
  public ObjectIterator<MAP.Entry KEY_VALUE_GENERIC> iterator() { return new EntryIterator(); }

  @Override
  public ObjectIterator<MAP.Entry KEY_VALUE_GENERIC> fastIterator() { return new FastEntryIterator(); }
#endif

  @Override
  SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
  public boolean contains(final Object o) {
   if (!(o instanceof Map.Entry)) return false;
   final Map.Entry<?,?> e = (Map.Entry<?,?>)o;
#if KEYS_PRIMITIVE
   if (e.getKey() == null || ! (e.getKey() instanceof KEY_CLASS)) return false;
#endif
#if VALUES_PRIMITIVE
   if (e.getValue() == null || ! (e.getValue() instanceof VALUE_CLASS)) return false;
#endif
   final KEY_GENERIC_TYPE k = KEY_OBJ2TYPE(KEY_GENERIC_CAST e.getKey());
   final VALUE_GENERIC_TYPE v = VALUE_OBJ2TYPE(VALUE_GENERIC_CAST e.getValue());

   if (KEY_EQUALS_NULL(k)) return OPEN_HASH_MAP.this.containsNullKey && VALUE_EQUALS(value[n],
v);

   KEY_GENERIC_TYPE curr;
   final KEY_GENERIC_TYPE[] key = OPEN_HASH_MAP.this.key;
   int pos;

   // The starting point.
   if (KEY_IS_NULL(curr = key[pos = KEY2INTHASH(k) & mask])) return false;
   if (KEY_EQUALS_NOT_NULL(k, curr)) return VALUE_EQUALS(value[pos], v);
   // There's always an unused entry.
   while(true) {
    if (KEY_IS_NULL(curr = key[pos = (pos + 1) & mask])) return false;
    if (KEY_EQUALS_NOT_NULL(k, curr)) return VALUE_EQUALS(value[pos], v);
   }
  }

  @Override
  SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
  public boolean remove(final Object o) {
   if (!(o instanceof Map.Entry)) return false;
   final Map.Entry<?,?> e = (Map.Entry<?,?>)o;
#if KEYS_PRIMITIVE
```

```
  if (e.getKey() == null || ! (e.getKey() instanceof KEY_CLASS)) return false;
#endif
#if VALUES_PRIMITIVE
  if (e.getValue() == null || ! (e.getValue() instanceof VALUE_CLASS)) return false;
#endif
  final KEY_GENERIC_TYPE k = KEY_OBJ2TYPE(KEY_GENERIC_CAST e.getKey());
  final VALUE_GENERIC_TYPE v = VALUE_OBJ2TYPE(VALUE_GENERIC_CAST e.getValue());


  if (KEY_EQUALS_NULL(k)) {
   if (containsNullKey && VALUE_EQUALS(value[n], v)) {
    removeNullEntry();
    return true;
   }
   return false;
  }

  KEY_GENERIC_TYPE curr;
  final KEY_GENERIC_TYPE[] key = OPEN_HASH_MAP.this.key;
  int pos;

  // The starting point.
  if (KEY_IS_NULL(curr = key[pos = KEY2INTHASH(k) & mask])) return false;
  if (KEY_EQUALS_NOT_NULL(curr, k)) {
   if (VALUE_EQUALS(value[pos], v)) {
    removeEntry(pos);
    return true;
   }
   return false;
  }

  while(true) {
   if (KEY_IS_NULL(curr = key[pos = (pos + 1) & mask])) return false;
   if (KEY_EQUALS_NOT_NULL(curr, k))  {
    if (VALUE_EQUALS(value[pos], v)) {
     removeEntry(pos);
     return true;
    }
   }
  }
 }

 @Override
 public int size() {
  return size;
 }

 @Override
```

```
  public void clear() {
   OPEN_HASH_MAP.this.clear();
  }

#ifdef Linked
  /** Returns a type-specific list iterator on the elements in this set, starting from a given element of the set.
   * Please see the class documentation for implementation details.
   *
   * @param from an element to start from.
   * @return a type-specific list iterator starting at the given element.
   * @throws IllegalArgumentException if {@code from} does not belong to the set.
   */
  @Override
  public ObjectListIterator<MAP.Entry KEY_VALUE_GENERIC> iterator(final MAP.Entry
KEY_VALUE_GENERIC from) {
   return new EntryIterator(from.ENTRY_GET_KEY());
  }

  /** Returns a type-specific fast list iterator on the elements in this set, starting from the first element.
   * Please see the class documentation for implementation details.
   *
   * @return a type-specific list iterator starting at the first element.
   */
  @Override
  public ObjectListIterator<MAP.Entry KEY_VALUE_GENERIC> fastIterator() {
   return new FastEntryIterator();
  }

  /** Returns a type-specific fast list iterator on the elements in this set, starting from a given element of the set.
   * Please see the class documentation for implementation details.
   *
   * @param from an element to start from.
   * @return a type-specific list iterator starting at the given element.
   * @throws IllegalArgumentException if {@code from} does not belong to the set.
   */
  @Override
  public ObjectListIterator<MAP.Entry KEY_VALUE_GENERIC> fastIterator(final MAP.Entry
KEY_VALUE_GENERIC from) {
   return new FastEntryIterator(from.ENTRY_GET_KEY());
  }

  /** {@inheritDoc} */
  @Override
  public void forEach(final Consumer<? super MAP.Entry KEY_VALUE_GENERIC> consumer) {
   for(int i = size, curr, next = first; i-- != 0;) {
    curr = next;
    next = GET_NEXT(link[curr]);
    consumer.accept(new ABSTRACT_MAP.BasicEntry KEY_VALUE_GENERIC(key[curr], value[curr]));
```

```
  }
 }

 /** {@inheritDoc} */
 @Override
 public void fastForEach(final Consumer<? super MAP.Entry KEY_VALUE_GENERIC> consumer) {
  final ABSTRACT_MAP.BasicEntry KEY_VALUE_GENERIC entry = new ABSTRACT_MAP.BasicEntry
KEY_VALUE_GENERIC_DIAMOND();
  for(int i = size, curr, next = first; i-- != 0;) {
   curr = next;
   next = GET_NEXT(link[curr]);
   entry.key = key[curr];
   entry.value = value[curr];
   consumer.accept(entry);
  }
 }

#else

 /** {@inheritDoc} */
 @Override
 public void forEach(final Consumer<? super MAP.Entry KEY_VALUE_GENERIC> consumer) {
  if (containsNullKey) consumer.accept(new ABSTRACT_MAP.BasicEntry KEY_VALUE_GENERIC(key[n],
value[n]));
  for(int pos = n; pos-- != 0;)
   if (! KEY_IS_NULL(key[pos])) consumer.accept(new ABSTRACT_MAP.BasicEntry
KEY_VALUE_GENERIC(key[pos], value[pos]));
 }

 /** {@inheritDoc} */
 @Override
 public void fastForEach(final Consumer<? super MAP.Entry KEY_VALUE_GENERIC> consumer) {
  final ABSTRACT_MAP.BasicEntry KEY_VALUE_GENERIC entry = new ABSTRACT_MAP.BasicEntry
KEY_VALUE_GENERIC_DIAMOND();
  if (containsNullKey) {
   entry.key = key[n];
   entry.value = value[n];
   consumer.accept(entry);
  }
  for(int pos = n; pos-- != 0;)
   if (! KEY_IS_NULL(key[pos])) {
    entry.key = key[pos];
    entry.value = value[pos];
    consumer.accept(entry);
   }
 }

#endif
```

```
 }


#ifdef Linked
 @Override
 public FastSortedEntrySet KEY_VALUE_GENERIC ENTRYSET() {
  if (entries == null) entries = new MapEntrySet();
#else
 @Override
 public FastEntrySet KEY_VALUE_GENERIC ENTRYSET() {
  if (entries == null) entries = new MapEntrySet();
#endif
  return entries;
 }

 /** An iterator on keys.
  *
  * <p>We simply override the {@link java.util.ListIterator#next()}/{@link java.util.ListIterator#previous()}
methods
  * (and possibly their type-specific counterparts) so that they return keys
  * instead of entries.
  */

#ifdef Linked
 private final class KeyIterator extends MapIterator implements KEY_LIST_ITERATOR KEY_GENERIC {
  public KeyIterator(final KEY_GENERIC_TYPE k) { super(k); }

  @Override
  public KEY_GENERIC_TYPE PREV_KEY() { return key[previousEntry()]; }

#else
 private final class KeyIterator extends MapIterator implements KEY_ITERATOR KEY_GENERIC {
#endif
  public KeyIterator() { super(); }

  @Override
  public KEY_GENERIC_TYPE NEXT_KEY() { return key[nextEntry()]; }
 }



#ifdef Linked
 private final class KeySet extends ABSTRACT_SORTED_SET KEY_GENERIC {

  @Override
  public KEY_LIST_ITERATOR KEY_GENERIC iterator(final KEY_GENERIC_TYPE from) { return new
KeyIterator(from); }
```

```
  @Override
  public KEY_LIST_ITERATOR KEY_GENERIC iterator() { return new KeyIterator(); }
#else
 private final class KeySet extends ABSTRACT_SET KEY_GENERIC {

  @Override
  public KEY_ITERATOR KEY_GENERIC iterator() { return new KeyIterator(); }
#endif

  /** {@inheritDoc} */
  @Override
#ifdef JDK_PRIMITIVE_KEY_CONSUMER
  public void forEach(final JDK_PRIMITIVE_KEY_CONSUMER consumer) {
#else
  public void forEach(final KEY_CONSUMER KEY_SUPER_GENERIC consumer) {
#endif
    if (containsNullKey) consumer.accept(key[n]);
    for(int pos = n; pos-- != 0;) {
     final KEY_GENERIC_TYPE k = key[pos];
     if (! KEY_IS_NULL(k)) consumer.accept(k);
    }
  }

  @Override
  public int size() { return size; }

  @Override
  public boolean contains(KEY_TYPE k) { return containsKey(k); }

  @Override
  public boolean remove(KEY_TYPE k) {
   final int oldSize = size;
   OPEN_HASH_MAP.this.REMOVE_VALUE(k);
   return size != oldSize;
  }

  @Override
  public void clear() { OPEN_HASH_MAP.this.clear();}

#ifdef Linked
  @Override
  public KEY_GENERIC_TYPE FIRST() {
   if (size == 0) throw new NoSuchElementException();
   return key[first];
  }

  @Override
  public KEY_GENERIC_TYPE LAST() {
```

```
  if (size == 0) throw new NoSuchElementException();
  return key[last];
  }

  @Override
  public KEY_COMPARATOR KEY_SUPER_GENERIC comparator() { return null; }

  @Override
  public SORTED_SET KEY_GENERIC tailSet(KEY_GENERIC_TYPE from) { throw new
UnsupportedOperationException(); }

  @Override
  public SORTED_SET KEY_GENERIC headSet(KEY_GENERIC_TYPE to) { throw new
UnsupportedOperationException(); }

  @Override
  public SORTED_SET KEY_GENERIC subSet(KEY_GENERIC_TYPE from, KEY_GENERIC_TYPE to) { throw
new UnsupportedOperationException(); }
#endif
  }


#ifdef Linked
  @Override
  public SORTED_SET KEY_GENERIC keySet() {
#else
  @Override
  public SET KEY_GENERIC keySet() {
#endif
  if (keys == null) keys = new KeySet();
  return keys;
  }


  /** An iterator on values.
  *
  * <p>We simply override the {@link java.util.ListIterator#next()}/{@link java.util.ListIterator#previous()}
methods
  * (and possibly their type-specific counterparts) so that they return values
  * instead of entries.
  */

#ifdef Linked
  private final class ValueIterator extends MapIterator implements VALUE_LIST_ITERATOR VALUE_GENERIC {
  @Override
  public VALUE_GENERIC_TYPE PREV_VALUE() { return value[previousEntry()]; }

#else
```

```
 private final class ValueIterator extends MapIterator implements VALUE_ITERATOR VALUE_GENERIC {
#endif
  public ValueIterator() { super(); }

  @Override
  public VALUE_GENERIC_TYPE NEXT_VALUE() { return value[nextEntry()]; }
 }

 @Override
 public VALUE_COLLECTION VALUE_GENERIC values() {
  if (values == null) values = new VALUE_ABSTRACT_COLLECTION VALUE_GENERIC() {
    @Override
    public VALUE_ITERATOR VALUE_GENERIC iterator() { return new ValueIterator(); }
    @Override
    public int size() { return size; }
    @Override
    public boolean contains(VALUE_TYPE v) { return containsValue(v); }
    @Override
    public void clear() { OPEN_HASH_MAP.this.clear(); }

    /** {@inheritDoc} */
    @Override
#ifdef JDK_PRIMITIVE_VALUE_CONSUMER
    public void forEach(final JDK_PRIMITIVE_VALUE_CONSUMER consumer) {
#else
    public void forEach(final VALUE_CONSUMER VALUE_SUPER_GENERIC consumer) {
#endif
      if (containsNullKey) consumer.accept(value[n]);
      for(int pos = n; pos-- != 0;)
       if (! KEY_IS_NULL(key[pos])) consumer.accept(value[pos]);
     }

   };

  return values;
 }


 /** Rehashes the map, making the table as small as possible.
  *
  * <p>This method rehashes the table to the smallest size satisfying the
  * load factor. It can be used when the set will not be changed anymore, so
  * to optimize access speed and size.
  *
  * <p>If the table size is already the minimum possible, this method
  * does nothing.
  *
  * @return true if there was enough memory to trim the map.
```

```
 * @see #trim(int)
 */

public boolean trim() {
 final int l = arraySize(size, f);
 if (l >= n || size > maxFill(l, f)) return true;
 try {
  rehash(l);
 }
 catch(OutOfMemoryError cantDoIt) { return false; }
 return true;
}


/** Rehashes this map if the table is too large.
 *
 * <p>Let <var>N</var> be the smallest table size that can hold
 * <code>max(n,{@link #size()})</code> entries, still satisfying the load factor. If the current
 * table size is smaller than or equal to <var>N</var>, this method does
 * nothing. Otherwise, it rehashes this map in a table of size
 * <var>N</var>.
 *
 * <p>This method is useful when reusing maps.  {@linkplain #clear() Clearing a
 * map} leaves the table size untouched. If you are reusing a map
 * many times, you can call this method with a typical
 * size to avoid keeping around a very large table just
 * because of a few large transient maps.
 *
 * @param n the threshold for the trimming.
 * @return true if there was enough memory to trim the map.
 * @see #trim()
 */

public boolean trim(final int n) {
 final int l = HashCommon.nextPowerOfTwo((int)Math.ceil(n / f));
 if (l >= n || size > maxFill(l, f)) return true;
 try {
  rehash(l);
 }
 catch(OutOfMemoryError cantDoIt) { return false; }
 return true;
}

/** Rehashes the map.
 *
 * <p>This method implements the basic rehashing strategy, and may be
 * overridden by subclasses implementing different rehashing strategies (e.g.,
 * disk-based rehashing). However, you should not override this method
```

```
 * unless you understand the internal workings of this class.
 *
 * @param newN the new size
 */

 SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
 protected void rehash(final int newN) {
  final KEY_GENERIC_TYPE key[] = this.key;
  final VALUE_GENERIC_TYPE value[] = this.value;

  final int mask = newN - 1; // Note that this is used by the hashing macro
  final KEY_GENERIC_TYPE newKey[] = KEY_GENERIC_ARRAY_CAST new KEY_TYPE[newN + 1];
  final VALUE_GENERIC_TYPE newValue[] = VALUE_GENERIC_ARRAY_CAST new VALUE_TYPE[newN +
1];

#ifdef Linked
  int i = first, prev = -1, newPrev = -1, t, pos;
  final long link[] = this.link;
  final long newLink[] = new long[newN + 1];
  first = -1;

  for(int j = size; j-- != 0;) {
   if (KEY_EQUALS_NULL(key[i])) pos = newN;
   else {
    pos = KEY2INTHASH(key[i]) & mask;
    while (! KEY_IS_NULL(newKey[pos])) pos = (pos + 1) & mask;
   }

   newKey[pos] = key[i];
   newValue[pos] = value[i];

   if (prev != -1) {
    SET_NEXT(newLink[newPrev], pos);
    SET_PREV(newLink[pos], newPrev);
    newPrev = pos;
   }
   else {
    newPrev = first = pos;
    // Special case of SET(newLink[pos], -1, -1);
    newLink[pos] = -1L;
   }

   t = i;
   i = GET_NEXT(link[i]);
   prev = t;
  }

  this.link = newLink;
```

```
  this.last = newPrev;
  if (newPrev != -1)
   // Special case of SET_NEXT(newLink[newPrev], -1);
   newLink[newPrev] |= -1 & 0xFFFFFFFFL;
 #else
  int i = n, pos;

  for(int j = realSize(); j-- != 0;) {
   while(KEY_IS_NULL(key[--i]));

   if (! KEY_IS_NULL(newKey[pos = KEY2INTHASH(key[i]) & mask]))
    while (! KEY_IS_NULL(newKey[pos = (pos + 1) & mask]));

   newKey[pos] = key[i];
   newValue[pos] = value[i];
  }

  newValue[newN] = value[n];

 #endif

  n = newN;
  this.mask = mask;
  maxFill = maxFill(n, f);
  this.key = newKey;
  this.value = newValue;
 }


 /** Returns a deep copy of this map.
  *
  * <p>This method performs a deep copy of this hash map; the data stored in the
  * map, however, is not cloned. Note that this makes a difference only for object keys.
  *
  *  @return a deep copy of this map.
  */
 @Override
 SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
 public OPEN_HASH_MAP KEY_VALUE_GENERIC clone() {
  OPEN_HASH_MAP KEY_VALUE_GENERIC c;
  try {
   c = (OPEN_HASH_MAP KEY_VALUE_GENERIC)super.clone();
  }
  catch(CloneNotSupportedException cantHappen) {
   throw new InternalError();
  }

  c.keys = null;
```

```
  c.values = null;
  c.entries = null;
  c.containsNullKey = containsNullKey;

  c.key = key.clone();
  c.value = value.clone();
#ifdef Linked
  c.link = link.clone();
#endif
#ifdef Custom
  c.strategy = strategy;
#endif
  return c;
 }


 /** Returns a hash code for this map.
  *
  * This method overrides the generic method provided by the superclass.
  * Since {@code equals()} is not overriden, it is important
  * that the value returned by this method is the same value as
  * the one returned by the overriden method.
  *
  * @return a hash code for this map.
  */

 @Override
 public int hashCode() {
  int h = 0;
  for(int j = realSize(), i = 0, t = 0; j-- != 0;) {
   while(KEY_IS_NULL(key[i])) i++;
#if KEYS_REFERENCE
   if (this != key[i])
#endif
    t = KEY2JAVAHASH_NOT_NULL(key[i]);
#if VALUES_REFERENCE
   if (this != value[i])
#endif
    t ^=  VALUE2JAVAHASH(value[i]);
   h += t;
   i++;
  }
  // Zero / null keys have hash zero.
  if (containsNullKey) h += VALUE2JAVAHASH(value[n]);
  return h;
 }
```

```
 private void writeObject(java.io.ObjectOutputStream s) throws java.io.IOException {
  final KEY_GENERIC_TYPE key[] = this.key;
  final VALUE_GENERIC_TYPE value[] = this.value;
  final MapIterator i = new MapIterator();

  s.defaultWriteObject();

  for(int j = size, e; j-- != 0;) {
   e = i.nextEntry();
   s.WRITE_KEY(key[e]);
   s.WRITE_VALUE(value[e]);
  }
 }



 SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
 private void readObject(java.io.ObjectInputStream s) throws java.io.IOException, ClassNotFoundException {
  s.defaultReadObject();

  n = arraySize(size, f);
  maxFill = maxFill(n, f);
  mask = n - 1;

  final KEY_GENERIC_TYPE key[] = this.key = KEY_GENERIC_ARRAY_CAST new KEY_TYPE[n + 1];
  final VALUE_GENERIC_TYPE value[] = this.value = VALUE_GENERIC_ARRAY_CAST new
VALUE_TYPE[n + 1];
#ifdef Linked
  final long link[] = this.link = new long[n + 1];
  int prev = -1;
  first = last = -1;
#endif

  KEY_GENERIC_TYPE k;
  VALUE_GENERIC_TYPE v;

  for(int i = size, pos; i-- != 0;) {

   k = KEY_GENERIC_CAST s.READ_KEY();
   v = VALUE_GENERIC_CAST s.READ_VALUE();

   if (KEY_EQUALS_NULL(k)) {
    pos = n;
    containsNullKey = true;
   }
   else {
    pos = KEY2INTHASH(k) & mask;
```

```
    while (! KEY_IS_NULL(key[pos])) pos = (pos + 1) & mask;
   }

   key[pos] = k;
   value[pos] = v;

#ifdef Linked
   if (first != -1) {
    SET_NEXT(link[prev], pos);
    SET_PREV(link[pos], prev);
    prev = pos;
   }
   else {
    prev = first = pos;
    // Special case of SET_PREV(newLink[pos], -1);
    link[pos] |= (-1L & 0xFFFFFFFFL) << 32;
   }
#endif
  }

#ifdef Linked
  last = prev;
  if (prev != -1)
   // Special case of SET_NEXT(link[prev], -1);
   link[prev] |= -1 & 0xFFFFFFFFL;
#endif

  if (ASSERTS) checkTable();
 }



#ifdef ASSERTS_CODE
 private void checkTable() {
  assert (n & -n) == n : "Table length is not a power of two: " + n;
  assert n == key.length - 1;
  int n = key.length - 1;
  while(n-- != 0)
   if (! KEY_IS_NULL(key[n]) && ! containsKey(key[n]))
    throw new AssertionError("Hash table has key " + key[n] + " marked as occupied, but the key does not belong to the
table");

#if KEYS_PRIMITIVE
  java.util.HashSet<KEY_GENERIC_CLASS> s = new java.util.HashSet<KEY_GENERIC_CLASS> ();
#else
  java.util.HashSet<Object> s = new java.util.HashSet<Object>();
#endif
```

```
  for(int i = key.length; i-- != 0;)
   if (! KEY_IS_NULL(key[i]) && ! s.add(key[i])) throw new AssertionError("Key " + key[i] + " appears twice at
position " + i);

#ifdef Linked
  KEY_BIDI_ITERATOR KEY_GENERIC i = keySet().iterator();
  KEY_GENERIC_TYPE k;
  n = size();
  while(n-- != 0)
   if (! containsKey(k = i.NEXT_KEY()))
    throw new AssertionError("Linked hash table forward enumerates key " + k + ", but the key does not belong to the
table");

  if (i.hasNext()) throw new AssertionError("Forward iterator not exhausted");

  n = size();
  if (n > 0) {
   i = keySet().iterator(LAST_KEY());
   while(n-- != 0)
    if (! containsKey(k = i.PREV_KEY()))
     throw new AssertionError("Linked hash table backward enumerates key " + k + ", but the key does not belong to the
table");

   if (i.hasPrevious()) throw new AssertionError("Previous iterator not exhausted");
  }
#endif
 }
#else
 private void checkTable() {}
#endif



#ifdef TEST

 private static long seed = System.currentTimeMillis();
 private static java.util.Random r = new java.util.Random(seed);

 private static KEY_TYPE genKey() {
#if KEY_CLASS_Byte || KEY_CLASS_Short || KEY_CLASS_Character
  return (KEY_TYPE)(r.nextInt());
#elif KEYS_PRIMITIVE
  return r.NEXT_KEY();
#elif !KEY_CLASS_Reference
#ifdef Custom
  int i = r.nextInt(3);
  byte a[] = new byte[i];
  while(i-- != 0) a[i] = (byte)r.nextInt();
  return a;
```

```
#else
  return Integer.toBinaryString(r.nextInt());
#endif
#else
  return new java.io.Serializable() {};
#endif
 }

 private static VALUE_TYPE genValue() {
#if VALUE_CLASS_Byte || VALUE_CLASS_Short || VALUE_CLASS_Character
  return (VALUE_TYPE)(r.nextInt());
#elif VALUES_PRIMITIVE
  return r.NEXT_VALUE();
#elif !VALUE_CLASS_Reference
  return Integer.toBinaryString(r.nextInt());
#else
  return new java.io.Serializable() {};
#endif
 }

 private static final class ArrayComparator implements java.util.Comparator {
  public int compare(Object a, Object b) {
   byte[] aa = (byte[])a;
   byte[] bb = (byte[])b;
   int length = Math.min(aa.length, bb.length);
   for(int i = 0; i < length; i++) {
    if (aa[i] < bb[i]) return -1;
    if (aa[i] > bb[i]) return 1;
   }
   return aa.length == bb.length ? 0 : (aa.length < bb.length ? -1 : 1);
  }
 }

 private static final class MockMap extends java.util.TreeMap {
  private java.util.List list = new java.util.ArrayList();

  public MockMap(java.util.Comparator c) { super(c); }

  public Object put(Object k, Object v) {
   if (! containsKey(k)) list.add(k);
   return super.put(k, v);
  }

  public void putAll(Map m) {
   java.util.Iterator i = m.entrySet().iterator();
   while(i.hasNext()) {
    Map.Entry e = (Map.Entry)i.next();
    put(e.getKey(), e.getValue());
```

```
 }
}

public Object remove(Object k) {
 if (containsKey(k)) {
  int i = list.size();
  while(i-- != 0) if (comparator().compare(list.get(i), k) == 0) {
   list.remove(i);
   break;
  }
 }
 return super.remove(k);
}

private void justRemove(Object k) { super.remove(k); }
private java.util.Set justEntrySet() { return super.entrySet(); }
private java.util.Set justKeySet() { return super.keySet(); }

public java.util.Set keySet() {
 return new java.util.AbstractSet() {
   final java.util.Set keySet = justKeySet();

   public boolean contains(Object k) { return keySet.contains(k); }
   public int size() { return keySet.size(); }
   public java.util.Iterator iterator() {
    return new java.util.Iterator() {
      final java.util.Iterator iterator = list.iterator();
      Object curr;
      public Object next() { return curr = iterator.next(); }
      public boolean hasNext() { return iterator.hasNext(); }
      public void remove() {
       justRemove(curr);
       iterator.remove();
      }
     };

   }
  };

}

public java.util.Set entrySet() {
 return new java.util.AbstractSet() {
   final java.util.Set entrySet = justEntrySet();

   public boolean contains(Object k) { return entrySet.contains(k); }
   public int size() { return entrySet.size(); }
   public java.util.Iterator iterator() {
```

```
      return new java.util.Iterator() {
        final java.util.Iterator iterator = list.iterator();
        Object curr;
        public Object next() {
         curr = iterator.next();
#if VALUE_CLASS_Reference
#if KEY_CLASS_Reference
         return new ABSTRACT_MAP.BasicEntry((Object)curr, (Object)get(curr)) {
#else
         return new ABSTRACT_MAP.BasicEntry((KEY_CLASS)curr, (Object)get(curr)) {
#endif
#else
#if KEY_CLASS_Reference
         return new ABSTRACT_MAP.BasicEntry((Object)curr, (VALUE_CLASS)get(curr)) {
#else
         return new ABSTRACT_MAP.BasicEntry((KEY_CLASS)curr, (VALUE_CLASS)get(curr)) {
#endif
#endif
           public VALUE_TYPE setValue(VALUE_TYPE v) {
            return VALUE_OBJ2TYPE(put(getKey(), VALUE2OBJ(v)));
           }
          };
        }
        public boolean hasNext() { return iterator.hasNext(); }
        public void remove() {
         justRemove(((Map.Entry)curr).getKey());
         iterator.remove();
        }
       };
     }
    };
  }
 }

 private static java.text.NumberFormat format = new java.text.DecimalFormat("#,###.00");
 private static java.text.FieldPosition fp = new java.text.FieldPosition(0);

 private static String format(double d) {
  StringBuffer s = new StringBuffer();
  return format.format(d, s, fp).toString();
 }


 private static void speedTest(int n, float f, boolean comp) {
#ifndef Custom
  int i, j;
  OPEN_HASH_MAP m;
#ifdef Linked
```

```
  java.util.LinkedHashMap t;
#else
  java.util.HashMap t;
#endif
  KEY_TYPE k[] = new KEY_TYPE[n];
  KEY_TYPE nk[] = new KEY_TYPE[n];
  VALUE_TYPE v[] = new VALUE_TYPE[n];
  long ns;

  for(i = 0; i < n; i++) {
   k[i] = genKey();
   nk[i] = genKey();
   v[i] = genValue();
  }

  double totPut = 0, totYes = 0, totNo = 0, totIter = 0, totRemYes = 0, totRemNo = 0, d;

  if (comp) { for(j = 0; j < 20; j++) {

#ifdef Linked
   t = new java.util.LinkedHashMap(16);
#else
   t = new java.util.HashMap(16);
#endif

   /* We put pairs to t. */
   ns = System.nanoTime();
   for(i = 0; i < n;  i++) t.put(KEY2OBJ(k[i]), VALUE2OBJ(v[i]));
   d = (System.nanoTime() - ns) / (double)n;
   if (j > 2) totPut += d;
   System.out.print("Put: " + format(d) + "ns ");

   /* We check for pairs in t. */
   ns = System.nanoTime();
   for(i = 0; i < n;  i++) t.containsKey(KEY2OBJ(k[i]));
   d = (System.nanoTime() - ns) / (double)n;
   if (j > 2) totYes += d;
   System.out.print("Yes: " + format(d) + "ns ");

   /* We check for pairs not in t. */
   ns = System.nanoTime();
   for(i = 0; i < n;  i++) t.containsKey(KEY2OBJ(nk[i]));
   d = (System.nanoTime() - ns) / (double)n;
   if (j > 2) totNo += d;
   System.out.print("No: " + format(d) + "ns ");

   /* We iterate on t. */
   ns = System.nanoTime();
```

```
 for(java.util.Iterator it = t.entrySet().iterator(); it.hasNext(); it.next());
 d = (System.nanoTime() - ns) / (double)n;
 if (j > 2) totIter += d;
 System.out.print("Iter: " + format(d) + "ns ");

 /* We delete pairs not in t. */
 ns = System.nanoTime();
 for(i = 0; i < n;  i++) t.remove(KEY2OBJ(nk[i]));
 d = (System.nanoTime() - ns) / (double)n;
 if (j > 2) totRemNo += d;
 System.out.print("RemNo: " + format(d) + "ns ");

 /* We delete pairs in t. */
 ns = System.nanoTime();
 for(i = 0; i < n;  i++) t.remove(KEY2OBJ(k[i]));
 d = (System.nanoTime() - ns) / (double)n;
 if (j > 2) totRemYes += d;
 System.out.print("RemYes: " + format(d) + "ns ");

 System.out.println();
}

System.out.println();
System.out.println("java.util Put: " + format(totPut/(j-3)) + "ns Yes: " + format(totYes/(j-3)) + "ns No: " +
format(totNo/(j-3)) + "ns Iter: " + format(totIter/(j-3)) + "ns RemNo: " + format(totRemNo/(j-3)) + "ns RemYes: " +
format(totRemYes/(j-3)) + "K/s");

System.out.println();

totPut = totYes = totNo = totIter = totRemYes = totRemNo = 0;

}

for(j = 0; j < 20; j++) {

 m = new OPEN_HASH_MAP(16, f);

 /* We put pairs to m. */
 ns = System.nanoTime();
 for(i = 0; i < n;  i++) m.put(k[i], v[i]);
 d = (System.nanoTime() - ns) / (double)n;
 if (j > 2) totPut += d;
 System.out.print("Put: " + format(d) + "ns ");

 /* We check for pairs in m. */
 ns = System.nanoTime();
 for(i = 0; i < n;  i++) m.containsKey(k[i]);
 d = (System.nanoTime() - ns) / (double)n;
```

```
  if (j > 2) totYes += d;
  System.out.print("Yes: " + format(d) + "ns ");

  /* We check for pairs not in m. */
  ns = System.nanoTime();
  for(i = 0; i < n;  i++) m.containsKey(nk[i]);
  d = (System.nanoTime() - ns) / (double)n;
  if (j > 2) totNo += d;
  System.out.print("No: " + format(d) + "ns ");

  /* We iterate on m. */
  ns = System.nanoTime();
  for(java.util.Iterator it = m.entrySet().iterator(); it.hasNext(); it.next());
  d = (System.nanoTime() - ns) / (double)n;
  if (j > 2) totIter += d;
  System.out.print("Iter: " + format(d) + "ns ");

  /* We delete pairs not in m. */
  ns = System.nanoTime();
  for(i = 0; i < n;  i++) m.remove(nk[i]);
  d = (System.nanoTime() - ns) / (double)n;
  if (j > 2) totRemNo += d;
  System.out.print("RemNo: " + format(d) + "ns ");

  /* We delete pairs in m. */
  ns = System.nanoTime();
  for(i = 0; i < n;  i++) m.remove(k[i]);
  d = (System.nanoTime() - ns) / (double)n;
  if (j > 2) totRemYes += d;
  System.out.print("RemYes: " + format(d) + "ns ");

  System.out.println();
  }


  System.out.println();
  System.out.println("fastutil  Put: " + format(totPut/(j-3)) + "ns Yes: " + format(totYes/(j-3)) + "ns No: " +
format(totNo/(j-3)) + "ns Iter: " + format(totIter/(j-3)) + "ns RemNo: " + format(totRemNo/(j-3)) + "ns RemYes: " +
format(totRemYes/(j-3)) + "ns");

  System.out.println();
#endif
  }

 private static boolean valEquals(Object o1, Object o2) {
  return o1 == null ? o2 == null : o1.equals(o2);
  }
```

```
 private static void fatal(String msg) {
  System.out.println(msg);
  System.exit(1);
 }

 private static void ensure(boolean cond, String msg) {
  if (cond) return;
  fatal(msg);
 }

 protected static void runTest(int n, float f) {
#if !defined(Custom) || KEYS_REFERENCE

#ifdef Custom
  OPEN_HASH_MAP m = new OPEN_HASH_MAP(Hash.DEFAULT_INITIAL_SIZE, f,
it.unimi.dsi.fastutil.bytes.ByteArrays.HASH_STRATEGY);
#else
  OPEN_HASH_MAP m = new OPEN_HASH_MAP(Hash.DEFAULT_INITIAL_SIZE, f);
#endif

#ifdef Linked
#ifdef Custom
  Map t = new MockMap(new ArrayComparator());
#else
  Map t = new java.util.LinkedHashMap();
#endif
#else
#ifdef Custom
  Map t = new java.util.TreeMap(new ArrayComparator());
#else
  Map t = new java.util.HashMap();
#endif
#endif

  /* First of all, we fill t with random data. */

  for(int i=0; i<n;  i++) t.put(KEY2OBJ(genKey()), VALUE2OBJ(genValue()));

  /* Now we add to m the same data */

  m.putAll(t);

  if (!m.equals(t)) System.out.println("Error (" + seed + "): !m.equals(t) after insertion");
  if (!t.equals(m)) System.out.println("Error (" + seed + "): !t.equals(m) after insertion");

  /* Now we check that m actually holds that data. */

  for(java.util.Iterator i=t.entrySet().iterator(); i.hasNext();) {
```

```
 java.util.Map.Entry e = (java.util.Map.Entry)i.next();
 if (!valEquals(e.getValue(), m.get(e.getKey())))
  System.out.println("Error (" + seed + "): m and t differ on an entry ("+e+") after insertion (iterating on t)");
}

/* Now we check that m actually holds that data, but iterating on m. */

for(java.util.Iterator i=m.entrySet().iterator(); i.hasNext();) {
 java.util.Map.Entry e = (java.util.Map.Entry)i.next();
 if (!valEquals(e.getValue(), t.get(e.getKey())))
  System.out.println("Error (" + seed + "): m and t differ on an entry ("+e+") after insertion (iterating on m)");
}

/* Now we check that m actually holds the same keys. */

for(java.util.Iterator i=t.keySet().iterator(); i.hasNext();) {
 Object o = i.next();
 if (!m.containsKey(o)) {
  System.out.println("Error (" + seed + "): m and t differ on a key ("+o+") after insertion (iterating on t)");
  System.exit(1);
 }
 if (!m.keySet().contains(o)) {
  System.out.println("Error (" + seed + "): m and t differ on a key ("+o+", in keySet()) after insertion (iterating on t)");
  System.exit(1);
 }
}

/* Now we check that m actually holds the same keys, but iterating on m. */

for(java.util.Iterator i=m.keySet().iterator(); i.hasNext();) {
 Object o = i.next();
 if (!t.containsKey(o)) {
  System.out.println("Error (" + seed + "): m and t differ on a key after insertion (iterating on m)");
  System.exit(1);
 }
 if (!t.keySet().contains(o)) {
  System.out.println("Error (" + seed + "): m and t differ on a key (in keySet()) after insertion (iterating on m)");
  System.exit(1);
 }
}


/* Now we check that m actually hold the same values. */

for(java.util.Iterator i=t.values().iterator(); i.hasNext();) {
 Object o = i.next();
 if (!m.containsValue(o)) {
  System.out.println("Error (" + seed + "): m and t differ on a value after insertion (iterating on t)");
```

```
   System.exit(1);
  }
  if (!m.values().contains(o)) {
   System.out.println("Error (" + seed + "): m and t differ on a value (in values()) after insertion (iterating on t)");
   System.exit(1);
  }
 }

 /* Now we check that m actually hold the same values, but iterating on m. */

 for(java.util.Iterator i=m.values().iterator(); i.hasNext();) {
  Object o = i.next();
  if (!t.containsValue(o)) {
   System.out.println("Error (" + seed + "): m and t differ on a value after insertion (iterating on m)");
   System.exit(1);
  }
  if (!t.values().contains(o)) {
   System.out.println("Error (" + seed + "): m and t differ on a value (in values()) after insertion (iterating on m)");
   System.exit(1);
  }
 }

 /* Now we check that inquiries about random data give the same answer in m and t. For
    m we use the polymorphic method. */

 for(int i=0; i<n;  i++) {
  KEY_TYPE T = genKey();
  if (m.containsKey(KEY2OBJ(T)) != t.containsKey(KEY2OBJ(T))) {
   System.out.println("Error (" + seed + "): divergence in keys between t and m (polymorphic method)");
   System.exit(1);
  }

#if (KEYS_REFERENCE) && ! (VALUES_REFERENCE)
  if ((m.GET_VALUE(T) != VALUE_NULL) != ((t.get(KEY2OBJ(T)) == null ? VALUE_NULL :
VALUE_OBJ2TYPE(t.get(KEY2OBJ(T)))) != VALUE_NULL) ||
   t.get(KEY2OBJ(T)) != null &&
   ! VALUE2OBJ(m.GET_VALUE(T)).equals(t.get(KEY2OBJ(T))))
#else
   if ((m.get(T) != VALUE_NULL) != ((t.get(KEY2OBJ(T)) == null ? VALUE_NULL :
VALUE_OBJ2TYPE(t.get(KEY2OBJ(T)))) != VALUE_NULL) ||
    t.get(KEY2OBJ(T)) != null &&
    ! m.get(KEY2OBJ(T)).equals(t.get(KEY2OBJ(T))))
#endif
    {
     System.out.println("Error (" + seed + "): divergence between t and m (polymorphic method)");
     System.exit(1);
    }
  }
```

```
/* Again, we check that inquiries about random data give the same answer in m and t, but
   for m we use the standard method. */

for(int i=0; i<n;  i++) {
 KEY_TYPE T = genKey();
 if (!valEquals(m.get(KEY2OBJ(T)), t.get(KEY2OBJ(T)))) {
  System.out.println("Error (" + seed + "): divergence between t and m (standard method)");
  System.exit(1);
 }
}

/* Now we put and remove random data in m and t, checking that the result is the same. */

for(int i=0; i<20*n;  i++) {
 KEY_TYPE T = genKey();
 VALUE_TYPE U = genValue();
 if (!valEquals(m.put(KEY2OBJ(T), VALUE2OBJ(U)), t.put(KEY2OBJ(T), VALUE2OBJ(U)))) {
  System.out.println("Error (" + seed + "): divergence in put() between t and m");
  System.exit(1);
 }
 T = genKey();
 if (!valEquals(m.remove(KEY2OBJ(T)), t.remove(KEY2OBJ(T)))) {
  System.out.println("Error (" + seed + "): divergence in remove() between t and m");
  System.exit(1);
 }
}

if (!m.equals(t)) System.out.println("Error (" + seed + "): !m.equals(t) after removal");
if (!t.equals(m)) System.out.println("Error (" + seed + "): !t.equals(m) after removal");


/* Now we check that m actually holds the same data. */

for(java.util.Iterator i=t.entrySet().iterator(); i.hasNext();) {
 java.util.Map.Entry e = (java.util.Map.Entry)i.next();
 if (!valEquals(e.getValue(), m.get(e.getKey()))) {
  System.out.println("Error (" + seed + "): m and t differ on an entry ("+e+") after removal (iterating on t)");
  System.exit(1);
 }
}

/* Now we check that m actually holds that data, but iterating on m. */

for(java.util.Iterator i=m.entrySet().iterator(); i.hasNext();) {
 java.util.Map.Entry e = (java.util.Map.Entry)i.next();
 if (!valEquals(e.getValue(), t.get(e.getKey()))) {
  System.out.println("Error (" + seed + "): m and t differ on an entry ("+e+") after removal (iterating on m)");
```

```
 System.exit(1);
 }
}

/* Now we check that m actually holds the same keys. */

for(java.util.Iterator i=t.keySet().iterator(); i.hasNext();) {
 Object o = i.next();
 if (!m.containsKey(o)) {
  System.out.println("Error (" + seed + "): m and t differ on a key ("+o+") after removal (iterating on t)");
  System.exit(1);
 }
 if (!m.keySet().contains(o)) {
  System.out.println("Error (" + seed + "): m and t differ on a key ("+o+", in keySet()) after removal (iterating on t)");
  System.exit(1);
 }
}

/* Now we check that m actually holds the same keys, but iterating on m. */

for(java.util.Iterator i=m.keySet().iterator(); i.hasNext();) {
 Object o = i.next();
 if (!t.containsKey(o)) {
  System.out.println("Error (" + seed + "): m and t differ on a key after removal (iterating on m)");
  System.exit(1);
 }
 if (!t.keySet().contains(o)) {
  System.out.println("Error (" + seed + "): m and t differ on a key (in keySet()) after removal (iterating on m)");
  System.exit(1);
 }
}


/* Now we check that m actually hold the same values. */

for(java.util.Iterator i=t.values().iterator(); i.hasNext();) {
 Object o = i.next();
 if (!m.containsValue(o)) {
  System.out.println("Error (" + seed + "): m and t differ on a value after removal (iterating on t)");
  System.exit(1);
 }
 if (!m.values().contains(o)) {
  System.out.println("Error (" + seed + "): m and t differ on a value (in values()) after removal (iterating on t)");
  System.exit(1);
 }
}

/* Now we check that m actually hold the same values, but iterating on m. */
```

```
 for(java.util.Iterator i=m.values().iterator(); i.hasNext();) {
  Object o = i.next();
  if (!t.containsValue(o)) {
   System.out.println("Error (" + seed + "): m and t differ on a value after removal (iterating on m)");
   System.exit(1);
  }
  if (!t.values().contains(o)) {
   System.out.println("Error (" + seed + "): m and t differ on a value (in values()) after removal (iterating on m)");
   System.exit(1);
  }
 }


 int h = m.hashCode();


 /* Now we save and read m. */

 try {
  java.io.File ff = new java.io.File("it.unimi.dsi.fastutil.test");
  java.io.OutputStream os = new java.io.FileOutputStream(ff);
  java.io.ObjectOutputStream oos = new java.io.ObjectOutputStream(os);

  oos.writeObject(m);
  oos.close();

  java.io.InputStream is = new java.io.FileInputStream(ff);
  java.io.ObjectInputStream ois = new java.io.ObjectInputStream(is);

  m = (OPEN_HASH_MAP)ois.readObject();
  ois.close();
  ff.delete();
 }
 catch(Exception e) {
  e.printStackTrace();
  System.exit(1);
 }


#if !KEY_CLASS_Reference && !VALUE_CLASS_Reference
 if (m.hashCode() != h) System.out.println("Error (" + seed + "): hashCode() changed after save/read");

 /* Now we check that m actually holds that data. */

 for(java.util.Iterator i=t.keySet().iterator(); i.hasNext();) {
  Object o = i.next();
  if (!valEquals(m.get(o),t.get(o))) {
```

```
    System.out.println("Error (" + seed + "): m and t differ on an entry after save/read");
    System.exit(1);
   }
  }
#else
  m.clear();
  m.putAll(t);
#endif

  /* Now we put and remove random data in m and t, checking that the result is the same. */

  for(int i=0; i<20*n;  i++) {
   KEY_TYPE T = genKey();
   VALUE_TYPE U = genValue();
   if (!valEquals(m.put(KEY2OBJ(T), VALUE2OBJ(U)), t.put(KEY2OBJ(T), VALUE2OBJ(U)))) {
    System.out.println("Error (" + seed + "): divergence in put() between t and m after save/read");
    System.exit(1);
   }
   T = genKey();
   if (!valEquals(m.remove(KEY2OBJ(T)), t.remove(KEY2OBJ(T)))) {
    System.out.println("Error (" + seed + "): divergence in remove() between t and m after save/read");
    System.exit(1);
   }
  }

  if (!m.equals(t)) System.out.println("Error (" + seed + "): !m.equals(t) after post-save/read removal");
  if (!t.equals(m)) System.out.println("Error (" + seed + "): !t.equals(m) after post-save/read removal");



#ifdef Linked


  /* Now we play with iterators. */

  {
   java.util.ListIterator i, j;
   Object J;
   Map.Entry E, F;
   i = (java.util.ListIterator)m.entrySet().iterator();
   j = new java.util.LinkedList(t.entrySet()).listIterator();

   for(int k = 0; k < 2*n; k++) {
    ensure(i.hasNext() == j.hasNext(), "Error (" + seed + "): divergence in hasNext()");
    ensure(i.hasPrevious() == j.hasPrevious(), "Error (" + seed + "): divergence in hasPrevious()");

    if (r.nextFloat() < .8 && i.hasNext()) {
#ifdef Custom
```

```
    ensure(m.strategy().equals((E=(java.util.Map.Entry)i.next()).getKey(), J = (F=(Map.Entry)j.next()).getKey()), "Error
(" + seed + "): divergence in next()");
#else
    ensure((E=(java.util.Map.Entry)i.next()).getKey().equals(J = (F=(Map.Entry)j.next()).getKey()), "Error (" + seed +
"): divergence in next()");
#endif

    if (r.nextFloat() < 0.3) {
     i.remove();
     j.remove();
     t.remove(J);
    }
    else if (r.nextFloat() < 0.3) {
     Object U = VALUE2OBJ(genValue());
     E.setValue(U);
     t.put(F.getKey(), U);
    }
   }
   else if (r.nextFloat() < .2 && i.hasPrevious()) {
#ifdef Custom
    ensure(m.strategy().equals((E=(java.util.Map.Entry)i.previous()).getKey(), J =
(F=(Map.Entry)j.previous()).getKey()), "Error (" + seed + "): divergence in previous()");
#else
    ensure((E=(java.util.Map.Entry)i.previous()).getKey().equals(J = (F=(Map.Entry)j.previous()).getKey()), "Error (" +
seed + "): divergence in previous()");
#endif

    if (r.nextFloat() < 0.3) {
     i.remove();
     j.remove();
     t.remove(J);
    }
    else if (r.nextFloat() < 0.3) {
     Object U = VALUE2OBJ(genValue());
     E.setValue(U);
     t.put(F.getKey(), U);
    }
   }

   ensure(i.nextIndex() == j.nextIndex(), "Error (" + seed + "): divergence in nextIndex()");
   ensure(i.previousIndex() == j.previousIndex(), "Error (" + seed + "): divergence in previousIndex()");

  }

 }

 if (t.size() > 0) {
  java.util.ListIterator i, j;
```

```
  Object J;
  j = new java.util.LinkedList(t.keySet()).listIterator();
  int e = r.nextInt(t.size());
  Object from;
  do from = j.next(); while(e-- != 0);

  i = (java.util.ListIterator)((SORTED_SET)m.keySet()).iterator(KEY_OBJ2TYPE(from));

  for(int k = 0; k < 2*n; k++) {
   ensure(i.hasNext() == j.hasNext(), "Error (" + seed + "): divergence in hasNext() (iterator with starting point " + from + ")");
   ensure(i.hasPrevious() == j.hasPrevious(), "Error (" + seed + "): divergence in hasPrevious() (iterator with starting point " + from + ")");

   if (r.nextFloat() < .8 && i.hasNext()) {
#ifdef Custom
    ensure(m.strategy().equals(i.next(), J = j.next()), "Error (" + seed + "): divergence in next() (iterator with starting point " + from + ")");
#else
    ensure(i.next().equals(J = j.next()), "Error (" + seed + "): divergence in next() (iterator with starting point " + from + ")");
#endif

    if (r.nextFloat() < 0.5) {
     i.remove();
     j.remove();
     t.remove(J);
    }
   }
   else if (r.nextFloat() < .2 && i.hasPrevious()) {
#ifdef Custom
    ensure(m.strategy().equals(i.previous(), J = j.previous()), "Error (" + seed + "): divergence in previous() (iterator with starting point " + from + ")");
#else
    ensure(i.previous().equals(J = j.previous()), "Error (" + seed + "): divergence in previous() (iterator with starting point " + from + ")");
#endif

    if (r.nextFloat() < 0.5) {
     i.remove();
     j.remove();
     t.remove(J);
    }
   }

   ensure(i.nextIndex() == j.nextIndex(), "Error (" + seed + "): divergence in nextIndex() (iterator with starting point " + from + ")");
   ensure(i.previousIndex() == j.previousIndex(), "Error (" + seed + "): divergence in previousIndex() (iterator with
```

```
starting point " + from + ")");

   }

  }



  /* Now we check that m actually holds that data. */

  ensure(m.equals(t), "Error (" + seed + "): ! m.equals(t) after iteration");
  ensure(t.equals(m), "Error (" + seed + "): ! t.equals(m) after iteration");

#endif


  /* Now we take out of m everything, and check that it is empty. */

  for(java.util.Iterator i=t.keySet().iterator(); i.hasNext();) m.remove(i.next());

  if (!m.isEmpty())  {
   System.out.println("Error (" + seed + "): m is not empty (as it should be)");
   System.exit(1);
  }

#ifdef NumericEnhancements
#if VALUE_CLASS_Byte || VALUE_CLASS_Character || VALUE_CLASS_Short || VALUE_CLASS_Integer ||
VALUE_CLASS_Long
  /* Now we check that increment works properly, using random data */

  {
   t.clear();
   m.clear();

   for(int k = 0; k < 2*n; k++) {
    KEY_TYPE T = genKey();
    VALUE_TYPE U = genValue();

    VALUE_TYPE rU = m.increment(T, U);
    VALUE_GENERIC_CLASS tU = (VALUE_GENERIC_CLASS) t.get(KEY2OBJ(T));
    if (null == tU) {
     ensure(m.defaultReturnValue() == rU, "Error (" + seed + "): map increment does not return proper starting value.");
     t.put(KEY2OBJ(T), VALUE2OBJ((VALUE_TYPE) (m.defaultReturnValue() + U)));
    }
    else {
     t.put(KEY2OBJ(T), VALUE2OBJ((VALUE_TYPE) (((VALUE_TYPE) tU) + U)));
    }
   }
```

```
  // Maps should contain identical values
  ensure(new java.util.HashMap(m).equals(new java.util.HashMap(t)),
   "Error(" + seed + "): incremented maps are not equal.");
 }
#endif
#endif


#if (KEY_CLASS_Integer || KEY_CLASS_Long) && (VALUE_CLASS_Integer || VALUE_CLASS_Long)
 m = new OPEN_HASH_MAP(n, f);
 t.clear();
 int x;

 /* Now we torture-test the hash table. This part is implemented only for integers and longs. */

 int p = m.key.length;

 for(int i=0; i<p; i++) {
  for (int j=0; j<20; j++) {
   m.put(i+(r.nextInt() % 10)*p, 1);
   m.remove(i+(r.nextInt() % 10)*p);
  }

  for (int j=-10; j<10; j++) m.remove(i+j*p);
 }

 t.putAll(m);

 /* Now all table entries are REMOVED. */

 for(int i=0; i<(p*f)/10; i++) {
  for (int j=0; j<10; j++) {
   if (!valEquals(m.put(KEY2OBJ(x = i+(r.nextInt() % 10)*p), VALUE2OBJ(1)), t.put(KEY2OBJ(x),
VALUE2OBJ(1))))
    System.out.println("Error (" + seed + "): m and t differ on an entry during torture-test insertion.");
  }
 }

 if (!m.equals(t)) System.out.println("Error (" + seed + "): !m.equals(t) after torture-test insertion");
 if (!t.equals(m)) System.out.println("Error (" + seed + "): !t.equals(m) after torture-test insertion");

 for(int i=0; i<p/10; i++) {
  for (int j=0; j<10; j++) {
   if (!valEquals(m.remove(KEY2OBJ(x = i+(r.nextInt() % 10)*p)), t.remove(KEY2OBJ(x))))
    System.out.println("Error (" + seed + "): m and t differ on an entry during torture-test removal.");
  }
 }
```

```
  if (!m.equals(t)) System.out.println("Error (" + seed + "): !m.equals(t) after torture-test removal");
  if (!t.equals(m)) System.out.println("Error (" + seed + "): !t.equals(m) after torture-test removal");

  if (!m.equals(m.clone())) System.out.println("Error (" + seed + "): !m.equals(m.clone()) after torture-test removal");
  if (!((OPEN_HASH_MAP)m.clone()).equals(m)) System.out.println("Error (" + seed + "): !m.clone().equals(m)
after torture-test removal");

  m.trim();

  if (!m.equals(t)) System.out.println("Error (" + seed + "): !m.equals(t) after trim()");
  if (!t.equals(m)) System.out.println("Error (" + seed + "): !t.equals(m) after trim()");
#endif

  System.out.println("Test OK");
  return;

#endif
 }


 public static void main(String args[]) {
  float f = Hash.DEFAULT_LOAD_FACTOR;
  int n  = Integer.parseInt(args[1]);
  if (args.length>2) f = Float.parseFloat(args[2]);
  if (args.length > 3) r = new java.util.Random(seed = Long.parseLong(args[3]));

  try {
   if ("speedTest".equals(args[0]) || "speedComp".equals(args[0])) speedTest(n, f, "speedComp".equals(args[0]));
   else if ("test".equals(args[0])) runTest(n, f);
  } catch(Throwable e) {
   e.printStackTrace(System.err);
   System.err.println("seed: " + seed);
  }

 }

#endif

}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/OpenHashMap.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
```

```
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

package PACKAGE;

import it.unimi.dsi.fastutil.objects.AbstractObjectSortedSet;
import it.unimi.dsi.fastutil.objects.ObjectBidirectionalIterator;
import it.unimi.dsi.fastutil.objects.ObjectListIterator;
import it.unimi.dsi.fastutil.objects.ObjectSortedSet;

import VALUE_PACKAGE.VALUE_COLLECTION;
import VALUE_PACKAGE.VALUE_ABSTRACT_COLLECTION;
import VALUE_PACKAGE.VALUE_ITERATOR;

import java.util.Comparator;
import java.util.Iterator;
import java.util.Map;
import java.util.SortedMap;
import java.util.NoSuchElementException;

#if VALUES_PRIMITIVE
import VALUE_PACKAGE.VALUE_LIST_ITERATOR;
#endif

/** A type-specific AVL tree map with a fast, small-footprint implementation.
*
* <p>The iterators provided by the views of this class are type-specific {@linkplain
* it.unimi.dsi.fastutil.BidirectionalIterator bidirectional iterators}.
* Moreover, the iterator returned by {@code iterator()} can be safely cast
* to a type-specific {@linkplain java.util.ListIterator list iterator}.
*/

public class AVL_TREE_MAP KEY_VALUE_GENERIC extends ABSTRACT_SORTED_MAP
KEY_VALUE_GENERIC implements java.io.Serializable, Cloneable {

 /** A reference to the root entry. */
 protected transient Entry KEY_VALUE_GENERIC tree;
```

```
/** Number of entries in this map. */
protected int count;

/** The first key in this map. */
protected transient Entry KEY_VALUE_GENERIC firstEntry;

/** The last key in this map. */
protected transient Entry KEY_VALUE_GENERIC lastEntry;

/** Cached set of entries. */
protected transient ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> entries;

/** Cached set of keys. */
protected transient SORTED_SET KEY_GENERIC keys;

/** Cached collection of values. */
protected transient VALUE_COLLECTION VALUE_GENERIC values;

/** The value of this variable remembers, after a {@code put()}
 * or a {@code remove()}, whether the <em>domain</em> of the map
 * has been modified. */
protected transient boolean modified;

/** This map's comparator, as provided in the constructor. */
protected Comparator<? super KEY_GENERIC_CLASS> storedComparator;

/** This map's actual comparator; it may differ from {@link #storedComparator} because it is
 always a type-specific comparator, so it could be derived from the former by wrapping. */
protected transient KEY_COMPARATOR KEY_SUPER_GENERIC actualComparator;

private static final long serialVersionUID = -7046029254386353129L;

{
 allocatePaths();
}

/** Creates a new empty tree map.
 */

public AVL_TREE_MAP() {
 tree = null;
 count = 0;
}

/** Generates the comparator that will be actually used.
 *
 * <p>When a given {@link Comparator} is specified and stored in {@link
```

```
 * #storedComparator}, we must check whether it is type-specific.  If it is
 * so, we can used directly, and we store it in {@link #actualComparator}. Otherwise,
 * we adapt it using a helper static method.
 */
 private void setActualComparator() {
#if KEY_CLASS_Object
  actualComparator = storedComparator;
#else
  actualComparator = COMPARATORS.AS_KEY_COMPARATOR(storedComparator);
#endif
 }


 /** Creates a new empty tree map with the given comparator.
  *
  * @param c a (possibly type-specific) comparator.
  */

 public AVL_TREE_MAP(final Comparator<? super KEY_GENERIC_CLASS> c) {
  this();
  storedComparator = c;
  setActualComparator();
 }


 /** Creates a new tree map copying a given map.
  *
  * @param m a {@link Map} to be copied into the new tree map.
  */

 public AVL_TREE_MAP(final Map<? extends KEY_GENERIC_CLASS, ? extends VALUE_GENERIC_CLASS>
m) {
  this();
  putAll(m);
 }

 /** Creates a new tree map copying a given sorted map (and its {@link Comparator}).
  *
  * @param m a {@link SortedMap} to be copied into the new tree map.
  */

 public AVL_TREE_MAP(final SortedMap<KEY_GENERIC_CLASS,VALUE_GENERIC_CLASS> m) {
  this(m.comparator());
  putAll(m);
 }

 /** Creates a new tree map copying a given map.
  *
```

```
 * @param m a type-specific map to be copied into the new tree map.
 */

public AVL_TREE_MAP(final MAP KEY_VALUE_EXTENDS_GENERIC m) {
 this();
 putAll(m);
}

/** Creates a new tree map copying a given sorted map (and its {@link Comparator}).
 *
 * @param m a type-specific sorted map to be copied into the new tree map.
 */

public AVL_TREE_MAP(final SORTED_MAP KEY_VALUE_GENERIC m) {
 this(m.comparator());
 putAll(m);
}

/** Creates a new tree map using the elements of two parallel arrays and the given comparator.
 *
 * @param k the array of keys of the new tree map.
 * @param v the array of corresponding values in the new tree map.
 * @param c a (possibly type-specific) comparator.
 * @throws IllegalArgumentException if {@code k} and {@code v} have different lengths.
 */

public AVL_TREE_MAP(final KEY_GENERIC_TYPE[] k, final VALUE_GENERIC_TYPE v[], final
Comparator<? super KEY_GENERIC_CLASS> c) {
 this(c);
 if (k.length != v.length) throw new IllegalArgumentException("The key array and the value array have different
lengths (" + k.length + " and " + v.length + ")");
 for(int i = 0; i < k.length; i++) this.put(k[i], v[i]);
}

/** Creates a new tree map using the elements of two parallel arrays.
 *
 * @param k the array of keys of the new tree map.
 * @param v the array of corresponding values in the new tree map.
 * @throws IllegalArgumentException if {@code k} and {@code v} have different lengths.
 */

public AVL_TREE_MAP(final KEY_GENERIC_TYPE[] k, final VALUE_GENERIC_TYPE v[]) {
 this(k, v, null);
}

/*
 * The following methods implements some basic building blocks used by
 * all accessors.  They are (and should be maintained) identical to those used in AVLTreeSet.drv.
```

```
 *
 * The put()/remove() code is derived from Ben Pfaff's GNU libavl
 * (http://www.msu.edu/~pfaffben/avl/). If you want to understand what's
 * going on, you should have a look at the literate code contained therein
 * first.
 */


/** Compares two keys in the right way.
 *
 * <p>This method uses the {@link #actualComparator} if it is non-{@code null}.
 * Otherwise, it resorts to primitive type comparisons or to {@link Comparable#compareTo(Object) compareTo()}.
 *
 * @param k1 the first key.
 * @param k2 the second key.
 * @return a number smaller than, equal to or greater than 0, as usual
 * (i.e., when k1 &lt; k2, k1 = k2 or k1 &gt; k2, respectively).
 */

SUPPRESS_WARNINGS_KEY_UNCHECKED
final int compare(final KEY_GENERIC_TYPE k1, final KEY_GENERIC_TYPE k2) {
 return actualComparator == null ? KEY_CMP(k1, k2) : actualComparator.compare(k1, k2);
}



/** Returns the entry corresponding to the given key, if it is in the tree; {@code null}, otherwise.
 *
 * @param k the key to search for.
 * @return the corresponding entry, or {@code null} if no entry with the given key exists.
 */

final Entry KEY_VALUE_GENERIC findKey(final KEY_GENERIC_TYPE k) {
 Entry KEY_VALUE_GENERIC e = tree;
 int cmp;

 while (e != null && (cmp = compare(k, e.key)) != 0) e = cmp < 0 ? e.left() : e.right();

 return e;
}

/** Locates a key.
 *
 * @param k a key.
 * @return the last entry on a search for the given key; this will be
 * the given key, if it present; otherwise, it will be either the smallest greater key or the greatest smaller key.
 */
```

```
final Entry KEY_VALUE_GENERIC locateKey(final KEY_GENERIC_TYPE k) {
 Entry KEY_VALUE_GENERIC e = tree, last = tree;
 int cmp = 0;

 while (e != null && (cmp = compare(k, e.key)) != 0) {
  last = e;
  e = cmp < 0 ? e.left() : e.right();
 }

 return cmp == 0 ? e : last;
}

/** This vector remembers the directions followed during
 * the current insertion. It suffices for about 2<sup>32</sup> entries. */
private transient boolean dirPath[];

private void allocatePaths() {
 dirPath = new boolean[48];
}

#if VALUES_PRIMITIVE && !VALUE_CLASS_Boolean
/** Adds an increment to value currently associated with a key.
*
* <p>Note that this method respects the {@linkplain #defaultReturnValue() default return value} semantics: when
* called with a key that does not currently appears in the map, the key
* will be associated with the default return value plus
* the given increment.
*
* @param k the key.
* @param incr the increment.
* @return the old value, or the {@linkplain #defaultReturnValue() default return value} if no value was present for
the given key.
*/
public VALUE_GENERIC_TYPE addTo(final KEY_GENERIC_TYPE k, final VALUE_GENERIC_TYPE incr) {
 Entry KEY_VALUE_GENERIC e = add(k);
 final VALUE_GENERIC_TYPE oldValue = e.value;
 e.value += incr;
 return oldValue;
}
#endif

@Override
public VALUE_GENERIC_TYPE put(final KEY_GENERIC_TYPE k, final VALUE_GENERIC_TYPE v) {
 Entry KEY_VALUE_GENERIC e = add(k);
 final VALUE_GENERIC_TYPE oldValue = e.value;
 e.value = v;
 return oldValue;
}
```

```
/** Returns a node with key k in the balanced tree, creating one with defRetValue if necessary.
*
* @param k the key
* @return a node with key k. If a node with key k already exists, then that node is returned,
*     otherwise a new node with defRetValue is created ensuring that the tree is balanced
    after creation of the node.
*/
private Entry KEY_VALUE_GENERIC add(final KEY_GENERIC_TYPE k) {
 /* After execution of this method, modified is true iff a new entry has
 been inserted. */
 modified = false;

 Entry KEY_VALUE_GENERIC e = null;
 if (tree == null) { // The case of the empty tree is treated separately.
  count++;
  e = tree = lastEntry = firstEntry = new Entry KEY_VALUE_GENERIC_DIAMOND(k, defRetValue);
  modified = true;
 }
 else {
  Entry KEY_VALUE_GENERIC p = tree, q = null, y = tree, z = null, w = null;
  int cmp, i = 0;

  while(true) {
   if ((cmp = compare(k, p.key)) == 0) {
    return p;
   }

   if (p.balance() != 0) {
    i = 0;
    z = q;
    y = p;
   }

   if (dirPath[i++] = cmp > 0) {
    if (p.succ()) {
     count++;
     e = new Entry KEY_VALUE_GENERIC_DIAMOND(k, defRetValue);

     modified = true;
     if (p.right == null) lastEntry = e;

     e.left = p;
     e.right = p.right;

     p.right(e);

     break;
```

```
 }

 q = p;
 p = p.right;
}
else {
 if (p.pred()) {
  count++;
  e = new Entry KEY_VALUE_GENERIC_DIAMOND(k, defRetValue);

  modified = true;
  if (p.left == null) firstEntry = e;

  e.right = p;
  e.left = p.left;

  p.left(e);

  break;
 }

 q = p;
 p = p.left;
 }
}

p = y;
i = 0;

while(p != e) {
 if (dirPath[i]) p.incBalance();
 else p.decBalance();

 p = dirPath[i++] ? p.right : p.left;
}

if (y.balance() == -2) {
 Entry KEY_VALUE_GENERIC x = y.left;

 if (x.balance() == -1) {
  w = x;
  if (x.succ()) {
   x.succ(false);
   y.pred(x);
  }
  else y.left = x.right;

  x.right = y;
```

```
 x.balance(0);
 y.balance(0);
 }
 else {
 assert x.balance() == 1;

 w = x.right;
 x.right = w.left;
 w.left = x;
 y.left = w.right;
 w.right = y;
 if (w.balance() == -1) {
  x.balance(0);
  y.balance(1);
 }
 else if (w.balance() == 0) {
  x.balance(0);
  y.balance(0);
 }
 else {
  x.balance(-1);
  y.balance(0);
 }
 w.balance(0);


 if (w.pred()) {
  x.succ(w);
  w.pred(false);
 }
 if (w.succ()) {
  y.pred(w);
  w.succ(false);
 }

 }
}
else if (y.balance() == +2) {
 Entry KEY_VALUE_GENERIC x = y.right;

 if (x.balance() == 1) {
  w = x;
  if (x.pred()) {
   x.pred(false);
   y.succ(x);
  }
  else y.right = x.left;
```

```
  x.left = y;
  x.balance(0);
  y.balance(0);
 }
 else {
  assert x.balance() == -1;

  w = x.left;
  x.left = w.right;
  w.right = x;
  y.right = w.left;
  w.left = y;
  if (w.balance() == 1) {
   x.balance(0);
   y.balance(-1);
  }
  else if (w.balance() == 0) {
   x.balance(0);
   y.balance(0);
  }
  else {
   x.balance(1);
   y.balance(0);
  }
  w.balance(0);


  if (w.pred()) {
   y.succ(w);
   w.pred(false);
  }
  if (w.succ()) {
   x.pred(w);
   w.succ(false);
  }

 }
}
else return e;

if (z == null) tree = w;
else {
 if (z.left == y) z.left = w;
 else z.right = w;
}
}

return e;
```

```
}

/** Finds the parent of an entry.
 *
 * @param e a node of the tree.
 * @return the parent of the given node, or {@code null} for the root.
 */

private Entry KEY_VALUE_GENERIC parent(final Entry KEY_VALUE_GENERIC e) {
 if (e == tree) return null;

 Entry KEY_VALUE_GENERIC x, y, p;
 x = y = e;

 while(true) {
  if (y.succ()) {
   p = y.right;
   if (p == null || p.left != e) {
    while(! x.pred()) x = x.left;
    p = x.left;
   }
   return p;
  }
  else if (x.pred()) {
   p = x.left;
   if (p == null || p.right != e) {
    while(! y.succ()) y = y.right;
    p = y.right;
   }
   return p;
  }

  x = x.left;
  y = y.right;
 }
}


/* After execution of this method, {@link #modified} is true iff an entry
has been deleted. */

SUPPRESS_WARNINGS_KEY_UNCHECKED
@Override
public VALUE_GENERIC_TYPE REMOVE_VALUE(final KEY_TYPE k) {
 modified = false;

 if (tree == null) return defRetValue;
```

```
int cmp;
Entry KEY_VALUE_GENERIC p = tree, q = null;
boolean dir = false;
final KEY_GENERIC_TYPE kk = KEY_GENERIC_CAST k;

while(true) {
 if ((cmp = compare(kk, p.key)) == 0) break;
 else if (dir = cmp > 0) {
  q = p;
  if ((p = p.right()) == null) return defRetValue;
 }
 else {
  q = p;
  if ((p = p.left()) == null) return defRetValue;
 }
}

if (p.left == null) firstEntry = p.next();
if (p.right == null) lastEntry = p.prev();

if (p.succ()) {
 if (p.pred()) {
  if (q != null) {
   if (dir) q.succ(p.right);
   else q.pred(p.left);
  }
  else tree = dir ? p.right : p.left;
 }
 else {
  p.prev().right = p.right;

  if (q != null) {
   if (dir) q.right = p.left;
   else q.left = p.left;
  }
  else tree = p.left;
 }
}
else {
 Entry KEY_VALUE_GENERIC r = p.right;

 if (r.pred()) {
  r.left = p.left;
  r.pred(p.pred());
  if (! r.pred()) r.prev().right = r;
  if (q != null) {
   if (dir) q.right = r;
   else q.left = r;
```

```
 }
 else tree = r;

 r.balance(p.balance());
 q = r;
 dir = true;

}
else {
 Entry KEY_VALUE_GENERIC s;

 while(true) {
  s = r.left;
  if (s.pred()) break;
  r = s;
 }

 if (s.succ()) r.pred(s);
 else r.left = s.right;

 s.left = p.left;

 if (! p.pred()) {
  p.prev().right = s;
  s.pred(false);
 }

 s.right = p.right;
 s.succ(false);

 if (q != null) {
  if (dir) q.right = s;
  else q.left = s;
 }
 else tree = s;

 s.balance(p.balance());
 q = r;
 dir = false;
 }
}

Entry KEY_VALUE_GENERIC y;

while(q != null) {
 y = q;
 q = parent(y);
```

```
if (! dir) {
 dir = q != null && q.left != y;
 y.incBalance();

 if (y.balance() == 1) break;
 else if (y.balance() == 2) {

  Entry KEY_VALUE_GENERIC x = y.right;
  assert x != null;

  if (x.balance() == -1) {
   Entry KEY_VALUE_GENERIC w;

   assert x.balance() == -1;

   w = x.left;
   x.left = w.right;
   w.right = x;
   y.right = w.left;
   w.left = y;

   if (w.balance() == 1) {
    x.balance(0);
    y.balance(-1);
   }
   else if (w.balance() == 0) {
    x.balance(0);
    y.balance(0);
   }
   else {
    assert w.balance() == -1;

    x.balance(1);
    y.balance(0);
   }

   w.balance(0);

   if (w.pred()) {
    y.succ(w);
    w.pred(false);
   }
   if (w.succ()) {
    x.pred(w);
    w.succ(false);
   }

   if (q != null) {
```

```
   if (dir) q.right = w;
   else q.left = w;
  }
  else tree = w;
 }
 else {
  if (q != null) {
   if (dir) q.right = x;
   else q.left = x;
  }
  else tree = x;

  if (x.balance() == 0) {
   y.right = x.left;
   x.left = y;
   x.balance(-1);
   y.balance(+1);
   break;
  }
  assert x.balance() == 1;

  if (x.pred()) {
   y.succ(true);
   x.pred(false);
  }
  else y.right = x.left;

  x.left = y;
  y.balance(0);
  x.balance(0);
 }
 }
}
else {
 dir = q != null && q.left != y;
 y.decBalance();

 if (y.balance() == -1) break;
 else if (y.balance() == -2) {

  Entry KEY_VALUE_GENERIC x = y.left;
  assert x != null;

  if (x.balance() == 1) {
   Entry KEY_VALUE_GENERIC w;

   assert x.balance() == 1;
```

```
 w = x.right;
 x.right = w.left;
 w.left = x;
 y.left = w.right;
 w.right = y;

 if (w.balance() == -1) {
  x.balance(0);
  y.balance(1);
 }
 else if (w.balance() == 0) {
  x.balance(0);
  y.balance(0);
 }
 else {
  assert w.balance() == 1;

  x.balance(-1);
  y.balance(0);
 }

 w.balance(0);

 if (w.pred()) {
  x.succ(w);
  w.pred(false);
 }
 if (w.succ()) {
  y.pred(w);
  w.succ(false);
 }

 if (q != null) {
  if (dir) q.right = w;
  else q.left = w;
 }
 else tree = w;
}
else {
 if (q != null) {
  if (dir) q.right = x;
  else q.left = x;
 }
 else tree = x;

 if (x.balance() == 0) {
  y.left = x.right;
  x.right = y;
```

```
      x.balance(+1);
      y.balance(-1);
      break;
     }
     assert x.balance() == -1;

     if (x.succ()) {
      y.pred(true);
      x.succ(false);
     }
     else y.left = x.right;

     x.right = y;
     y.balance(0);
     x.balance(0);
    }
   }
  }
 }

 modified = true;
 count--;
 return p.value;
}


@Override
public boolean containsValue(final VALUE_TYPE v) {
 final ValueIterator i = new ValueIterator();
 VALUE_GENERIC_TYPE ev;

 int j = count;
 while(j-- != 0) {
  ev = i.NEXT_VALUE();
  if (VALUE_EQUALS(ev, v)) return true;
 }

 return false;
}

@Override
public void clear() {
 count = 0;
 tree = null;
 entries = null;
 values = null;
 keys = null;
 firstEntry = lastEntry = null;
```

```
}


/** This class represent an entry in a tree map.
 *
 * <p>We use the only "metadata", i.e., {@link Entry#info}, to store
 * information about balance, predecessor status and successor status.
 *
 * <p>Note that since the class is recursive, it can be
 * considered equivalently a tree.
 */

private static final class Entry KEY_VALUE_GENERIC extends ABSTRACT_MAP.BasicEntry
KEY_VALUE_GENERIC implements Cloneable {
 /** If the bit in this mask is true, {@link #right} points to a successor. */
 private static final int SUCC_MASK = 1 << 31;
 /** If the bit in this mask is true, {@link #left} points to a predecessor. */
 private static final int PRED_MASK = 1 << 30;
 /** The bits in this mask hold the node balance info. You can get it just by casting to byte. */
 private static final int BALANCE_MASK = 0xFF;
 /** The pointers to the left and right subtrees. */
 Entry KEY_VALUE_GENERIC left, right;
 /** This integers holds different information in different bits (see {@link #SUCC_MASK}, {@link
#PRED_MASK} and {@link #BALANCE_MASK}). */
 int info;

 Entry() {
  super(KEY_NULL, VALUE_NULL);
 }

 /** Creates a new entry with the given key and value.
  *
  * @param k a key.
  * @param v a value.
  */
 Entry(final KEY_GENERIC_TYPE k, final VALUE_GENERIC_TYPE v) {
  super(k, v);
  info = SUCC_MASK | PRED_MASK;
 }

 /** Returns the left subtree.
  *
  * @return the left subtree ({@code null} if the left
  * subtree is empty).
  */
 Entry KEY_VALUE_GENERIC left() {
  return (info & PRED_MASK) != 0 ? null : left;
 }
```

```
/** Returns the right subtree.
 *
 * @return the right subtree ({@code null} if the right
 * subtree is empty).
 */
Entry KEY_VALUE_GENERIC right() {
 return (info & SUCC_MASK) != 0 ? null : right;
}

/** Checks whether the left pointer is really a predecessor.
 * @return true if the left pointer is a predecessor.
 */
boolean pred() {
 return (info & PRED_MASK) != 0;
}

/** Checks whether the right pointer is really a successor.
 * @return true if the right pointer is a successor.
 */
boolean succ() {
 return (info & SUCC_MASK) != 0;
}

/** Sets whether the left pointer is really a predecessor.
 * @param pred if true then the left pointer will be considered a predecessor.
 */
void pred(final boolean pred) {
 if (pred) info |= PRED_MASK;
 else info &= ~PRED_MASK;
}

/** Sets whether the right pointer is really a successor.
 * @param succ if true then the right pointer will be considered a successor.
 */
void succ(final boolean succ) {
 if (succ) info |= SUCC_MASK;
 else info &= ~SUCC_MASK;
}

/** Sets the left pointer to a predecessor.
 * @param pred the predecessr.
 */
void pred(final Entry KEY_VALUE_GENERIC pred) {
 info |= PRED_MASK;
 left = pred;
}
```

```
/** Sets the right pointer to a successor.
 * @param succ the successor.
 */
void succ(final Entry KEY_VALUE_GENERIC succ) {
 info |= SUCC_MASK;
 right = succ;
}

/** Sets the left pointer to the given subtree.
 * @param left the new left subtree.
 */
void left(final Entry KEY_VALUE_GENERIC left) {
 info &= ~PRED_MASK;
 this.left = left;
}

/** Sets the right pointer to the given subtree.
 * @param right the new right subtree.
 */
void right(final Entry KEY_VALUE_GENERIC right) {
 info &= ~SUCC_MASK;
 this.right = right;
}

/** Returns the current level of the node.
 * @return the current level of this node.
 */
int balance() {
 return (byte)info;
}

/** Sets the level of this node.
 * @param level the new level of this node.
 */
void balance(int level) {
 info &= ~BALANCE_MASK;
 info |= (level & BALANCE_MASK);
}

/** Increments the level of this node. */
void incBalance() {
 info = info & ~BALANCE_MASK | ((byte)info + 1) & 0xFF;
}

/** Decrements the level of this node. */
protected void decBalance() {
 info = info & ~BALANCE_MASK | ((byte)info - 1) & 0xFF;
}
```

```
/** Computes the next entry in the set order.
 *
 * @return the next entry ({@code null}) if this is the last entry).
 */

Entry KEY_VALUE_GENERIC next() {
 Entry KEY_VALUE_GENERIC next = this.right;
 if ((info & SUCC_MASK) == 0) while ((next.info & PRED_MASK) == 0) next = next.left;
 return next;
}

/** Computes the previous entry in the set order.
 *
 * @return the previous entry ({@code null}) if this is the first entry).
 */

Entry KEY_VALUE_GENERIC prev() {
 Entry KEY_VALUE_GENERIC prev = this.left;
 if ((info & PRED_MASK) == 0) while ((prev.info & SUCC_MASK) == 0) prev = prev.right;
 return prev;
}

@Override
public VALUE_GENERIC_TYPE setValue(final VALUE_GENERIC_TYPE value) {
 final VALUE_GENERIC_TYPE oldValue = this.value;
 this.value = value;
 return oldValue;
}

@Override
SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
public Entry KEY_VALUE_GENERIC clone() {
 Entry KEY_VALUE_GENERIC c;
 try {
  c = (Entry KEY_VALUE_GENERIC)super.clone();
 }
 catch(CloneNotSupportedException cantHappen) {
  throw new InternalError();
 }

 c.key = key;
 c.value = value;
 c.info = info;

 return c;
}
```

```
 @Override
 @SuppressWarnings("unchecked")
 public boolean equals(final Object o) {
  if (!(o instanceof Map.Entry)) return false;
  Map.Entry<KEY_GENERIC_CLASS, VALUE_GENERIC_CLASS> e = (Map.Entry<KEY_GENERIC_CLASS,
VALUE_GENERIC_CLASS>)o;

  return KEY_EQUALS(key, KEY_CLASS2TYPE(e.getKey())) && VALUE_EQUALS(value,
VALUE_CLASS2TYPE(e.getValue()));
 }

 @Override
 public int hashCode() {
  return KEY2JAVAHASH_NOT_NULL(key) ^ VALUE2JAVAHASH(value);
 }



 @Override
 public String toString() {
  return key + "=>" + value;
 }



 /*
 public void prettyPrint() {
  prettyPrint(0);
 }

 public void prettyPrint(int level) {
  if (pred()) {
   for (int i = 0; i < level; i++)
    System.err.print("  ");
   System.err.println("pred: " + left);
  }
  else if (left != null)
   left.prettyPrint(level +1);
  for (int i = 0; i < level; i++)
   System.err.print("  ");
  System.err.println(key + "=" + value + " (" + balance() + ")");
  if (succ()) {
   for (int i = 0; i < level; i++)
    System.err.print("  ");
   System.err.println("succ: " + right);
  }
  else if (right != null)
   right.prettyPrint(level + 1);
 }
 */
```

```
}

/*
public void prettyPrint() {
 System.err.println("size: " + count);
 if (tree != null) tree.prettyPrint();
}
*/

SUPPRESS_WARNINGS_KEY_UNCHECKED
@Override
public boolean containsKey(final KEY_TYPE k) {
 return findKey(KEY_GENERIC_CAST k) != null;
}

@Override
public int size() {
 return count;
}

@Override
public boolean isEmpty() {
 return count == 0;
}

SUPPRESS_WARNINGS_KEY_UNCHECKED
@Override
public VALUE_GENERIC_TYPE GET_VALUE(final KEY_TYPE k) {
 final Entry KEY_VALUE_GENERIC e = findKey(KEY_GENERIC_CAST k);
 return e == null ? defRetValue : e.value;
}

@Override
public KEY_GENERIC_TYPE FIRST_KEY() {
 if (tree == null) throw new NoSuchElementException();
 return firstEntry.key;
}

@Override
public KEY_GENERIC_TYPE LAST_KEY() {
 if (tree == null) throw new NoSuchElementException();
 return lastEntry.key;
}



/** An abstract iterator on the whole range.
 *
 * <p>This class can iterate in both directions on a threaded tree.
```

```
 */

 private class TreeIterator {
  /** The entry that will be returned by the next call to {@link java.util.ListIterator#previous()} (or {@code null} if no previous entry exists). */
  Entry KEY_VALUE_GENERIC prev;
  /** The entry that will be returned by the next call to {@link java.util.ListIterator#next()} (or {@code null} if no next entry exists). */
  Entry KEY_VALUE_GENERIC next;
  /** The last entry that was returned (or {@code null} if we did not iterate or used {@link #remove()}). */
  Entry KEY_VALUE_GENERIC curr;
  /** The current index (in the sense of a {@link java.util.ListIterator}). Note that this value is not meaningful when this {@link TreeIterator} has been created using the nonempty constructor.*/
  int index = 0;

  TreeIterator() {
   next = firstEntry;
  }

  TreeIterator(final KEY_GENERIC_TYPE k) {
   if ((next = locateKey(k)) != null) {
    if (compare(next.key, k) <= 0) {
     prev = next;
     next = next.next();
    }
    else prev = next.prev();
   }
  }

  public boolean hasNext() { return next != null; }

  public boolean hasPrevious() { return prev != null; }

  void updateNext() {
   next = next.next();
  }

  Entry KEY_VALUE_GENERIC nextEntry() {
   if (! hasNext()) throw new NoSuchElementException();
   curr = prev = next;
   index++;
   updateNext();
   return curr;
  }

  void updatePrevious() {
   prev = prev.prev();
  }
```

```
Entry KEY_VALUE_GENERIC previousEntry() {
 if (! hasPrevious()) throw new NoSuchElementException();
 curr = next = prev;
 index--;
 updatePrevious();
 return curr;
 }

public int nextIndex() {
 return index;
 }

public int previousIndex() {
 return index - 1;
 }

public void remove() {
 if (curr == null) throw new IllegalStateException();
 /* If the last operation was a next(), we are removing an entry that preceeds
    the current index, and thus we must decrement it. */
 if (curr == prev) index--;
 next = prev = curr;
 updatePrevious();
 updateNext();
 AVL_TREE_MAP.this.REMOVE_VALUE(curr.key);
 curr = null;
 }

public int skip(final int n) {
 int i = n;
 while(i-- != 0 && hasNext()) nextEntry();
 return n - i - 1;
 }

public int back(final int n) {
 int i = n;
 while(i-- != 0 && hasPrevious()) previousEntry();
 return n - i - 1;
 }
}


/** An iterator on the whole range.
 *
 * <p>This class can iterate in both directions on a threaded tree.
 */
```

```
 private class EntryIterator extends TreeIterator implements ObjectListIterator<MAP.Entry
KEY_VALUE_GENERIC> {
  EntryIterator() {}

  EntryIterator(final KEY_GENERIC_TYPE k) {
   super(k);
  }

  @Override
  public MAP.Entry KEY_VALUE_GENERIC next() { return nextEntry(); }

  @Override
  public MAP.Entry KEY_VALUE_GENERIC previous() { return previousEntry(); }

  @Override
  public void set(MAP.Entry KEY_VALUE_GENERIC ok) { throw new UnsupportedOperationException(); }

  @Override
  public void add(MAP.Entry KEY_VALUE_GENERIC ok) { throw new UnsupportedOperationException(); }
 }



 @Override
 public ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> ENTRYSET() {
  if (entries == null) entries = new AbstractObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC>() {
    final Comparator<? super MAP.Entry KEY_VALUE_GENERIC> comparator = (Comparator<MAP.Entry
KEY_VALUE_GENERIC>) (x, y) -> AVL_TREE_MAP.this.actualComparator.compare(x.ENTRY_GET_KEY(),
y.ENTRY_GET_KEY());

    @Override
    public Comparator<? super MAP.Entry KEY_VALUE_GENERIC> comparator() { return comparator; }

    @Override
    public ObjectBidirectionalIterator<MAP.Entry KEY_VALUE_GENERIC> iterator() { return new EntryIterator(); }

    @Override
    public ObjectBidirectionalIterator<MAP.Entry KEY_VALUE_GENERIC> iterator(final MAP.Entry
KEY_VALUE_GENERIC from) { return new EntryIterator(from.ENTRY_GET_KEY()); }

    @Override
    SUPPRESS_WARNINGS_KEY_UNCHECKED
    public boolean contains(final Object o) {
     if (!(o instanceof Map.Entry)) return false;
     final Map.Entry<?,?> e = (Map.Entry<?,?>)o;
#if KEYS_PRIMITIVE
     if (e.getKey() == null || ! (e.getKey() instanceof KEY_CLASS)) return false;
#endif
#if VALUES_PRIMITIVE
```

```
    if (e.getValue() == null || ! (e.getValue() instanceof VALUE_CLASS)) return false;
#endif
    final Entry KEY_VALUE_GENERIC f = findKey(KEY_OBJ2TYPE(KEY_GENERIC_CAST e.getKey()));
    return e.equals(f);
   }

   @Override
   SUPPRESS_WARNINGS_KEY_UNCHECKED
   public boolean remove(final Object o) {
    if (!(o instanceof Map.Entry)) return false;
    final Map.Entry<?,?> e = (Map.Entry<?,?>)o;
#if KEYS_PRIMITIVE
    if (e.getKey() == null || ! (e.getKey() instanceof KEY_CLASS)) return false;
#endif
#if VALUES_PRIMITIVE
    if (e.getValue() == null || ! (e.getValue() instanceof VALUE_CLASS)) return false;
#endif
    final Entry KEY_VALUE_GENERIC f = findKey(KEY_OBJ2TYPE(KEY_GENERIC_CAST e.getKey()));
    if (f == null || ! VALUE_EQUALS(f.ENTRY_GET_VALUE(), VALUE_OBJ2TYPE(e.getValue()))) return false;
    AVL_TREE_MAP.this.REMOVE_VALUE(f.key);
    return true;
   }

   @Override
   public int size() { return count; }

   @Override
   public void clear() { AVL_TREE_MAP.this.clear(); }

   @Override
   public MAP.Entry KEY_VALUE_GENERIC first() { return firstEntry; }

   @Override
   public MAP.Entry KEY_VALUE_GENERIC last() { return lastEntry; }

   @Override
   public ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> subSet(MAP.Entry KEY_VALUE_GENERIC
from, MAP.Entry KEY_VALUE_GENERIC to) { return subMap(from.ENTRY_GET_KEY(),
to.ENTRY_GET_KEY()).ENTRYSET(); }

   @Override
   public ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> headSet(MAP.Entry KEY_VALUE_GENERIC
to) { return headMap(to.ENTRY_GET_KEY()).ENTRYSET(); }

   @Override
   public ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> tailSet(MAP.Entry KEY_VALUE_GENERIC
from) { return tailMap(from.ENTRY_GET_KEY()).ENTRYSET(); }
  };
```

```
 return entries;
 }

 /** An iterator on the whole range of keys.
  *
  * <p>This class can iterate in both directions on the keys of a threaded tree. We
  * simply override the {@link java.util.ListIterator#next()}/{@link java.util.ListIterator#previous()} methods (and
possibly
  * their type-specific counterparts) so that they return keys instead of entries.
  */
 private final class KeyIterator extends TreeIterator implements KEY_LIST_ITERATOR KEY_GENERIC {
  public KeyIterator() {}
  public KeyIterator(final KEY_GENERIC_TYPE k) { super(k); }
  @Override
  public KEY_GENERIC_TYPE NEXT_KEY() { return nextEntry().key; }
  @Override
  public KEY_GENERIC_TYPE PREV_KEY() { return previousEntry().key; }
 }

 /** A keyset implementation using a more direct implementation for iterators. */
 private class KeySet extends ABSTRACT_SORTED_MAP KEY_VALUE_GENERIC.KeySet {
  @Override
  public KEY_BIDI_ITERATOR KEY_GENERIC iterator() { return new KeyIterator(); }
  @Override
  public KEY_BIDI_ITERATOR KEY_GENERIC iterator(final KEY_GENERIC_TYPE from) { return new
KeyIterator(from); }
 }

 /** Returns a type-specific sorted set view of the keys contained in this map.
  *
  * <p>In addition to the semantics of {@link java.util.Map#keySet()}, you can
  * safely cast the set returned by this call to a type-specific sorted
  * set interface.
  *
  * @return a type-specific sorted set view of the keys contained in this map.
  */
 @Override
 public SORTED_SET KEY_GENERIC keySet() {
  if (keys == null) keys = new KeySet();
  return keys;
 }

 /** An iterator on the whole range of values.
  *
  * <p>This class can iterate in both directions on the values of a threaded tree. We
  * simply override the {@link java.util.ListIterator#next()}/{@link java.util.ListIterator#previous()} methods (and
possibly
```

```
 * their type-specific counterparts) so that they return values instead of entries.
 */
private final class ValueIterator extends TreeIterator implements VALUE_LIST_ITERATOR VALUE_GENERIC {
 @Override
 public VALUE_GENERIC_TYPE NEXT_VALUE() { return nextEntry().value; }
 @Override
 public VALUE_GENERIC_TYPE PREV_VALUE() { return previousEntry().value; }
}

/** Returns a type-specific collection view of the values contained in this map.
 *
 * <p>In addition to the semantics of {@link java.util.Map#values()}, you can
 * safely cast the collection returned by this call to a type-specific collection
 * interface.
 *
 * @return a type-specific collection view of the values contained in this map.
 */
@Override
public VALUE_COLLECTION VALUE_GENERIC values() {
 if (values == null) values = new VALUE_ABSTRACT_COLLECTION VALUE_GENERIC() {
   @Override
   public VALUE_ITERATOR VALUE_GENERIC iterator() { return new ValueIterator(); }
   @Override
   public boolean contains(final VALUE_TYPE k) { return containsValue(k); }
   @Override
   public int size() { return count; }
   @Override
   public void clear() { AVL_TREE_MAP.this.clear(); }
  };

 return values;
}

@Override
public KEY_COMPARATOR KEY_SUPER_GENERIC comparator() { return actualComparator; }

@Override
public SORTED_MAP KEY_VALUE_GENERIC headMap(KEY_GENERIC_TYPE to) { return new
Submap(KEY_NULL, true, to, false); }

@Override
public SORTED_MAP KEY_VALUE_GENERIC tailMap(KEY_GENERIC_TYPE from) { return new
Submap(from, false, KEY_NULL, true); }

@Override
public SORTED_MAP KEY_VALUE_GENERIC subMap(KEY_GENERIC_TYPE from, KEY_GENERIC_TYPE
to) { return new Submap(from, false, to, false); }
```

```
 /** A submap with given range.
  *
  * <p>This class represents a submap. One has to specify the left/right
  * limits (which can be set to -&infin; or &infin;). Since the submap is a
  * view on the map, at a given moment it could happen that the limits of
  * the range are not any longer in the main map. Thus, things such as
  * {@link java.util.SortedMap#firstKey()} or {@link java.util.Collection#size()} must be always computed
  * on-the-fly.
  */
 private final class Submap extends ABSTRACT_SORTED_MAP KEY_VALUE_GENERIC implements
java.io.Serializable {
  private static final long serialVersionUID = -7046029254386353129L;

  /** The start of the submap range, unless {@link #bottom} is true. */
  KEY_GENERIC_TYPE from;
  /** The end of the submap range, unless {@link #top} is true. */
  KEY_GENERIC_TYPE to;
  /** If true, the submap range starts from -&infin;. */
  boolean bottom;
  /** If true, the submap range goes to &infin;. */
  boolean top;
  /** Cached set of entries. */
  protected transient ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> entries;
  /** Cached set of keys. */
  protected transient SORTED_SET KEY_GENERIC keys;
  /** Cached collection of values. */
  protected transient VALUE_COLLECTION VALUE_GENERIC values;

  /** Creates a new submap with given key range.
   *
   * @param from the start of the submap range.
   * @param bottom if true, the first parameter is ignored and the range starts from -&infin;.
   * @param to the end of the submap range.
   * @param top if true, the third parameter is ignored and the range goes to &infin;.
   */
  public Submap(final KEY_GENERIC_TYPE from, final boolean bottom, final KEY_GENERIC_TYPE to, final
boolean top) {
   if (! bottom && ! top && AVL_TREE_MAP.this.compare(from, to) > 0) throw new
IllegalArgumentException("Start key (" + from  + ") is larger than end key (" + to + ")");

   this.from = from;
   this.bottom = bottom;
   this.to = to;
   this.top = top;
   this.defRetValue = AVL_TREE_MAP.this.defRetValue;
  }

  @Override
```

```
 public void clear() {
  final SubmapIterator i = new SubmapIterator();
  while(i.hasNext()) {
   i.nextEntry();
   i.remove();
  }
 }

 /** Checks whether a key is in the submap range.
  * @param k a key.
  * @return true if is the key is in the submap range.
  */
 final boolean in(final KEY_GENERIC_TYPE k) {
  return (bottom || AVL_TREE_MAP.this.compare(k, from) >= 0) &&
   (top || AVL_TREE_MAP.this.compare(k, to) < 0);
 }

 @Override
 public ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> ENTRYSET() {
  if (entries == null) entries = new AbstractObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC>() {
    @Override
    public ObjectBidirectionalIterator<MAP.Entry KEY_VALUE_GENERIC> iterator() { return new
SubmapEntryIterator(); }
    @Override
    public ObjectBidirectionalIterator<MAP.Entry KEY_VALUE_GENERIC> iterator(final MAP.Entry
KEY_VALUE_GENERIC from) { return new SubmapEntryIterator(from.ENTRY_GET_KEY()); }
    @Override
    public Comparator<? super MAP.Entry KEY_VALUE_GENERIC> comparator() { return
AVL_TREE_MAP.this.ENTRYSET().comparator(); }
    @Override
    SUPPRESS_WARNINGS_KEY_UNCHECKED
    public boolean contains(final Object o) {
     if (!(o instanceof Map.Entry)) return false;
     final Map.Entry<?,?> e = (Map.Entry<?,?>)o;
#if KEYS_PRIMITIVE
     if (e.getKey() == null || ! (e.getKey() instanceof KEY_CLASS)) return false;
#endif
#if VALUES_PRIMITIVE
     if (e.getValue() == null || ! (e.getValue() instanceof VALUE_CLASS)) return false;
#endif
     final AVL_TREE_MAP.Entry KEY_VALUE_GENERIC f = findKey(KEY_OBJ2TYPE(KEY_GENERIC_CAST
e.getKey()));
     return f != null && in(f.key) && e.equals(f);
    }
    @Override
    SUPPRESS_WARNINGS_KEY_UNCHECKED
    public boolean remove(final Object o) {
     if (!(o instanceof Map.Entry)) return false;
```

```
    final Map.Entry<?,?> e = (Map.Entry<?,?>)o;
#if KEYS_PRIMITIVE
    if (e.getKey() == null || ! (e.getKey() instanceof KEY_CLASS)) return false;
#endif
#if VALUES_PRIMITIVE
    if (e.getValue() == null || ! (e.getValue() instanceof VALUE_CLASS)) return false;
#endif
    final AVL_TREE_MAP.Entry KEY_VALUE_GENERIC f = findKey(KEY_OBJ2TYPE(KEY_GENERIC_CAST e.getKey()));
    if (f != null && in(f.key)) Submap.this.REMOVE_VALUE(f.key);
    return f != null;
    }
    @Override
    public int size() {
     int c = 0;
     for(Iterator<?> i = iterator(); i.hasNext(); i.next()) c++;
     return c;
    }
    @Override
    public boolean isEmpty() { return ! new SubmapIterator().hasNext(); }
    @Override
    public void clear() { Submap.this.clear(); }
    @Override
    public MAP.Entry KEY_VALUE_GENERIC first() { return firstEntry(); }
    @Override
    public MAP.Entry KEY_VALUE_GENERIC last() { return lastEntry(); }
    @Override
    public ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> subSet(MAP.Entry KEY_VALUE_GENERIC from, MAP.Entry KEY_VALUE_GENERIC to) { return subMap(from.ENTRY_GET_KEY(), to.ENTRY_GET_KEY()).ENTRYSET(); }
    @Override
    public ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> headSet(MAP.Entry KEY_VALUE_GENERIC to) { return headMap(to.ENTRY_GET_KEY()).ENTRYSET(); }
    @Override
    public ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> tailSet(MAP.Entry KEY_VALUE_GENERIC from) { return tailMap(from.ENTRY_GET_KEY()).ENTRYSET(); }
   };

  return entries;
 }

 private class KeySet extends ABSTRACT_SORTED_MAP KEY_VALUE_GENERIC.KeySet {
  @Override
  public KEY_BIDI_ITERATOR KEY_GENERIC iterator() { return new SubmapKeyIterator(); }
  @Override
  public KEY_BIDI_ITERATOR KEY_GENERIC iterator(final KEY_GENERIC_TYPE from) { return new SubmapKeyIterator(from); }
 }
```

```
@Override
public SORTED_SET KEY_GENERIC keySet() {
 if (keys == null) keys = new KeySet();
 return keys;
}

@Override
public VALUE_COLLECTION VALUE_GENERIC values() {
 if (values == null) values = new VALUE_ABSTRACT_COLLECTION VALUE_GENERIC() {
  @Override
  public VALUE_ITERATOR VALUE_GENERIC iterator() { return new SubmapValueIterator(); }
  @Override
  public boolean contains(final VALUE_TYPE k) { return containsValue(k); }
  @Override
  public int size() { return Submap.this.size();}
  @Override
  public void clear() { Submap.this.clear(); }
 };

 return values;
}

@Override
SUPPRESS_WARNINGS_KEY_UNCHECKED
public boolean containsKey(final KEY_TYPE k) {
 return in(KEY_GENERIC_CAST k) && AVL_TREE_MAP.this.containsKey(k);
}

@Override
public boolean containsValue(final VALUE_TYPE v) {
 final SubmapIterator i = new SubmapIterator();
 VALUE_TYPE ev;

 while(i.hasNext()) {
  ev = i.nextEntry().value;
  if (VALUE_EQUALS(ev, v)) return true;
 }

 return false;
}

@Override
SUPPRESS_WARNINGS_KEY_UNCHECKED
public VALUE_GENERIC_TYPE GET_VALUE(final KEY_TYPE k) {
 final AVL_TREE_MAP.Entry KEY_VALUE_GENERIC e;
 final KEY_GENERIC_TYPE kk = KEY_GENERIC_CAST k;
 return in(kk) && (e = findKey(kk)) != null ? e.value : this.defRetValue;
```

```
 }

 @Override
 public VALUE_GENERIC_TYPE put(final KEY_GENERIC_TYPE k, final VALUE_GENERIC_TYPE v) {
  modified = false;
  if (! in(k)) throw new IllegalArgumentException("Key (" + k + ") out of range [" + (bottom ? "-" :
String.valueOf(from)) + ", " + (top ? "-" : String.valueOf(to)) + ")");
  final VALUE_GENERIC_TYPE oldValue = AVL_TREE_MAP.this.put(k, v);
  return modified ? this.defRetValue : oldValue;
 }

 @Override
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public VALUE_GENERIC_TYPE REMOVE_VALUE(final KEY_TYPE k) {
  modified = false;
  if (! in(KEY_GENERIC_CAST k)) return this.defRetValue;
  final VALUE_GENERIC_TYPE oldValue = AVL_TREE_MAP.this.REMOVE_VALUE(k);
  return modified ? oldValue : this.defRetValue;
 }

 @Override
 public int size() {
  final SubmapIterator i = new SubmapIterator();
  int n = 0;

  while(i.hasNext()) {
   n++;
   i.nextEntry();
  }

  return n;
 }

 @Override
 public boolean isEmpty() { return ! new SubmapIterator().hasNext(); }

 @Override
 public KEY_COMPARATOR KEY_SUPER_GENERIC comparator() { return actualComparator; }

 @Override
 public SORTED_MAP KEY_VALUE_GENERIC headMap(final KEY_GENERIC_TYPE to) {
  if (top) return new Submap(from, bottom, to, false);
  return compare(to, this.to) < 0 ? new Submap(from, bottom, to, false) : this;
 }

 @Override
 public SORTED_MAP KEY_VALUE_GENERIC tailMap(final KEY_GENERIC_TYPE from) {
  if (bottom) return new Submap(from, false, to, top);
```

```
 return compare(from, this.from) > 0 ? new Submap(from, false, to, top) : this;
 }

 @Override
 public SORTED_MAP KEY_VALUE_GENERIC subMap(KEY_GENERIC_TYPE from, KEY_GENERIC_TYPE
to) {
 if (top && bottom) return new Submap(from, false, to, false);
 if (! top) to = compare(to, this.to) < 0 ? to : this.to;
 if (! bottom) from = compare(from, this.from) > 0 ? from : this.from;
  if (! top && ! bottom && from == this.from && to == this.to) return this;
 return new Submap(from, false, to, false);
 }

 /** Locates the first entry.
  *
  * @return the first entry of this submap, or {@code null} if the submap is empty.
  */
 public AVL_TREE_MAP.Entry KEY_VALUE_GENERIC firstEntry() {
 if (tree == null) return null;
 // If this submap goes to -infinity, we return the main map first entry; otherwise, we locate the start of the map.
 AVL_TREE_MAP.Entry KEY_VALUE_GENERIC e;
 if (bottom) e = firstEntry;
 else {
  e = locateKey(from);
  // If we find either the start or something greater we're OK.
  if (compare(e.key, from) < 0) e = e.next();
 }
 // Finally, if this subset doesn't go to infinity, we check that the resulting key isn't greater than the end.
 if (e == null || ! top && compare(e.key, to) >= 0) return null;
 return e;
 }

 /** Locates the last entry.
  *
  * @return the last entry of this submap, or {@code null} if the submap is empty.
  */
 public AVL_TREE_MAP.Entry KEY_VALUE_GENERIC lastEntry() {
 if (tree == null) return null;
 // If this submap goes to infinity, we return the main map last entry; otherwise, we locate the end of the map.
 AVL_TREE_MAP.Entry KEY_VALUE_GENERIC e;
 if (top) e = lastEntry;
 else {
  e = locateKey(to);
  // If we find something smaller than the end we're OK.
  if (compare(e.key, to) >= 0) e = e.prev();
 }
 // Finally, if this subset doesn't go to -infinity, we check that the resulting key isn't smaller than the start.
 if (e == null || ! bottom && compare(e.key, from) < 0) return null;
```

```
 return e;
}

@Override
public KEY_GENERIC_TYPE FIRST_KEY() {
 AVL_TREE_MAP.Entry KEY_VALUE_GENERIC e = firstEntry();
 if (e == null) throw new NoSuchElementException();
 return e.key;
}

@Override
public KEY_GENERIC_TYPE LAST_KEY() {
 AVL_TREE_MAP.Entry KEY_VALUE_GENERIC e = lastEntry();
 if (e == null) throw new NoSuchElementException();
 return e.key;
}

/** An iterator for subranges.
 *
 * <p>This class inherits from {@link TreeIterator}, but overrides the methods that
 * update the pointer after a {@link java.util.ListIterator#next()} or {@link java.util.ListIterator#previous()}. If we
would
 * move out of the range of the submap we just overwrite the next or previous
 * entry with {@code null}.
 */
private class SubmapIterator extends TreeIterator {
 SubmapIterator() { next = firstEntry(); }
 SubmapIterator(final KEY_GENERIC_TYPE k) {
  this();

  if (next != null) {
   if (! bottom && compare(k, next.key) < 0) prev = null;
   else if (! top && compare(k, (prev = lastEntry()).key) >= 0) next = null;
   else {
    next = locateKey(k);

    if (compare(next.key, k) <= 0) {
     prev = next;
     next = next.next();
    }
    else prev = next.prev();
   }
  }
 }

 @Override
 void updatePrevious() {
  prev = prev.prev();
```

```
  if (! bottom && prev != null && AVL_TREE_MAP.this.compare(prev.key, from) < 0) prev = null;
  }

  @Override
  void updateNext() {
   next = next.next();
   if (! top && next != null && AVL_TREE_MAP.this.compare(next.key, to) >= 0) next = null;
  }
 }

 private class SubmapEntryIterator extends SubmapIterator implements ObjectListIterator<MAP.Entry
KEY_VALUE_GENERIC>  {
  SubmapEntryIterator() {}
  SubmapEntryIterator(final KEY_GENERIC_TYPE k) { super(k); }

  @Override
  public MAP.Entry KEY_VALUE_GENERIC next() { return nextEntry(); }
  @Override
  public MAP.Entry KEY_VALUE_GENERIC previous() { return previousEntry(); }
 }



 /** An iterator on a subrange of keys.
  *
  * <p>This class can iterate in both directions on a subrange of the
  * keys of a threaded tree. We simply override the {@link
  * java.util.ListIterator#next()}/{@link java.util.ListIterator#previous()} methods (and possibly their
  * type-specific counterparts) so that they return keys instead of
  * entries.
  */
 private final class SubmapKeyIterator extends SubmapIterator implements KEY_LIST_ITERATOR
KEY_GENERIC {
  public SubmapKeyIterator() { super(); }
  public SubmapKeyIterator(KEY_GENERIC_TYPE from) { super(from); }

  @Override
  public KEY_GENERIC_TYPE NEXT_KEY() { return nextEntry().key; }
  @Override
  public KEY_GENERIC_TYPE PREV_KEY() { return previousEntry().key; }
 };

 /** An iterator on a subrange of values.
  *
  * <p>This class can iterate in both directions on the values of a
  * subrange of the keys of a threaded tree. We simply override the
  * {@link java.util.ListIterator#next()}/{@link java.util.ListIterator#previous()} methods (and possibly their
  * type-specific counterparts) so that they return values instead of
  * entries.
```

```
  */
  private final class SubmapValueIterator extends SubmapIterator implements VALUE_LIST_ITERATOR
VALUE_GENERIC {
   @Override
   public VALUE_GENERIC_TYPE NEXT_VALUE() { return nextEntry().value; }
   @Override
   public VALUE_GENERIC_TYPE PREV_VALUE() { return previousEntry().value; }
  };
 }

 /** Returns a deep copy of this tree map.
  *
  * <p>This method performs a deep copy of this tree map; the data stored in the
  * set, however, is not cloned. Note that this makes a difference only for object keys.
  *
  * @return a deep copy of this tree map.
  */

 @Override
 SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
 public AVL_TREE_MAP KEY_VALUE_GENERIC clone() {
  AVL_TREE_MAP KEY_VALUE_GENERIC c;
  try {
   c = (AVL_TREE_MAP KEY_VALUE_GENERIC)super.clone();
  }
  catch(CloneNotSupportedException cantHappen) {
   throw new InternalError();
  }

  c.keys = null;
  c.values = null;
  c.entries = null;
  c.allocatePaths();

  if (count != 0) {
   // Also this apparently unfathomable code is derived from GNU libavl.
   Entry KEY_VALUE_GENERIC e, p, q, rp = new Entry KEY_VALUE_GENERIC_DIAMOND(), rq = new Entry
KEY_VALUE_GENERIC_DIAMOND();

   p = rp;
   rp.left(tree);

   q = rq;
   rq.pred(null);

   while(true) {
    if (! p.pred()) {
     e = p.left.clone();
```

```
    e.pred(q.left);
    e.succ(q);
    q.left(e);

    p = p.left;
    q = q.left;
   }
   else {
    while(p.succ()) {
     p = p.right;

     if (p == null) {
      q.right = null;
      c.tree = rq.left;

      c.firstEntry = c.tree;
      while(c.firstEntry.left != null) c.firstEntry = c.firstEntry.left;
      c.lastEntry = c.tree;
      while(c.lastEntry.right != null) c.lastEntry = c.lastEntry.right;

      return c;
     }
     q = q.right;
    }

    p = p.right;
    q = q.right;
   }

   if (! p.succ()) {
    e = p.right.clone();
    e.succ(q.right);
    e.pred(q);
    q.right(e);
   }
  }
 }

 return c;
}


private void writeObject(java.io.ObjectOutputStream s) throws java.io.IOException {
 int n = count;
 EntryIterator i = new EntryIterator();
 Entry KEY_VALUE_GENERIC e;

 s.defaultWriteObject();
```

```
 while(n-- != 0) {
  e = i.nextEntry();
  s.WRITE_KEY(e.key);
  s.WRITE_VALUE(e.value);
 }
}


 /** Reads the given number of entries from the input stream, returning the corresponding tree.
  *
  * @param s the input stream.
  * @param n the (positive) number of entries to read.
  * @param pred the entry containing the key that preceeds the first key in the tree.
  * @param succ the entry containing the key that follows the last key in the tree.
  */
 SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
 private Entry KEY_VALUE_GENERIC readTree(final java.io.ObjectInputStream s, final int n, final Entry
KEY_VALUE_GENERIC pred, final Entry KEY_VALUE_GENERIC succ) throws java.io.IOException,
ClassNotFoundException {
  if (n == 1) {
   final Entry KEY_VALUE_GENERIC top = new Entry
KEY_VALUE_GENERIC_DIAMOND(KEY_GENERIC_CAST s.READ_KEY(), VALUE_GENERIC_CAST
s.READ_VALUE());
   top.pred(pred);
   top.succ(succ);

   return top;
  }

  if (n == 2) {
   /* We handle separately this case so that recursion will
    *always* be on nonempty subtrees. */
   final Entry KEY_VALUE_GENERIC top = new Entry
KEY_VALUE_GENERIC_DIAMOND(KEY_GENERIC_CAST s.READ_KEY(), VALUE_GENERIC_CAST
s.READ_VALUE());
   top.right(new Entry KEY_VALUE_GENERIC_DIAMOND(KEY_GENERIC_CAST s.READ_KEY(),
VALUE_GENERIC_CAST s.READ_VALUE()));
   top.right.pred(top);
   top.balance(1);
   top.pred(pred);
   top.right.succ(succ);

   return top;
  }

  // The right subtree is the largest one.
  final int rightN = n / 2, leftN = n - rightN - 1;
```

```
 final Entry KEY_VALUE_GENERIC top = new Entry KEY_VALUE_GENERIC_DIAMOND();

 top.left(readTree(s, leftN, pred, top));

 top.key = KEY_GENERIC_CAST s.READ_KEY();
 top.value = VALUE_GENERIC_CAST s.READ_VALUE();

 top.right(readTree(s, rightN, top, succ));

 if (n == (n & -n)) top.balance(1); // Quick test for determining whether n is a power of 2.

 return top;
}



private void readObject(java.io.ObjectInputStream s) throws java.io.IOException, ClassNotFoundException {
 s.defaultReadObject();
 /* The storedComparator is now correctly set, but we must restore
    on-the-fly the actualComparator. */
 setActualComparator();
 allocatePaths();

 if (count != 0) {
  tree = readTree(s, count, null, null);
  Entry KEY_VALUE_GENERIC e;

  e = tree;
  while(e.left() != null) e = e.left();
  firstEntry = e;

  e = tree;
  while(e.right() != null) e = e.right();
  lastEntry = e;
 }
}


#ifdef ASSERTS_CODE
 private static KEY_VALUE_GENERIC int checkTree(Entry KEY_VALUE_GENERIC e) {
  if (e == null) return 0;

  final int leftN = checkTree(e.left()), rightN = checkTree(e.right());
  if (leftN + e.balance() != rightN)
   throw new AssertionError("Mismatch between left tree size (" + leftN + "), right tree size (" + rightN + ") and
balance (" + e.balance() + ")");
```

```
 return Math.max(leftN , rightN) + 1;
 }
#endif

#ifdef TEST

 private static long seed = System.currentTimeMillis();
 private static java.util.Random r = new java.util.Random(seed);

 private static KEY_TYPE genKey() {
#if KEY_CLASS_Byte || KEY_CLASS_Short || KEY_CLASS_Character
  return (KEY_TYPE)(r.nextInt());
#elif KEYS_PRIMITIVE
  return r.NEXT_KEY();
#else
  return Integer.toBinaryString(r.nextInt());
#endif
 }

 private static VALUE_TYPE genValue() {
#if VALUE_CLASS_Byte || VALUE_CLASS_Short || VALUE_CLASS_Character
  return (VALUE_TYPE)(r.nextInt());
#elif VALUES_PRIMITIVE
  return r.NEXT_VALUE();
#elif !VALUE_CLASS_Reference || KEY_CLASS_Reference
  return Integer.toBinaryString(r.nextInt());
#else
  return new java.io.Serializable() {};
#endif
 }


 private static java.text.NumberFormat format = new java.text.DecimalFormat("#,###.00");
 private static java.text.FieldPosition p = new java.text.FieldPosition(0);

 private static String format(double d) {
  StringBuffer s = new StringBuffer();
  return format.format(d, s, p).toString();
 }

 private static void speedTest(int n, boolean comp) {
  int i, j;
  AVL_TREE_MAP m;
  java.util.TreeMap t;
  KEY_TYPE k[] = new KEY_TYPE[n];
  KEY_TYPE nk[] = new KEY_TYPE[n];
  VALUE_TYPE v[] = new VALUE_TYPE[n];
  long ms;
```

```
for(i = 0; i < n; i++) {
 k[i] = genKey();
 nk[i] = genKey();
 v[i] = genValue();
}

double totPut = 0, totYes = 0, totNo = 0, totAddTo = 0, totIterFor = 0, totIterBack = 0, totRemYes = 0, d, dd, ddd;

if (comp) { for(j = 0; j < 20; j++) {

 t = new java.util.TreeMap();

 /* We first add all pairs to t. */
 for(i = 0; i < n;  i++) t.put(KEY2OBJ(k[i]), VALUE2OBJ(v[i]));

 /* Then we remove the first half and put it back. */
 for(i = 0; i < n/2;  i++) t.remove(KEY2OBJ(k[i]));

 ms = System.currentTimeMillis();
 for(i = 0; i < n/2;  i++) t.put(KEY2OBJ(k[i]), VALUE2OBJ(v[i]));
 d = System.currentTimeMillis() - ms;

 /* Then we remove the other half and put it back again. */
 ms = System.currentTimeMillis();
 for(i = n/2; i < n;  i++) t.remove(KEY2OBJ(k[i]));
 dd = System.currentTimeMillis() - ms ;

 ms = System.currentTimeMillis();
 for(i = n/2; i < n;  i++) t.put(KEY2OBJ(k[i]), VALUE2OBJ(v[i]));
 d += System.currentTimeMillis() - ms;
 if (j > 2) totPut += n/d;
 System.out.print("Add: " + format(n/d) +" K/s ");

 /* Then we remove again the first half. */
 ms = System.currentTimeMillis();
 for(i = 0; i < n/2;  i++) t.remove(KEY2OBJ(k[i]));
 dd += System.currentTimeMillis() - ms ;
 if (j > 2) totRemYes += n/dd;
 System.out.print("RemYes: " + format(n/dd) +" K/s ");

 /* And then we put it back. */
 for(i = 0; i < n/2;  i++) t.put(KEY2OBJ(k[i]), VALUE2OBJ(v[i]));

 #if VALUES_PRIMITIVE && !VALUE_CLASS_Boolean
 /* we perform n/2 addTo() operations with get then put */
 ms = System.currentTimeMillis();
 for(i = 0; i < n/2; i++) t.put(KEY2OBJ(k[i]), (VALUE_TYPE) ((VALUE_CLASS) t.get(KEY2OBJ(k[i])) + i));
```

```
  ddd = System.currentTimeMillis() - ms;
  if (j > 2) totAddTo += n/ddd;
  System.out.print("AddTo: " + format(n/ddd) +" K/s ");
  #endif

  /* We check for pairs in t. */
  ms = System.currentTimeMillis();
  for(i = 0; i < n;  i++) t.containsKey(KEY2OBJ(k[i]));
  d = 1.0 * n / (System.currentTimeMillis() - ms);
  if (j > 2) totYes += d;
  System.out.print("Yes: " + format(d) +" K/s ");

  /* We check for pairs not in t. */
  ms = System.currentTimeMillis();
  for(i = 0; i < n;  i++) t.containsKey(KEY2OBJ(nk[i]));
  d = 1.0 * n / (System.currentTimeMillis() - ms);
  if (j > 2) totNo += d;
  System.out.print("No: " + format(d) +" K/s ");

  /* We iterate on t. */
  ms = System.currentTimeMillis();
  for(Iterator it = t.entrySet().iterator(); it.hasNext(); it.next());
  d = 1.0 * n / (System.currentTimeMillis() - ms);
  if (j > 2) totIterFor += d;
  System.out.print("IterFor: " + format(d) +" K/s ");

  System.out.println();
  }

  System.out.println();
  System.out.println("java.util Put: " + format(totPut/(j-3)) + " K/s RemYes: " + format(totRemYes/(j-3)) + " K/s Yes:
" + format(totYes/(j-3)) + " K/s No: " + format(totNo/(j-3))+ "K/s AddTo: " + format(totAddTo/(j-3)) + " K/s
IterFor: " + format(totIterFor/(j-3))  + " K/s");

  System.out.println();

  t = null;
  totPut = totYes = totNo = totIterFor = totIterBack = totRemYes = totAddTo = 0;

  }

  for(j = 0; j < 20; j++) {

  m = new AVL_TREE_MAP();

  /* We first add all pairs to m. */
  for(i = 0; i < n;  i++) m.put(k[i], v[i]);
```

```
/* Then we remove the first half and put it back. */
for(i = 0; i < n/2;  i++) m.remove(k[i]);

ms = System.currentTimeMillis();
for(i = 0; i < n/2;  i++) m.put(k[i], v[i]);
d = System.currentTimeMillis() - ms;

/* Then we remove the other half and put it back again. */
ms = System.currentTimeMillis();
for(i = n/2; i < n;  i++) m.remove(k[i]);
dd = System.currentTimeMillis() - ms ;

ms = System.currentTimeMillis();
for(i = n/2; i < n;  i++) m.put(k[i], v[i]);
d += System.currentTimeMillis() - ms;
if (j > 2) totPut += n/d;
System.out.print("Add: " + format(n/d) +" K/s ");

/* Then we remove again the first half. */
ms = System.currentTimeMillis();
for(i = 0; i < n/2;  i++) m.remove(k[i]);
dd += System.currentTimeMillis() - ms ;
if (j > 2) totRemYes += n/dd;
System.out.print("RemYes: " + format(n/dd) +" K/s ");

/* And then we put it back. */
for(i = 0; i < n/2;  i++) m.put(k[i], v[i]);

#if VALUES_PRIMITIVE && !VALUE_CLASS_Boolean
/* we perform n/2 addTo() operations with get then put */
ms = System.currentTimeMillis();
for(i = 0; i < n/2; i++) m.addTo(k[i], (VALUE_TYPE) i);
ddd = System.currentTimeMillis() - ms;
if (j > 2) totAddTo += n/ddd;
System.out.print("AddTo: " + format(n/ddd) +" K/s ");
#endif

/* We check for pairs in m. */
ms = System.currentTimeMillis();
for(i = 0; i < n;  i++) m.containsKey(k[i]);
d = 1.0 * n / (System.currentTimeMillis() - ms);
if (j > 2) totYes += d;
System.out.print("Yes: " + format(d) +" K/s ");

/* We check for pairs not in m. */
ms = System.currentTimeMillis();
for(i = 0; i < n;  i++) m.containsKey(nk[i]);
d = 1.0 * n / (System.currentTimeMillis() - ms);
```

```
  if (j > 2) totNo += d;
  System.out.print("No: " + format(d) +" K/s ");


  /* We iterate on m. */
  java.util.ListIterator it = (java.util.ListIterator)m.entrySet().iterator();
  ms = System.currentTimeMillis();
  for(it = (java.util.ListIterator)m.entrySet().iterator(); it.hasNext(); it.next());
  d = 1.0 * n / (System.currentTimeMillis() - ms);
  if (j > 2) totIterFor += d;
  System.out.print("IterFor: " + format(d) +" K/s ");

  /* We iterate back on m. */
  ms = System.currentTimeMillis();
  for(; it.hasPrevious(); it.previous());
  d = 1.0 * n / (System.currentTimeMillis() - ms);
  if (j > 2) totIterBack += d;
  System.out.print("IterBack: " + format(d) +" K/s ");

  System.out.println();
 }


 System.out.println();
 System.out.println("fastutil Put: " + format(totPut/(j-3)) + " K/s RemYes: " + format(totRemYes/(j-3)) + " K/s Yes:
" + format(totYes/(j-3)) + " K/s No: " + format(totNo/(j-3))+ "K/s AddTo: " + format(totAddTo/(j-3)) + " K/s
IterFor: " + format(totIterFor/(j-3))  + " K/s");

 System.out.println();

 }


 private static boolean valEquals(Object o1, Object o2) {
  return o1 == null ? o2 == null : o1.equals(o2);
 }

 private static void fatal(String msg) {
  System.out.println(msg);
  System.exit(1);
 }

 private static void ensure(boolean cond, String msg) {
  if (cond) return;
  fatal(msg);
 }

 private static void compareMT(SORTED_MAP m, SortedMap t, int level, long seed) {
```

```
  /* Now we check that m and t are equal. */
  if (!m.equals(t) || ! t.equals(m)) System.err.println("m: " + m + " t: " + t);

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) at start");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) at start");



  /* Now we check that m actually holds that data. */
  for(Iterator i=t.entrySet().iterator(); i.hasNext();) {
   java.util.Map.Entry e = (java.util.Map.Entry)i.next();
   ensure(valEquals(e.getValue(), m.get(e.getKey())), "Error (" + level + ", " + seed + "): m and t differ on an entry
("+e+") after insertion (iterating on t)");
  }

  /* Now we check that m actually holds that data, but iterating on m. */
  for(Iterator i=m.entrySet().iterator(); i.hasNext();) {
   Entry e = (Entry)i.next();
   ensure(valEquals(e.getValue(), t.get(e.getKey())), "Error (" + level + ", " + seed + "): m and t differ on an entry
("+e+") after insertion (iterating on m)");
  }

  /* Now we check that m actually holds the same keys. */
  for(Iterator i=t.keySet().iterator(); i.hasNext();) {
   Object o = i.next();
   ensure(m.containsKey(o), "Error (" + level + ", " + seed + "): m and t differ on a key ("+o+") after insertion
(iterating on t)");
   ensure(m.keySet().contains(o), "Error (" + level + ", " + seed + "): m and t differ on a key ("+o+", in keySet()) after
insertion (iterating on t)");
  }

  /* Now we check that m actually holds the same keys, but iterating on m. */
  for(Iterator i=m.keySet().iterator(); i.hasNext();) {
   Object o = i.next();
   ensure(t.containsKey(o), "Error (" + level + ", " + seed + "): m and t differ on a key after insertion (iterating on m)");
   ensure(t.keySet().contains(o), "Error (" + level + ", " + seed + "): m and t differ on a key (in keySet()) after insertion
(iterating on m)");
  }



  /* Now we check that m actually hold the same values. */
  for(Iterator i=t.values().iterator(); i.hasNext();) {
   Object o = i.next();
   ensure(m.containsValue(o), "Error (" + level + ", " + seed + "): m and t differ on a value after insertion (iterating on
t)");
   ensure(m.values().contains(o), "Error (" + level + ", " + seed + "): m and t differ on a value (in values()) after
insertion (iterating on t)");
  }
```

```
  /* Now we check that m actually hold the same values, but iterating on m. */
  for(Iterator i=m.values().iterator(); i.hasNext();) {
   Object o = i.next();
   ensure(t.containsValue(o), "Error (" + level + ", " + seed + "): m and t differ on a value after insertion (iterating on
m)");
   ensure(t.values().contains(o), "Error (" + level + ", " + seed + "): m and t differ on a value (in values()) after
insertion (iterating on m)");
  }

 }

 private static Object[] k, v, nk;
 private static KEY_TYPE kt[];
 private static KEY_TYPE nkt[];
 private static VALUE_TYPE vt[];
 private static AVL_TREE_MAP topMap;

 protected static void testMaps(SORTED_MAP m, SortedMap t, int n, int level) {
  long ms;
  boolean mThrowsIllegal, tThrowsIllegal, mThrowsNoElement, tThrowsNoElement;
  Object rt = null, rm = null;

  if (level > 4) return;


  /* Now we check that both maps agree on first/last keys. */

  mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

  try {
   m.firstKey();
  }
  catch (NoSuchElementException e) { mThrowsNoElement = true; }
  try {
   t.firstKey();
  }
  catch (NoSuchElementException e) { tThrowsNoElement = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): firstKey() divergence at
start in NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
  if (! mThrowsNoElement) ensure(t.firstKey().equals(m.firstKey()), "Error (" + level + ", " + seed + "): m and t differ
at start on their first key (" + m.firstKey() + ", " + t.firstKey() +")");

  mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

  try {
   m.lastKey();
```

```
  }
  catch (NoSuchElementException e) { mThrowsNoElement = true; }
  try {
   t.lastKey();
  }
  catch (NoSuchElementException e) { tThrowsNoElement = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): lastKey() divergence at start
in NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");


  if (! mThrowsNoElement) ensure(t.lastKey().equals(m.lastKey()), "Error (" + level + ", " + seed + "): m and t differ
at start on their last key (" + m.lastKey() + ", " + t.lastKey() +")");

  compareMT(m, t, level, seed);

  /* Now we check that inquiries about random data give the same answer in m and t. For
     m we use the polymorphic method. */

  for(int i=0; i<n;  i++) {
   KEY_TYPE T = genKey();

   mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

   try {
    m.containsKey(KEY2OBJ(T));
   }
   catch (NoSuchElementException e) { mThrowsNoElement = true; }
   catch (IllegalArgumentException e) { mThrowsIllegal = true; }

   try {
    t.containsKey(KEY2OBJ(T));
   }
   catch (NoSuchElementException e) { tThrowsNoElement = true; }
   catch (IllegalArgumentException e) { tThrowsIllegal = true; }

   ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): containsKey() divergence in
NoSuchElementException (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
   ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): containsKey() divergence in
IllegalArgumentException (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
   if (!mThrowsNoElement && !mThrowsIllegal) {
    ensure(m.containsKey(KEY2OBJ(T)) == t.containsKey(KEY2OBJ(T)), "Error (" + level + ", " + seed + "):
divergence in keys between t and m (polymorphic method)");

#if KEY_CLASS_Object && ! (VALUES_REFERENCE)
    if ((m.GET_VALUE(T) != VALUE_NULL) != ((t.get(KEY2OBJ(T)) == null ? VALUE_NULL :
VALUE_OBJ2TYPE(t.get(KEY2OBJ(T)))) != VALUE_NULL) ||
     t.get(KEY2OBJ(T)) != null &&
```

```
    ! VALUE2OBJ(m.GET_VALUE(T)).equals(t.get(KEY2OBJ(T))))
#else
    if ((m.get(T) != VALUE_NULL) != ((t.get(KEY2OBJ(T)) == null ? VALUE_NULL :
VALUE_OBJ2TYPE(t.get(KEY2OBJ(T)))) != VALUE_NULL) ||
     t.get(KEY2OBJ(T)) != null &&
     ! m.get(KEY2OBJ(T)).equals(t.get(KEY2OBJ(T))))
#endif
     {
      System.out.println("Error (" + level + ", " + seed + "): divergence between t and m (polymorphic method)");
      System.exit(1);
     }
   }
  }

  /* Again, we check that inquiries about random data give the same answer in m and t, but
    for m we use the standard method. */

  for(int i=0; i<n;  i++) {
   KEY_TYPE T = genKey();
   mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

   try {
    m.get(KEY2OBJ(T));
   }
   catch (NoSuchElementException e) { mThrowsNoElement = true; }
   catch (IllegalArgumentException e) { mThrowsIllegal = true; }

   try {
    t.get(KEY2OBJ(T));
   }
   catch (NoSuchElementException e) { tThrowsNoElement = true; }
   catch (IllegalArgumentException e) { tThrowsIllegal = true; }

   ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): get() divergence in
NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
   ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): get() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
   if (!mThrowsNoElement && !mThrowsIllegal) ensure(valEquals(m.get(KEY2OBJ(T)), t.get(KEY2OBJ(T))),
"Error (" + level + ", " + seed + "): divergence between t and m (standard method)");
  }

  /* Now we put and remove random data in m and t, checking that the result is the same. */

  for(int i=0; i<20*n;  i++) {
   KEY_TYPE T = genKey();
   VALUE_TYPE U = genValue();

   mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;
```

```
  try {
   rm = m.put(KEY2OBJ(T), VALUE2OBJ(U));
  }
  catch (NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }

  try {
   rt = t.put(KEY2OBJ(T), VALUE2OBJ(U));
  }
  catch (NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }


  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): put() divergence in
NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): put() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
  if (!mThrowsNoElement && !mThrowsIllegal) ensure(valEquals(rm, rt), "Error (" + level + ", " + seed + "):
divergence in put() between t and m (" + rt + ", " + rm + ")");

  T = genKey();


  mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

  try {
   rm = m.remove(KEY2OBJ(T));
  }
  catch (NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }

  try {
   rt = t.remove(KEY2OBJ(T));
  }
  catch (NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }


  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): remove() divergence in
NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): remove() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
  if (!mThrowsNoElement && !mThrowsIllegal) ensure(valEquals(rm, rt), "Error (" + level + ", " + seed + "):
divergence in remove() between t and m (" + rt + ", " + rm + ")");
  }

 compareMT(m, t, level, seed);
```

```
#if VALUES_PRIMITIVE && !VALUE_CLASS_Boolean
/* Now we check that addTo results in the same values as get/put */
if (m instanceof AVL_TREE_MAP) {
 for (Iterator i=m.entrySet().iterator(); i.hasNext();) {
  Entry e = (Entry)i.next();
  VALUE_TYPE incr = genValue();
  ((AVL_TREE_MAP) m).addTo(e.key, incr);
  t.put(e.key, (VALUE_TYPE) ((VALUE_CLASS) t.get(e.key) + incr));
 }
 compareMT(m, t, level, seed);

 /* Now we make sure that addTo works with a key that did not previously exist (with special default return value)*/
 KEY_TYPE newKey;

 do {
  newKey = genKey();
 } while (m.containsKey(newKey));

 VALUE_TYPE newValue, drv;
 drv = m.defaultReturnValue();
 newValue = genValue();
 m.defaultReturnValue(genValue());

 ((AVL_TREE_MAP) m).addTo(newKey, newValue);
 t.put(newKey, (VALUE_TYPE) (newValue + m.defaultReturnValue()));

 compareMT(m, t, level, seed);
 m.defaultReturnValue(drv); // set the default value back to the old drv
}
#endif

/* Now we check that both maps agree on first/last keys. */

mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

try {
 m.firstKey();
}
catch (NoSuchElementException e) { mThrowsNoElement = true; }
try {
 t.firstKey();
}
catch (NoSuchElementException e) { tThrowsNoElement = true; }

ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): firstKey() divergence in
NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
if (! mThrowsNoElement) ensure(t.firstKey().equals(m.firstKey()), "Error (" + level + ", " + seed + "): m and t differ
```

```
on their first key (" + m.firstKey() + ", " + t.firstKey() +")");

  mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

  try {
   m.lastKey();
  }
  catch (NoSuchElementException e) { mThrowsNoElement = true; }
  try {
   t.lastKey();
  }
  catch (NoSuchElementException e) { tThrowsNoElement = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): lastKey() divergence in
NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");

  if (! mThrowsNoElement) ensure(t.lastKey().equals(m.lastKey()), "Error (" + level + ", " + seed + "): m and t differ
on their last key (" + m.lastKey() + ", " + t.lastKey() +")");

  /* Now we check cloning. */

  if (level == 0) {
   ensure(m.equals(((AVL_TREE_MAP)m).clone()), "Error (" + level + ", " + seed + "): m does not equal m.clone()");
   ensure(((AVL_TREE_MAP)m).clone().equals(m), "Error (" + level + ", " + seed + "): m.clone() does not equal m");
  }

  int h = m.hashCode();


  /* Now we save and read m. */

  SORTED_MAP m2 = null;

  try {
   java.io.File ff = new java.io.File("it.unimi.dsi.fastutil.test");
   java.io.OutputStream os = new java.io.FileOutputStream(ff);
   java.io.ObjectOutputStream oos = new java.io.ObjectOutputStream(os);

   oos.writeObject(m);
   oos.close();

   java.io.InputStream is = new java.io.FileInputStream(ff);
   java.io.ObjectInputStream ois = new java.io.ObjectInputStream(is);

   m2 = (SORTED_MAP)ois.readObject();
   ois.close();
   ff.delete();
  }
```

```
  catch(Exception e) {
   e.printStackTrace();
   System.exit(1);
  }

#if !VALUE_CLASS_Reference
  ensure(m2.hashCode() == h, "Error (" + level + ", " + seed + "): hashCode() changed after save/read");

  /* Now we check that m2 actually holds that data. */

  ensure(m2.equals(t), "Error (" + level + ", " + seed + "): ! m2.equals(t) after save/read");
  ensure(t.equals(m2), "Error (" + level + ", " + seed + "): ! t.equals(m2) after save/read");
#else
  m2.clear();
  m2.putAll(m);
#endif

  /* Now we take out of m everything, and check that it is empty. */

  for(Iterator i=t.keySet().iterator(); i.hasNext();) m2.remove(i.next());

  ensure(m2.isEmpty(), "Error (" + level + ", " + seed + "): m2 is not empty (as it should be)");

  /* Now we play with iterators. */

  {
   java.util.ListIterator i, j;
   Map.Entry E, F;
   Object J;
   i = (java.util.ListIterator)m.entrySet().iterator();
   j = new java.util.LinkedList(t.entrySet()).listIterator();

   for(int k = 0; k < 2*n; k++) {
    ensure(i.hasNext() == j.hasNext(), "Error (" + level + ", " + seed + "): divergence in hasNext()");
    ensure(i.hasPrevious() == j.hasPrevious(), "Error (" + level + ", " + seed + "): divergence in hasPrevious()");

    if (r.nextFloat() < .8 && i.hasNext()) {
     ensure((E=(Entry)i.next()).getKey().equals(J = (F=(Map.Entry)j.next()).getKey()), "Error (" + level + ", " + seed +
"): divergence in next()");

     if (r.nextFloat() < 0.3) {
      i.remove();
      j.remove();
      t.remove(J);
     }
     else if (r.nextFloat() < 0.3) {
      Object U = VALUE2OBJ(genValue());
      E.setValue(U);
```

```
                t.put(F.getKey(), U);
     }
     }
     else if (r.nextFloat() < .2 && i.hasPrevious()) {
      ensure((E=(Entry)i.previous()).getKey().equals(J = (F=(Map.Entry)j.previous()).getKey()), "Error (" + level + ", " +
seed + "): divergence in previous()");

      if (r.nextFloat() < 0.3) {
       i.remove();
       j.remove();
       t.remove(J);
      }
      else if (r.nextFloat() < 0.3) {
       Object U = VALUE2OBJ(genValue());
       E.setValue(U);
                t.put(F.getKey(), U);
     }
     }

     ensure(i.nextIndex() == j.nextIndex(), "Error (" + level + ", " + seed + "): divergence in nextIndex()");
     ensure(i.previousIndex() == j.previousIndex(), "Error (" + level + ", " + seed + "): divergence in previousIndex()");

    }

   }


   {
    boolean badPrevious = false;
    Object previous = null;
    it.unimi.dsi.fastutil.BidirectionalIterator i;
    java.util.ListIterator j;
    Object I, J;
    KEY_TYPE from = genKey();
    j = new java.util.LinkedList(t.keySet()).listIterator();
    while(j.hasNext()) {
     Object k = j.next();
     if (((Comparable)k).compareTo(KEY2OBJ(from)) > 0) {
      badPrevious = true;
      j.previous();
      break;
     }
     previous = k;
    }

    i = (it.unimi.dsi.fastutil.BidirectionalIterator)((SORTED_SET)m.keySet()).iterator(from);

    for(int k = 0; k < 2*n; k++) {
```

```
    ensure(i.hasNext() == j.hasNext(), "Error (" + level + ", " + seed + "): divergence in hasNext() (iterator with starting point " + from + ")");
    ensure(i.hasPrevious() == j.hasPrevious() || badPrevious && (i.hasPrevious() == (previous != null)), "Error (" + level + ", " + seed + "): divergence in hasPrevious() (iterator with starting point " + from + ")" + badPrevious);

    if (r.nextFloat() < .8 && i.hasNext()) {
     ensure((I = i.next()).equals(J = j.next()), "Error (" + level + ", " + seed + "): divergence in next() (" + I + ", " + J + ", iterator with starting point " + from + ")");
     //System.err.println("Done next " + I + " " + J + "  " + badPrevious);

     badPrevious = false;

     if (r.nextFloat() < 0.5) {
      //System.err.println("Removing in next");
      i.remove();
      j.remove();
      t.remove(J);
     }
    }
    else if (!badPrevious && r.nextFloat() < .2 && i.hasPrevious()) {
     ensure((I = i.previous()).equals(J = j.previous()), "Error (" + level + ", " + seed + "): divergence in previous() (" + I + ", " + J + ", iterator with starting point " + from + ")");

     if (r.nextFloat() < 0.5) {
      //System.err.println("Removing in prev");
      i.remove();
      j.remove();
      t.remove(J);
     }
    }
   }

  }

  /* Now we check that m actually holds that data. */

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after iteration");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after iteration");

  /* Now we select a pair of keys and create a submap. */

  if (! m.isEmpty()) {
   java.util.ListIterator i;
   Object start = m.firstKey(), end = m.firstKey();
   for(i = (java.util.ListIterator)m.keySet().iterator(); i.hasNext() && r.nextFloat() < .3; start = end = i.next());
   for(; i.hasNext() && r.nextFloat() < .95; end = i.next());

   //System.err.println("Checking subMap from " + start + " to " + end + " (level=" + (level+1) + ")...");
```

```
  testMaps((SORTED_MAP)m.subMap((KEY_CLASS)start, (KEY_CLASS)end), t.subMap(start, end), n, level + 1);

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after subMap");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after subMap");

  //System.err.println("Checking headMap to " + end + " (level=" + (level+1) + ")...");
  testMaps((SORTED_MAP)m.headMap((KEY_CLASS)end), t.headMap(end), n, level + 1);

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after headMap");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after headMap");

  //System.err.println("Checking tailMap from " + start + " (level=" + (level+1) + ")...");
  testMaps((SORTED_MAP)m.tailMap((KEY_CLASS)start), t.tailMap(start), n, level + 1);

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after tailMap");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after tailMap");
  }


 }


 private static void runTest(int n) {
  AVL_TREE_MAP m = new AVL_TREE_MAP();
  SortedMap t = new java.util.TreeMap();
  topMap = m;
  k = new Object[n];
  v = new Object[n];
  nk = new Object[n];
  kt = new KEY_TYPE[n];
  nkt = new KEY_TYPE[n];
  vt = new VALUE_TYPE[n];

  for(int i = 0; i < n; i++) {
#if KEY_CLASS_Object
   k[i] = kt[i] = genKey();
   nk[i] = nkt[i] = genKey();
#else
   k[i] = new KEY_CLASS(kt[i] = genKey());
   nk[i] = new KEY_CLASS(nkt[i] = genKey());
#endif
#if VALUES_REFERENCE
   v[i] = vt[i] = genValue();
#else
   v[i] = new VALUE_CLASS(vt[i] = genValue());
#endif
  }
```

```
 /* We add pairs to t. */
 for(int i = 0; i < n;  i++) t.put(k[i], v[i]);

 /* We add to m the same data */
 m.putAll(t);

 testMaps(m, t, n, 0);

 System.out.println("Test OK");
 return;
}


public static void main(String args[]) {
 int n  = Integer.parseInt(args[1]);
 if (args.length > 2) r = new java.util.Random(seed = Long.parseLong(args[2]));

 try {
  if ("speedTest".equals(args[0]) || "speedComp".equals(args[0])) speedTest(n, "speedComp".equals(args[0]));
  else if ("test".equals(args[0])) runTest(n);
 } catch(Throwable e) {
  e.printStackTrace(System.err);
  System.err.println("seed: " + seed);
 }
}

#endif


}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/AVLTreeMap.drv
No license file was found, but licenses were detected in source scan.

distributed under the <a HREF="http://www.apache.org/licenses/liCENSE-2.0.html">Apache License 2.0.

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/overview.html
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2004-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
```

```
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/



#if KEY_CLASS_Byte

// HORRIBLE kluges to work around bug #6478546

private static final int MAX_IO_LENGTH = 1024 * 1024;

private static int read(final InputStream is, final byte a[], final int offset, final int length) throws IOException {
 if (length == 0) return 0;

 int read = 0, result;
 do {
  result = is.read(a, offset + read, Math.min(length - read, MAX_IO_LENGTH));
  if (result < 0) return read;
  read += result;
 } while(read < length);

 return read;
}

private static void write(final OutputStream outputStream, final byte a[], final int offset, final int length) throws
IOException {
 int written = 0;
 while(written < length) {
  outputStream.write(a, offset + written, Math.min(length - written, MAX_IO_LENGTH));
  written += Math.min(length - written, MAX_IO_LENGTH);
 }
}

private static void write(final DataOutput dataOutput, final byte a[], final int offset, final int length) throws
IOException {
 int written = 0;
 while(written < length) {
  dataOutput.write(a, offset + written, Math.min(length - written, MAX_IO_LENGTH));
  written += Math.min(length - written, MAX_IO_LENGTH);
 }
```

```
}

// Additional read/write methods to work around the DataInput/DataOutput schizophrenia.

/** Loads bytes from a given input stream, storing them in a given array fragment.
*
* <p>Note that this method is going to be significantly faster than {@link #loadBytes(DataInput,byte[],int,int)}
* as it uses {@link InputStream}'s bulk-read methods.
*
* @param inputStream an input stream.
* @param array an array which will be filled with data from {@code inputStream}.
* @param offset the index of the first element of {@code array} to be filled.
* @param length the number of elements of {@code array} to be filled.
* @return the number of elements actually read from {@code inputStream} (it might be less than {@code length} if
{@code inputStream} ends).
*/
public static int LOAD_KEYS(final InputStream inputStream, final KEY_TYPE[] array, final int offset, final int
length) throws IOException {
 return read(inputStream, array, offset, length);
}

/** Loads bytes from a given input stream, storing them in a given array.
*
* <p>Note that this method is going to be significantly faster than {@link #loadBytes(DataInput,byte[])}
* as it uses {@link InputStream}'s bulk-read methods.
*
* @param inputStream an input stream.
* @param array an array which will be filled with data from {@code inputStream}.
* @return the number of elements actually read from {@code inputStream} (it might be less than the array length if
{@code inputStream} ends).
*/
public static int LOAD_KEYS(final InputStream inputStream, final KEY_TYPE[] array) throws IOException {
 return read(inputStream, array, 0, array.length);
}

/** Stores an array fragment to a given output stream.
*
* <p>Note that this method is going to be significantly faster than {@link #storeBytes(byte[],int,int,DataOutput)}
* as it uses {@link OutputStream}'s bulk-read methods.
*
* @param array an array whose elements will be written to {@code outputStream}.
* @param offset the index of the first element of {@code array} to be written.
* @param length the number of elements of {@code array} to be written.
* @param outputStream an output stream.
*/
public static void STORE_KEYS(final KEY_TYPE array[], final int offset, final int length, final OutputStream
outputStream) throws IOException {
 write(outputStream, array, offset, length);
```

```
}

/** Stores an array to a given output stream.
*
* <p>Note that this method is going to be significantly faster than {@link #storeBytes(byte[],DataOutput)}
* as it uses {@link OutputStream}'s bulk-read methods.
*
* @param array an array whose elements will be written to {@code outputStream}.
* @param outputStream an output stream.
*/
public static void STORE_KEYS(final KEY_TYPE array[], final OutputStream outputStream) throws IOException
{
 write(outputStream, array, 0, array.length);
}


private static long read(final InputStream is, final byte a[][], final long offset, final long length) throws IOException
{
 if (length == 0) return 0;

 long read = 0;
 int segment = segment(offset);
 int displacement = displacement(offset);
 int result;
 do {
  result = is.read(a[segment], displacement, (int)Math.min(a[segment].length - displacement, Math.min(length - read,
MAX_IO_LENGTH)));
  if (result < 0) return read;
  read += result;
  displacement += result;
  if (displacement == a[segment].length) {
   segment++;
   displacement = 0;
  }
 } while(read < length);

 return read;
}

private static void write(final OutputStream outputStream, final byte a[][], final long offset, final long length) throws
IOException {
 if (length == 0) return;
 long written = 0;
 int toWrite;
 int segment = segment(offset);
 int displacement = displacement(offset);
 do {
  toWrite = (int)Math.min(a[segment].length - displacement, Math.min(length - written, MAX_IO_LENGTH));
```

```
  outputStream.write(a[segment], displacement, toWrite);
  written += toWrite;
  displacement += toWrite;
  if (displacement == a[segment].length) {
   segment++;
   displacement = 0;
  }
 } while(written < length);
}

private static void write(final DataOutput dataOutput, final byte a[][], final long offset, final long length) throws
IOException {
 if (length == 0) return;
 long written = 0;
 int toWrite;
 int segment = segment(offset);
 int displacement = displacement(offset);
 do {
  toWrite = (int)Math.min(a[segment].length - displacement, Math.min(length - written, MAX_IO_LENGTH));
  dataOutput.write(a[segment], displacement, toWrite);
  written += toWrite;
  displacement += toWrite;
  if (displacement == a[segment].length) {
   segment++;
   displacement = 0;
  }
 } while(written < length);
}

// Additional read/write methods to work around the DataInput/DataOutput schizophrenia.


/** Loads bytes from a given input stream, storing them in a given big-array fragment.
*
* <p>Note that this method is going to be significantly faster than {@link
#loadBytes(DataInput,byte[][],long,long)}
* as it uses {@link InputStream}'s bulk-read methods.
*
* @param inputStream an input stream.
* @param array a big array which will be filled with data from {@code inputStream}.
* @param offset the index of the first element of {@code array} to be filled.
* @param length the number of elements of {@code array} to be filled.
* @return the number of elements actually read from {@code inputStream} (it might be less than {@code length} if
{@code inputStream} ends).
*/
public static long LOAD_KEYS(final InputStream inputStream, final KEY_TYPE[][] array, final long offset, final
long length) throws IOException {
 return read(inputStream, array, offset, length);
```

```
}

/** Loads bytes from a given input stream, storing them in a given big array.
*
* <p>Note that this method is going to be significantly faster than {@link #loadBytes(DataInput,byte[][])}
* as it uses {@link InputStream}'s bulk-read methods.
*
* @param inputStream an input stream.
* @param array a big array which will be filled with data from {@code inputStream}.
* @return the number of elements actually read from {@code inputStream} (it might be less than the array length if
{@code inputStream} ends).
*/
public static long LOAD_KEYS(final InputStream inputStream, final KEY_TYPE[][] array) throws IOException {
 return read(inputStream, array, 0, BIG_ARRAYS.length(array));
}

/** Stores a big-array fragment to a given output stream.
*
* <p>Note that this method is going to be significantly faster than {@link
#storeBytes(byte[][],long,long,DataOutput)}
* as it uses {@link OutputStream}'s bulk-read methods.
*
* @param array a big array whose elements will be written to {@code outputStream}.
* @param offset the index of the first element of {@code array} to be written.
* @param length the number of elements of {@code array} to be written.
* @param outputStream an output stream.
*/
public static void STORE_KEYS(final KEY_TYPE array[][], final long offset, final long length, final OutputStream
outputStream) throws IOException {
 write(outputStream, array, offset, length);
}

/** Stores a big array to a given output stream.
*
* <p>Note that this method is going to be significantly faster than {@link #storeBytes(byte[][],DataOutput)}
* as it uses {@link OutputStream}'s bulk-read methods.
*
* @param array a big array whose elements will be written to {@code outputStream}.
* @param outputStream an output stream.
*/
public static void STORE_KEYS(final KEY_TYPE array[][], final OutputStream outputStream) throws
IOException {
 write(outputStream, array, 0, BIG_ARRAYS.length(array));
}

#endif
```

```
/** Loads elements from a given data input, storing them in a given array fragment.
*
* @param dataInput a data input.
* @param array an array which will be filled with data from {@code dataInput}.
* @param offset the index of the first element of {@code array} to be filled.
* @param length the number of elements of {@code array} to be filled.
* @return the number of elements actually read from {@code dataInput} (it might be less than {@code length} if
{@code dataInput} ends).
*/
public static int LOAD_KEYS(final DataInput dataInput, final KEY_TYPE[] array, final int offset, final int length)
throws IOException {
 PACKAGE.ARRAYS.ensureOffsetLength(array, offset, length);
 int i = 0;
 try {
  for(i = 0; i < length; i++) array[i + offset] = dataInput.READ_KEY();
 }
 catch(EOFException itsOk) {}
 return i;
}

/** Loads elements from a given data input, storing them in a given array.
*
* @param dataInput a data input.
* @param array an array which will be filled with data from {@code dataInput}.
* @return the number of elements actually read from {@code dataInput} (it might be less than the array length if
{@code dataInput} ends).
*/
public static int LOAD_KEYS(final DataInput dataInput, final KEY_TYPE[] array) throws IOException {
 int i = 0;
 try {
  final int length = array.length;
  for(i = 0; i < length; i++) array[i] = dataInput.READ_KEY();
 }
 catch(EOFException itsOk) {}
 return i;
}

/** Loads elements from a file given by a {@link File} object, storing them in a given array fragment.
*
* @param file a file.
* @param array an array which will be filled with data from the specified file.
* @param offset the index of the first element of {@code array} to be filled.
* @param length the number of elements of {@code array} to be filled.
* @return the number of elements actually read from the given file (it might be less than {@code length} if the file
is too short).
*/
public static int LOAD_KEYS(final File file, final KEY_TYPE[] array, final int offset, final int length) throws
IOException {
```

```
 PACKAGE.ARRAYS.ensureOffsetLength(array, offset, length);

 final FileInputStream fis = new FileInputStream(file);
#if KEY_CLASS_Byte
 final int result = read(fis, array, offset, length);
 fis.close();
 return result;
#else
 final DataInputStream dis = new DataInputStream(new FastBufferedInputStream(fis));

 int i = 0;
 try {
  for(i = 0; i < length; i++) array[i + offset] = dis.READ_KEY();
 }
 catch(EOFException itsOk) {}

 dis.close();
 return i;
#endif
}

/** Loads elements from a file given by a pathname, storing them in a given array fragment.
*
* @param filename a filename.
* @param array an array which will be filled with data from the specified file.
* @param offset the index of the first element of {@code array} to be filled.
* @param length the number of elements of {@code array} to be filled.
* @return the number of elements actually read from the given file (it might be less than {@code length} if the file is too short).
*/
public static int LOAD_KEYS(final CharSequence filename, final KEY_TYPE[] array, final int offset, final int length) throws IOException {
 return LOAD_KEYS(new File(filename.toString()), array, offset, length);
}

/** Loads elements from a file given by a {@link File} object, storing them in a given array.
*
* @param file a file.
* @param array an array which will be filled with data from the specified file.
* @return the number of elements actually read from the given file (it might be less than the array length if the file is too short).
*/
public static int LOAD_KEYS(final File file, final KEY_TYPE[] array) throws IOException {
 final FileInputStream fis = new FileInputStream(file);
#if KEY_CLASS_Byte
 final int result = read(fis, array, 0, array.length);
 fis.close();
 return result;
```

```
#else
 final DataInputStream dis = new DataInputStream(new FastBufferedInputStream(fis));

 int i = 0;
 try {
  final int length = array.length;
  for(i = 0; i < length; i++) array[i] = dis.READ_KEY();
 }
 catch(EOFException itsOk) {}

 dis.close();

 return i;
#endif
}

/** Loads elements from a file given by a pathname, storing them in a given array.
*
* @param filename a filename.
* @param array an array which will be filled with data from the specified file.
* @return the number of elements actually read from the given file (it might be less than the array length if the file
is too short).
*/
public static int LOAD_KEYS(final CharSequence filename, final KEY_TYPE[] array) throws IOException {
 return LOAD_KEYS(new File(filename.toString()), array);
}

/** Loads elements from a file given by a {@link File} object, storing them in a new array.
*
* <p>Note that the length of the returned array will be computed
* dividing the specified file size by the number of bytes used to
* represent each element.
*
* @param file a file.
* @return an array filled with the content of the specified file.
*/
public static KEY_TYPE[] LOAD_KEYS(final File file) throws IOException {
 final FileInputStream fis = new FileInputStream(file);

#if KEY_CLASS_Boolean
 final long length = fis.getChannel().size();
#else
 final long length = fis.getChannel().size() / (KEY_CLASS.SIZE / 8);
#endif

 if (length > Integer.MAX_VALUE) {
  fis.close();
  throw new IllegalArgumentException("File too long: " + fis.getChannel().size()+ " bytes (" + length + " elements)");
```

```
 }

 final KEY_TYPE[] array = new KEY_TYPE[(int)length];

#if KEY_CLASS_Byte
 if (read(fis, array, 0, (int)length) < length) throw new EOFException();
 fis.close();
#else
 final DataInputStream dis = new DataInputStream(new FastBufferedInputStream(fis));
 for(int i = 0; i < length; i++) array[i] = dis.READ_KEY();
 dis.close();
#endif
 return array;
}

/** Loads elements from a file given by a filename, storing them in a new array.
*
* <p>Note that the length of the returned array will be computed
* dividing the specified file size by the number of bytes used to
* represent each element.
*
* @param filename a filename.
* @return an array filled with the content of the specified file.
*/
public static KEY_TYPE[] LOAD_KEYS(final CharSequence filename) throws IOException {
 return LOAD_KEYS(new File(filename.toString()));
}

/** Stores an array fragment to a given data output.
*
* @param array an array whose elements will be written to {@code dataOutput}.
* @param offset the index of the first element of {@code array} to be written.
* @param length the number of elements of {@code array} to be written.
* @param dataOutput a data output.
*/
public static void STORE_KEYS(final KEY_TYPE array[], final int offset, final int length, final DataOutput
dataOutput) throws IOException {
 PACKAGE.ARRAYS.ensureOffsetLength(array, offset, length);
#if KEY_CLASS_Byte
 write(dataOutput, array, offset, length);
#else
 for(int i = 0; i < length; i++) dataOutput.WRITE_KEY(array[offset + i]);
#endif
}

/** Stores an array to a given data output.
*
* @param array an array whose elements will be written to {@code dataOutput}.
```

```
* @param dataOutput a data output.
*/
public static void STORE_KEYS(final KEY_TYPE array[], final DataOutput dataOutput) throws IOException {
#if KEY_CLASS_Byte
 write(dataOutput, array, 0, array.length);
#else
 final int length = array.length;
 for(int i = 0; i < length; i++) dataOutput.WRITE_KEY(array[i]);
#endif
}

/** Stores an array fragment to a file given by a {@link File} object.
*
* @param array an array whose elements will be written to {@code filename}.
* @param offset the index of the first element of {@code array} to be written.
* @param length the number of elements of {@code array} to be written.
* @param file a file.
*/
public static void STORE_KEYS(final KEY_TYPE array[], final int offset, final int length, final File file) throws
IOException {
 PACKAGE.ARRAYS.ensureOffsetLength(array, offset, length);
#if KEY_CLASS_Byte
 final OutputStream os = new FastBufferedOutputStream(new FileOutputStream(file));
 write(os, array, offset, length);
 os.close();
#else
 final DataOutputStream dos = new DataOutputStream(new FastBufferedOutputStream(new
FileOutputStream(file)));
 for(int i = 0; i < length; i++) dos.WRITE_KEY(array[offset + i]);
 dos.close();
#endif
}

/** Stores an array fragment to a file given by a pathname.
*
* @param array an array whose elements will be written to {@code filename}.
* @param offset the index of the first element of {@code array} to be written.
* @param length the number of elements of {@code array} to be written.
* @param filename a filename.
*/
public static void STORE_KEYS(final KEY_TYPE array[], final int offset, final int length, final CharSequence
filename) throws IOException {
 STORE_KEYS(array, offset, length, new File(filename.toString()));
}

/** Stores an array to a file given by a {@link File} object.
*
* @param array an array whose elements will be written to {@code filename}.
```

```
* @param file a file.
*/
public static void STORE_KEYS(final KEY_TYPE array[], final File file) throws IOException {
#if KEY_CLASS_Byte
 final OutputStream os = new FastBufferedOutputStream(new FileOutputStream(file));
 write(os, array, 0, array.length);
 os.close();
#else
 final int length = array.length;
 final DataOutputStream dos = new DataOutputStream(new FastBufferedOutputStream(new
FileOutputStream(file)));
 for(int i = 0; i < length; i++) dos.WRITE_KEY(array[i]);
 dos.close();
#endif
}

/** Stores an array to a file given by a pathname.
*
* @param array an array whose elements will be written to {@code filename}.
* @param filename a filename.
*/
public static void STORE_KEYS(final KEY_TYPE array[], final CharSequence filename) throws IOException {
 STORE_KEYS(array, new File(filename.toString()));
}




/** Loads elements from a given data input, storing them in a given big-array fragment.
*
* @param dataInput a data input.
* @param array a big array which will be filled with data from {@code dataInput}.
* @param offset the index of the first element of {@code bigArray} to be filled.
* @param length the number of elements of {@code bigArray} to be filled.
* @return the number of elements actually read from {@code dataInput} (it might be less than {@code length} if
{@code dataInput} ends).
*/
public static long LOAD_KEYS(final DataInput dataInput, final KEY_TYPE[][] array, final long offset, final long
length) throws IOException {
 PACKAGE.BIG_ARRAYS.ensureOffsetLength(array, offset, length);
 long c = 0;
 try {
  for(int i = segment(offset); i < segment(offset + length + SEGMENT_MASK); i++) {
   final KEY_TYPE[] t = array[i];
   final int l = (int)Math.min(t.length, offset + length - start(i));
   for(int d = (int)Math.max(0, offset - start(i)); d < l; d++) {
    t[d] = dataInput.READ_KEY();
```

```
    c++;
   }
  }
 }
 catch(EOFException itsOk) {}
 return c;
}

/** Loads elements from a given data input, storing them in a given big array.
*
* @param dataInput a data input.
* @param array a big array which will be filled with data from {@code dataInput}.
* @return the number of elements actually read from {@code dataInput} (it might be less than the array length if
{@code dataInput} ends).
*/
public static long LOAD_KEYS(final DataInput dataInput, final KEY_TYPE[][] array) throws IOException {
 long c = 0;
 try {
  for(int i = 0; i < array.length; i++) {
   final KEY_TYPE[] t = array[i];
   final int l = t.length;
   for(int d = 0; d < l; d++) {
    t[d] = dataInput.READ_KEY();
    c++;
   }
  }
 }
 catch(EOFException itsOk) {}
 return c;
}

/** Loads elements from a file given by a {@link File} object, storing them in a given big-array fragment.
*
* @param file a file.
* @param array a big array which will be filled with data from the specified file.
* @param offset the index of the first element of {@code array} to be filled.
* @param length the number of elements of {@code array} to be filled.
* @return the number of elements actually read from the given file (it might be less than {@code length} if the file
is too short).
*/
public static long LOAD_KEYS(final File file, final KEY_TYPE[][] array, final long offset, final long length)
throws IOException {
 PACKAGE.BIG_ARRAYS.ensureOffsetLength(array, offset, length);

 final FileInputStream fis = new FileInputStream(file);
#if KEY_CLASS_Byte
 final long result = read(fis, array, offset, length);
 fis.close();
```

```
 return result;
#else
 final DataInputStream dis = new DataInputStream(new FastBufferedInputStream(fis));

 long c = 0;
 try {
  for(int i = segment(offset); i < segment(offset + length + SEGMENT_MASK); i++) {
   final KEY_TYPE[] t = array[i];
   final int l = (int)Math.min(t.length, offset + length - start(i));
   for(int d = (int)Math.max(0, offset - start(i)); d < l; d++) {
    t[d] = dis.READ_KEY();
    c++;
   }
  }
 }
 catch(EOFException itsOk) {}
 dis.close();
 return c;
#endif
}

/** Loads elements from a file given by a pathname, storing them in a given big-array fragment.
*
* @param filename a filename.
* @param array an array which will be filled with data from the specified file.
* @param offset the index of the first element of {@code array} to be filled.
* @param length the number of elements of {@code array} to be filled.
* @return the number of elements actually read from the given file (it might be less than {@code length} if the file
is too short).
*/
public static long LOAD_KEYS(final CharSequence filename, final KEY_TYPE[][] array, final long offset, final
long length) throws IOException {
 return LOAD_KEYS(new File(filename.toString()), array, offset, length);
}

/** Loads elements from a file given by a {@link File} object, storing them in a given big array.
*
* @param file a file.
* @param array a big array which will be filled with data from the specified file.
* @return the number of elements actually read from the given file (it might be less than the array length if the file
is too short).
*/
public static long LOAD_KEYS(final File file, final KEY_TYPE[][] array) throws IOException {
 final FileInputStream fis = new FileInputStream(file);
#if KEY_CLASS_Byte
 final long result = read(fis, array, 0, BIG_ARRAYS.length(array));
 fis.close();
 return result;
```

```
#else
 final DataInputStream dis = new DataInputStream(new FastBufferedInputStream(fis));

 long c = 0;
 try {
  for(int i = 0; i < array.length; i++) {
   final KEY_TYPE[] t = array[i];
   final int l = t.length;
   for(int d = 0; d < l; d++) {
    t[d] = dis.READ_KEY();
    c++;
   }
  }
 }
 catch(EOFException itsOk) {}

 dis.close();

 return c;
#endif
}

/** Loads elements from a file given by a pathname, storing them in a given big array.
*
* @param filename a filename.
* @param array a big array which will be filled with data from the specified file.
* @return the number of elements actually read from the given file (it might be less than the array length if the file
is too short).
*/
public static long LOAD_KEYS(final CharSequence filename, final KEY_TYPE[][] array) throws IOException {
 return LOAD_KEYS(new File(filename.toString()), array);
}

/** Loads elements from a file given by a {@link File} object, storing them in a new big array.
*
* <p>Note that the length of the returned big array will be computed
* dividing the specified file size by the number of bytes used to
* represent each element.
*
* @param file a file.
* @return a big array filled with the content of the specified file.
*/
public static KEY_TYPE[][] LOAD_KEYS_BIG(final File file) throws IOException {
 final FileInputStream fis = new FileInputStream(file);

#if KEY_CLASS_Boolean
 final long length = fis.getChannel().size();
#else
```

```
 final long length = fis.getChannel().size() / (KEY_CLASS.SIZE / 8);
#endif

 final KEY_TYPE[][] array = BIG_ARRAYS.newBigArray(length);

#if KEY_CLASS_Byte
 if (read(fis, array, 0, length) < length) throw new EOFException();
 fis.close();
#else
 final DataInputStream dis = new DataInputStream(new FastBufferedInputStream(fis));

 for(int i = 0; i < array.length; i++) {
  final KEY_TYPE[] t = array[i];
  final int l = t.length;
  for(int d = 0; d < l; d++) t[d] = dis.READ_KEY();
 }

 dis.close();
#endif
 return array;
}

/** Loads elements from a file given by a filename, storing them in a new big array.
*
* <p>Note that the length of the returned big array will be computed
* dividing the specified file size by the number of bytes used to
* represent each element.
*
* @param filename a filename.
* @return a big array filled with the content of the specified file.
*/
public static KEY_TYPE[][] LOAD_KEYS_BIG(final CharSequence filename) throws IOException {
 return LOAD_KEYS_BIG(new File(filename.toString()));
}

/** Stores an array fragment to a given data output.
*
* @param array an array whose elements will be written to {@code dataOutput}.
* @param offset the index of the first element of {@code array} to be written.
* @param length the number of elements of {@code array} to be written.
* @param dataOutput a data output.
*/
public static void STORE_KEYS(final KEY_TYPE array[][], final long offset, final long length, final DataOutput
dataOutput) throws IOException {
 PACKAGE.BIG_ARRAYS.ensureOffsetLength(array, offset, length);
#if KEY_CLASS_Byte
 write(dataOutput, array, offset, length);
#else
```

```
 for(int i = segment(offset); i < segment(offset + length + SEGMENT_MASK); i++) {
  final KEY_TYPE[] t = array[i];
  final int l = (int)Math.min(t.length, offset + length - start(i));
  for(int d = (int)Math.max(0, offset - start(i)); d < l; d++) dataOutput.WRITE_KEY(t[d]);
 }
#endif
}

/** Stores a big array to a given data output.
*
* @param array a big array whose elements will be written to {@code dataOutput}.
* @param dataOutput a data output.
*/
public static void STORE_KEYS(final KEY_TYPE array[][], final DataOutput dataOutput) throws IOException {
#if KEY_CLASS_Byte
 write(dataOutput, array, 0, BIG_ARRAYS.length(array));
#else
 for(int i = 0; i < array.length; i++) {
  final KEY_TYPE[] t = array[i];
  final int l = t.length;
  for(int d = 0; d < l; d++) dataOutput.WRITE_KEY(t[d]);
 }
#endif
}

/** Stores a big-array fragment to a file given by a {@link File} object.
*
* @param array a big array whose elements will be written to {@code filename}.
* @param offset the index of the first element of {@code array} to be written.
* @param length the number of elements of {@code array} to be written.
* @param file a file.
*/
public static void STORE_KEYS(final KEY_TYPE array[][], final long offset, final long length, final File file)
throws IOException {
 PACKAGE.BIG_ARRAYS.ensureOffsetLength(array, offset, length);
#if KEY_CLASS_Byte
 final OutputStream os = new FastBufferedOutputStream(new FileOutputStream(file));
 write(os, array, offset, length);
 os.close();
#else
 final DataOutputStream dos = new DataOutputStream(new FastBufferedOutputStream(new
FileOutputStream(file)));
 for(int i = segment(offset); i < segment(offset + length + SEGMENT_MASK); i++) {
  final KEY_TYPE[] t = array[i];
  final int l = (int)Math.min(t.length, offset + length - start(i));
  for(int d = (int)Math.max(0, offset - start(i)); d < l; d++) dos.WRITE_KEY(t[d]);
 }
 dos.close();
```

```
#endif
}

/** Stores a big-array fragment to a file given by a pathname.
*
* @param array a big array whose elements will be written to {@code filename}.
* @param offset the index of the first element of {@code array} to be written.
* @param length the number of elements of {@code array} to be written.
* @param filename a filename.
*/
public static void STORE_KEYS(final KEY_TYPE array[][], final long offset, final long length, final
CharSequence filename) throws IOException {
 STORE_KEYS(array, offset, length, new File(filename.toString()));
}

/** Stores an array to a file given by a {@link File} object.
*
* @param array an array whose elements will be written to {@code filename}.
* @param file a file.
*/
public static void STORE_KEYS(final KEY_TYPE array[][], final File file) throws IOException {
#if KEY_CLASS_Byte
 final OutputStream os = new FastBufferedOutputStream(new FileOutputStream(file));
 write(os, array, 0, BIG_ARRAYS.length(array));
 os.close();
#else
 final DataOutputStream dos = new DataOutputStream(new FastBufferedOutputStream(new
FileOutputStream(file)));
 for(int i = 0; i < array.length; i++) {
  final KEY_TYPE[] t = array[i];
  final int l = t.length;
  for(int d = 0; d < l; d++) dos.WRITE_KEY(t[d]);
 }
 dos.close();
#endif
}

/** Stores a big array to a file given by a pathname.
*
* @param array a big array whose elements will be written to {@code filename}.
* @param filename a filename.
*/
public static void STORE_KEYS(final KEY_TYPE array[][], final CharSequence filename) throws IOException {
 STORE_KEYS(array, new File(filename.toString()));
}

/** Stores the element returned by an iterator to a given data output.
```

```
*
* @param i an iterator whose output will be written to {@code dataOutput}.
* @param dataOutput a filename.
*/
public static void STORE_KEYS(final KEY_ITERATOR i, final DataOutput dataOutput) throws IOException {
 while(i.hasNext()) dataOutput.WRITE_KEY(i.NEXT_KEY());
}

/** Stores the element returned by an iterator to a file given by a {@link File} object.
*
* @param i an iterator whose output will be written to {@code filename}.
* @param file a file.
*/
public static void STORE_KEYS(final KEY_ITERATOR i, final File file) throws IOException {
 final DataOutputStream dos = new DataOutputStream(new FastBufferedOutputStream(new
FileOutputStream(file)));
 while(i.hasNext()) dos.WRITE_KEY(i.NEXT_KEY());
 dos.close();
}

/** Stores the element returned by an iterator to a file given by a pathname.
*
* @param i an iterator whose output will be written to {@code filename}.
* @param filename a filename.
*/
public static void STORE_KEYS(final KEY_ITERATOR i, final CharSequence filename) throws IOException {
 STORE_KEYS(i, new File(filename.toString()));
}

/** A wrapper that exhibits the content of a data input stream as a type-specific iterator. */

private static final class KEY_DATA_INPUT_WRAPPER implements KEY_ITERATOR {
 private final DataInput dataInput;
 private boolean toAdvance = true;
 private boolean endOfProcess = false;
 private KEY_TYPE next;

 public KEY_DATA_INPUT_WRAPPER(final DataInput dataInput) {
  this.dataInput = dataInput;
 }

 @Override
 public boolean hasNext() {
  if (! toAdvance) return ! endOfProcess;

  toAdvance = false;

  try { next = dataInput.READ_KEY(); }
```

```
 catch(EOFException eof) { endOfProcess = true; }
 catch(IOException rethrow) { throw new RuntimeException(rethrow); }

 return ! endOfProcess;
 }

 @Override
 public KEY_TYPE NEXT_KEY() {
  if (! hasNext()) throw new NoSuchElementException();
  toAdvance = true;
  return next;
 }
}

/** Wraps the given data input stream into an iterator.
*
* @param dataInput a data input.
*/
public static KEY_ITERATOR AS_KEY_ITERATOR(final DataInput dataInput) {
 return new KEY_DATA_INPUT_WRAPPER(dataInput);
}

/** Wraps a file given by a {@link File} object into an iterator.
*
* @param file a file.
*/
public static KEY_ITERATOR AS_KEY_ITERATOR(final File file) throws IOException {
 return new KEY_DATA_INPUT_WRAPPER(new DataInputStream(new FastBufferedInputStream(new
FileInputStream(file))));
}

/** Wraps a file given by a pathname into an iterator.
*
* @param filename a filename.
*/
public static KEY_ITERATOR AS_KEY_ITERATOR(final CharSequence filename) throws IOException {
 return AS_KEY_ITERATOR(new File(filename.toString()));
}

/** Wraps a file given by a {@link File} object into an iterable object.
*
* @param file a file.
*/
public static KEY_ITERABLE AS_KEY_ITERABLE(final File file) {
 return () -> {
  try { return AS_KEY_ITERATOR(file); }
  catch(IOException e) { throw new RuntimeException(e); }
 };
```

```
}

/** Wraps a file given by a pathname into an iterable object.
*
* @param filename a filename.
*/
public static KEY_ITERABLE AS_KEY_ITERABLE(final CharSequence filename) {
 return () -> {
  try { return AS_KEY_ITERATOR(filename); }
  catch(IOException e) { throw new RuntimeException(e); }
 };
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/BinIOFragment.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2010-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import java.util.List;
import it.unimi.dsi.fastutil.BigList;
import it.unimi.dsi.fastutil.Size64;

#if ! KEY_CLASS_Reference

/** A type-specific {@link BigList}; provides some additional methods that use polymorphism to avoid (un)boxing.
*
* <p>Additionally, this interface strengthens {@link #iterator()}, {@link #listIterator()},
* {@link #listIterator(long)} and {@link #subList(long,long)}.
*
```

```
* <p>Besides polymorphic methods, this interfaces specifies methods to copy into an array or remove contiguous
* sublists. Although the abstract implementation of this interface provides simple, one-by-one implementations
* of these methods, it is expected that concrete implementation override them with optimized versions.
*
* @see List
*/

public interface BIG_LIST KEY_GENERIC extends BigList<KEY_GENERIC_CLASS>, COLLECTION
KEY_GENERIC, Size64, Comparable<BigList<? extends KEY_GENERIC_CLASS>> {
#else

/** A type-specific {@link BigList}; provides some additional methods that use polymorphism to avoid (un)boxing.
*
* <p>Additionally, this interface strengthens {@link #listIterator()},
* {@link #listIterator(long)} and {@link #subList(long,long)}.
*
* <p>Besides polymorphic methods, this interfaces specifies methods to copy into an array or remove contiguous
* sublists. Although the abstract implementation of this interface provides simple, one-by-one implementations
* of these methods, it is expected that concrete implementation override them with optimized versions.
*
* @see List
*/

public interface BIG_LIST KEY_GENERIC extends BigList<KEY_GENERIC_CLASS>, COLLECTION
KEY_GENERIC, Size64 {
#endif

 /** Returns a type-specific iterator on the elements of this list.
  *
  * <p>Note that this specification strengthens the one given in {@link java.util.Collection#iterator()}.
  * @see java.util.Collection#iterator()
  */
 @Override
 KEY_BIG_LIST_ITERATOR KEY_GENERIC iterator();


 /** Returns a type-specific big-list iterator on this type-specific big list.
  *
  * <p>Note that this specification strengthens the one given in {@link BigList#listIterator()}.
  * @see BigList#listIterator()
  */
 @Override
 KEY_BIG_LIST_ITERATOR KEY_GENERIC listIterator();

 /** Returns a type-specific list iterator on this type-specific big list starting at a given index.
  *
  * <p>Note that this specification strengthens the one given in {@link BigList#listIterator(long)}.
  * @see BigList#listIterator(long)
```

```
 */
@Override
KEY_BIG_LIST_ITERATOR KEY_GENERIC listIterator(long index);

/** Returns a type-specific view of the portion of this type-specific big list from the index {@code from}, inclusive, to the index {@code to}, exclusive.
 *
 * <p>Note that this specification strengthens the one given in {@link BigList#subList(long,long)}.
 *
 * @see BigList#subList(long,long)
 */
@Override
BIG_LIST KEY_GENERIC subList(long from, long to);

/** Copies (hopefully quickly) elements of this type-specific big list into the given big array.
 *
 * @param from the start index (inclusive).
 * @param a the destination big array.
 * @param offset the offset into the destination big array where to store the first element copied.
 * @param length the number of elements to be copied.
 */
void getElements(long from, KEY_TYPE a[][], long offset, long length);

/** Removes (hopefully quickly) elements of this type-specific big list.
 *
 * @param from the start index (inclusive).
 * @param to the end index (exclusive).
 */
void removeElements(long from, long to);

/** Add (hopefully quickly) elements to this type-specific big list.
 *
 * @param index the index at which to add elements.
 * @param a the big array containing the elements.
 */
void addElements(long index, KEY_GENERIC_TYPE a[][]);

/** Add (hopefully quickly) elements to this type-specific big list.
 *
 * @param index the index at which to add elements.
 * @param a the big array containing the elements.
 * @param offset the offset of the first element to add.
 * @param length the number of elements to add.
 */
void addElements(long index, KEY_GENERIC_TYPE a[][], long offset, long length);

#if KEYS_PRIMITIVE
```

```
 /** Inserts the specified element at the specified position in this type-specific big list (optional operation).
  * @see BigList#add(long,Object)
  */
 void add(long index, KEY_TYPE key);

 /** Inserts all of the elements in the specified type-specific collection into this type-specific big list at the specified position (optional operation).
  * @see List#addAll(int,java.util.Collection)
  */
 boolean addAll(long index, COLLECTION c);

 /** Inserts all of the elements in the specified type-specific big list into this type-specific big list at the specified position (optional operation).
  * @see List#addAll(int,java.util.Collection)
  */
 boolean addAll(long index, BIG_LIST c);

 /** Appends all of the elements in the specified type-specific big list to the end of this type-specific big list (optional operation).
  * @see List#addAll(int,java.util.Collection)
  */
 boolean addAll(BIG_LIST c);

 /** Returns the element at the specified position.
  * @see BigList#get(long)
  */
 KEY_TYPE GET_KEY(long index);

 /** Removes the element at the specified position.
  * @see BigList#remove(long) */
 KEY_TYPE REMOVE_KEY(long index);

 /** Replaces the element at the specified position in this big list with the specified element (optional operation).
  * @see BigList#set(long,Object)
  */
 KEY_TYPE set(long index, KEY_TYPE k);

 /** Returns the index of the first occurrence of the specified element in this type-specific big list, or -1 if this big list does not contain the element.
  * @see BigList#indexOf(Object)
  */
 long indexOf(KEY_TYPE k);

 /** Returns the index of the last occurrence of the specified element in this type-specific big list, or -1 if this big list does not contain the element.
  * @see BigList#lastIndexOf(Object)
  */
 long lastIndexOf(KEY_TYPE k);
```

```
/** {@inheritDoc}
 * @deprecated Please use the corresponding type-specific method instead. */
@Deprecated
@Override
void add(long index, KEY_CLASS key);

/** {@inheritDoc}
 * @deprecated Please use the corresponding type-specific method instead. */
@Deprecated
@Override
KEY_GENERIC_CLASS get(long index);

/** {@inheritDoc}
 * @deprecated Please use the corresponding type-specific method instead. */
@Deprecated
@Override
long indexOf(Object o);

/** {@inheritDoc}
 * @deprecated Please use the corresponding type-specific method instead. */
@Deprecated
@Override
long lastIndexOf(Object o);

/** {@inheritDoc}
 * @deprecated Please use the corresponding type-specific method instead. */
@Deprecated
@Override
KEY_GENERIC_CLASS remove(long index);

/** {@inheritDoc}
 * @deprecated Please use the corresponding type-specific method instead. */
@Deprecated
@Override
KEY_GENERIC_CLASS set(long index, KEY_GENERIC_CLASS k);
#endif

}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/BigList.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2010-2017 Sebastiano Vigna
*
```

```
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import it.unimi.dsi.fastutil.BigListIterator;
import it.unimi.dsi.fastutil.SafeMath;

/** A type-specific {@link BigListIterator}.
*
* @see BigListIterator
*/

public interface KEY_BIG_LIST_ITERATOR KEY_GENERIC extends KEY_BIDI_ITERATOR
KEY_GENERIC, BigListIterator<KEY_GENERIC_CLASS> {

 /**
  * Replaces the last element returned by {@link BigListIterator#next() next()} or
  * {@link BigListIterator#previous() previous()} with the specified element (optional operation).
  * @see java.util.ListIterator#set(Object)
  */

#if KEYS_REFERENCE
 @Override
#endif
 default void set(@SuppressWarnings("unused") final KEY_GENERIC_TYPE k) { throw new
UnsupportedOperationException(); }

 /**
  * Inserts the specified element into the list (optional operation).
  * @see java.util.ListIterator#add(Object)
  */

#if KEYS_REFERENCE
 @Override
#endif
 default void add(@SuppressWarnings("unused") final KEY_GENERIC_TYPE k)  { throw new
```

```
UnsupportedOperationException(); }

#if KEYS_PRIMITIVE

 /** Replaces the last element returned by {@link #next()} or {@link #previous()} with the specified element
(optional operation).
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 default void set(final KEY_GENERIC_CLASS k) { set(k.KEY_VALUE()); }

 /** Inserts the specified element into the list (optional operation).
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 default void add(final KEY_GENERIC_CLASS k) { add(k.KEY_VALUE()); }

#endif


 /** Skips the given number of elements.
  *
  * <p>The effect of this call is exactly the same as that of
  * calling {@link BigListIterator#next() next()} for {@code n} times (possibly stopping
  * if {@link #hasNext()} becomes false).
  *
  * @param n the number of elements to skip.
  * @return the number of elements actually skipped.
  * @see BigListIterator#next()
  */
 default long skip(final long n) {
  long i = n;
  while(i-- != 0 && hasNext()) NEXT_KEY();
  return n - i - 1;
 }

 /** Moves back for the given number of elements.
  *
  * <p>The effect of this call is exactly the same as that of
  * calling {@link BigListIterator#previous() previous()} for {@code n} times (possibly stopping
  * if {@link #hasPrevious()} becomes false).
  *
  * @param n the number of elements to skip back.
  * @return the number of elements actually skipped.
  * @see BigListIterator#previous()
  */
 default long back(final long n) {
  long i = n;
```

```
 while(i-- != 0 && hasPrevious()) PREV_KEY();
 return n - i - 1;
 }

 /** {@inheritDoc}
  */
 @Override
 default int skip(int n) {
  return SafeMath.safeLongToInt(skip((long) n));
 }
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/BigListIterator.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

/**  An abstract class facilitating the creation of type-specific {@linkplain it.unimi.dsi.fastutil.BigListIterator big-list iterators}.
*
* @deprecated As of fastutil 8 this class is no longer necessary, as its previous abstract
* methods are now default methods of the type-specific interface.
*/

@Deprecated
public abstract class KEY_ABSTRACT_BIG_LIST_ITERATOR KEY_GENERIC extends
KEY_ABSTRACT_BIDI_ITERATOR KEY_GENERIC implements KEY_BIG_LIST_ITERATOR
KEY_GENERIC {
 protected KEY_ABSTRACT_BIG_LIST_ITERATOR() {}
```

}

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/AbstractBigListIterator.drv
No license file was found, but licenses were detected in source scan.


```
package PACKAGE;

#if KEYS_REFERENCE
import it.unimi.dsi.fastutil.Stack;
#endif

import java.util.List;
import java.util.Iterator;
import java.util.ListIterator;
import java.util.Collection;
import java.util.NoSuchElementException;

/**  An abstract class providing basic methods for lists implementing a type-specific list interface.
*
* <p>As an additional bonus, this class implements on top of the list operations a type-specific stack.
*/

public abstract class ABSTRACT_LIST KEY_GENERIC extends ABSTRACT_COLLECTION KEY_GENERIC implements LIST KEY_GENERIC, STACK KEY_GENERIC {

 protected ABSTRACT_LIST() {}

 /** Ensures that the given index is nonnegative and not greater than the list size.
  *
```

```
 * @param index an index.
 * @throws IndexOutOfBoundsException if the given index is negative or greater than the list size.
 */
protected void ensureIndex(final int index) {
 if (index < 0)  throw new IndexOutOfBoundsException("Index (" + index + ") is negative");
 if (index > size()) throw new IndexOutOfBoundsException("Index (" + index + ") is greater than list size (" +
(size()) + ")");
}

/** Ensures that the given index is nonnegative and smaller than the list size.
 *
 * @param index an index.
 * @throws IndexOutOfBoundsException if the given index is negative or not smaller than the list size.
 */
protected void ensureRestrictedIndex(final int index) {
 if (index < 0)  throw new IndexOutOfBoundsException("Index (" + index + ") is negative");
 if (index >= size()) throw new IndexOutOfBoundsException("Index (" + index + ") is greater than or equal to list
size (" + (size()) + ")");
}

/** {@inheritDoc}
 *
 * <p>This implementation always throws an {@link UnsupportedOperationException}.
 */
@Override
public void add(final int index, final KEY_GENERIC_TYPE k) {
 throw new UnsupportedOperationException();
}

/** {@inheritDoc}
 *
 * <p>This implementation delegates to the type-specific version of {@link List#add(int, Object)}.
 */
@Override
public boolean add(final KEY_GENERIC_TYPE k) {
 add(size(), k);
 return true;
}

/** {@inheritDoc}
 *
 * <p>This implementation always throws an {@link UnsupportedOperationException}.
 */
@Override
public KEY_GENERIC_TYPE REMOVE_KEY(final int i) {
 throw new UnsupportedOperationException();
}
```

```
/** {@inheritDoc}
 *
 * <p>This implementation always throws an {@link UnsupportedOperationException}.
 */
@Override
public KEY_GENERIC_TYPE set(final int index, final KEY_GENERIC_TYPE k) {
 throw new UnsupportedOperationException();
}

/** Adds all of the elements in the specified collection to this list (optional operation). */
@Override
public boolean addAll(int index, final Collection<? extends KEY_GENERIC_CLASS> c) {
 ensureIndex(index);
 final Iterator<? extends KEY_GENERIC_CLASS> i = c.iterator();
 final boolean retVal = i.hasNext();
 while(i.hasNext()) add(index++, KEY_CLASS2TYPE(i.next()));
 return retVal;
}

/** {@inheritDoc}
 *
 * <p>This implementation delegates to the type-specific version of {@link List#addAll(int, Collection)}.
 */
@Override
public boolean addAll(final Collection<? extends KEY_GENERIC_CLASS> c) {
 return addAll(size(), c);
}

/** {@inheritDoc}
 *
 * <p>This implementation delegates to {@link #listIterator()}.
 */
@Override
public KEY_LIST_ITERATOR KEY_GENERIC iterator() {
 return listIterator();
}

/** {@inheritDoc}
 *
 * <p>This implementation delegates to {@link #listIterator(int) listIterator(0)}.
 */
@Override
public KEY_LIST_ITERATOR KEY_GENERIC listIterator() {
 return listIterator(0);
}

/** {@inheritDoc}
 * <p>This implementation is based on the random-access methods. */
```

```
 @Override
 public KEY_LIST_ITERATOR KEY_GENERIC listIterator(final int index) {
  ensureIndex(index);

  return new KEY_LIST_ITERATOR KEY_GENERIC() {
    int pos = index, last = -1;

    @Override
    public boolean hasNext() { return pos < ABSTRACT_LIST.this.size(); }
    @Override
    public boolean hasPrevious() { return pos > 0; }
    @Override
    public KEY_GENERIC_TYPE NEXT_KEY() { if (! hasNext()) throw new NoSuchElementException(); return
ABSTRACT_LIST.this.GET_KEY(last = pos++); }
    @Override
    public KEY_GENERIC_TYPE PREV_KEY() { if (! hasPrevious()) throw new NoSuchElementException(); return
ABSTRACT_LIST.this.GET_KEY(last = --pos); }
    @Override
    public int nextIndex() { return pos; }
    @Override
    public int previousIndex() { return pos - 1; }
    @Override
    public void add(final KEY_GENERIC_TYPE k) {
     ABSTRACT_LIST.this.add(pos++, k);
     last = -1;
    }
    @Override
    public void set(final KEY_GENERIC_TYPE k) {
     if (last == -1) throw new IllegalStateException();
     ABSTRACT_LIST.this.set(last, k);
    }
    @Override
    public void remove() {
     if (last == -1) throw new IllegalStateException();
     ABSTRACT_LIST.this.REMOVE_KEY(last);
     /* If the last operation was a next(), we are removing an element *before* us, and we must decrease pos
correspondingly. */
     if (last < pos) pos--;
     last = -1;
    }
   };
 }

 /** Returns true if this list contains the specified element.
  * <p>This implementation delegates to {@code indexOf()}.
  * @see List#contains(Object)
  */
```

```
@Override
public boolean contains(final KEY_TYPE k) {
 return indexOf(k) >= 0;
}

@Override
public int indexOf(final KEY_TYPE k) {
 final KEY_LIST_ITERATOR KEY_GENERIC i = listIterator();
 KEY_GENERIC_TYPE e;
 while(i.hasNext()) {
  e = i.NEXT_KEY();
  if (KEY_EQUALS(k, e)) return i.previousIndex();
 }
 return -1;
}

@Override
public int lastIndexOf(final KEY_TYPE k) {
 KEY_LIST_ITERATOR KEY_GENERIC i = listIterator(size());
 KEY_GENERIC_TYPE e;
 while(i.hasPrevious()) {
  e = i.PREV_KEY();
  if (KEY_EQUALS(k, e)) return i.nextIndex();
 }
 return -1;
}

@Override
public void size(final int size) {
 int i = size();
 if (size > i) while(i++ < size) add(KEY_NULL);
 else while(i-- != size) REMOVE_KEY(i);
}


@Override
public LIST KEY_GENERIC subList(final int from, final int to) {
 ensureIndex(from);
 ensureIndex(to);
 if (from > to) throw new IndexOutOfBoundsException("Start index (" + from + ") is greater than end index (" + to +
")");

 return new SUBLIST KEY_GENERIC_DIAMOND(this, from, to);
}

/** {@inheritDoc}
 *
 * <p>This is a trivial iterator-based implementation. It is expected that
```

```
 * implementations will override this method with a more optimized version.
 */
@Override
public void removeElements(final int from, final int to) {
 ensureIndex(to);
 KEY_LIST_ITERATOR KEY_GENERIC i = listIterator(from);
 int n = to - from;
 if (n < 0) throw new IllegalArgumentException("Start index (" + from + ") is greater than end index (" + to + ")");
 while(n-- != 0) {
  i.NEXT_KEY();
  i.remove();
 }
}

/** {@inheritDoc}
 *
 * <p>This is a trivial iterator-based implementation. It is expected that
 * implementations will override this method with a more optimized version.
 */
@Override
public void addElements(int index, final KEY_GENERIC_TYPE a[], int offset, int length) {
 ensureIndex(index);
 if (offset < 0) throw new ArrayIndexOutOfBoundsException("Offset (" + offset + ") is negative");
 if (offset + length > a.length) throw new ArrayIndexOutOfBoundsException("End index (" + (offset + length) + ") is
greater than array length (" + a.length + ")");
 while(length-- != 0) add(index++, a[offset++]);
}

/** {@inheritDoc}
 *
 * <p>This implementation delegates to the analogous method for array fragments.
 */
@Override
public void addElements(final int index, final KEY_GENERIC_TYPE a[]) {
 addElements(index, a, 0, a.length);
}

/** {@inheritDoc}
 *
 * <p>This is a trivial iterator-based implementation. It is expected that
 * implementations will override this method with a more optimized version.
 */
@Override
public void getElements(final int from, final KEY_TYPE a[], int offset, int length) {
 KEY_LIST_ITERATOR KEY_GENERIC i = listIterator(from);
 if (offset < 0) throw new ArrayIndexOutOfBoundsException("Offset (" + offset + ") is negative");
 if (offset + length > a.length) throw new ArrayIndexOutOfBoundsException("End index (" + (offset + length) + ") is
greater than array length (" + a.length + ")");
```

```
 if (from + length > size()) throw new IndexOutOfBoundsException("End index (" + (from + length) + ") is greater
than list size (" + size() + ")");
 while(length-- != 0) a[offset++] = i.NEXT_KEY();
 }

 /** {@inheritDoc}
  * <p>This implementation delegates to {@link #removeElements(int, int)}.*/
 @Override
 public void clear() {
  removeElements(0, size());
 }


#if ! KEY_CLASS_Reference
 private boolean valEquals(final Object a, final Object b) {
  return a == null ? b == null : a.equals(b);
 }
#endif

 /** Returns the hash code for this list, which is identical to {@link java.util.List#hashCode()}.
  *
  * @return the hash code for this list.
  */
 @Override
 public int hashCode() {
  KEY_ITERATOR KEY_GENERIC i = iterator();
  int h = 1, s = size();
  while (s-- != 0) {
   KEY_GENERIC_TYPE k = i.NEXT_KEY();
   h = 31 * h + KEY2JAVAHASH(k);
  }
  return h;
 }


 @Override
 public boolean equals(final Object o) {
  if (o == this) return true;
  if (! (o instanceof List)) return false;

  final List<?> l = (List<?>)o;
  int s = size();
  if (s != l.size()) return false;

#if KEYS_PRIMITIVE
  if (l instanceof LIST) {
   final KEY_LIST_ITERATOR KEY_GENERIC i1 = listIterator(), i2 = ((LIST KEY_GENERIC)l).listIterator();
   while(s-- != 0) if (i1.NEXT_KEY() != i2.NEXT_KEY()) return false;
```

```
  return true;
  }
#endif

  final ListIterator<?> i1 = listIterator(), i2 = l.listIterator();

#if KEY_CLASS_Reference
  while(s-- !=  0) if (i1.next() != i2.next()) return false;
#else
  while(s-- !=  0) if (! valEquals(i1.next(), i2.next())) return false;
#endif
  return true;
  }

#if ! KEY_CLASS_Reference
 /** Compares this list to another object. If the
  * argument is a {@link java.util.List}, this method performs a lexicographical comparison; otherwise,
  * it throws a {@code ClassCastException}.
  *
  * @param l a list.
  * @return if the argument is a {@link java.util.List}, a negative integer,
  * zero, or a positive integer as this list is lexicographically less than, equal
  * to, or greater than the argument.
  * @throws ClassCastException if the argument is not a list.
  */

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 @Override
 public int compareTo(final List<? extends KEY_GENERIC_CLASS> l) {
  if (l == this) return 0;

  if (l instanceof LIST) {

   final KEY_LIST_ITERATOR KEY_GENERIC i1 = listIterator(), i2 = ((LIST KEY_GENERIC)l).listIterator();
   int r;
   KEY_GENERIC_TYPE e1, e2;

   while(i1.hasNext() && i2.hasNext()) {
    e1 = i1.NEXT_KEY();
    e2 = i2.NEXT_KEY();
    if ((r = KEY_CMP(e1, e2)) != 0) return r;
   }
   return i2.hasNext() ? -1 : (i1.hasNext() ? 1 : 0);
  }

  ListIterator<? extends KEY_GENERIC_CLASS> i1 = listIterator(), i2 = l.listIterator();
  int r;
```

```
 while(i1.hasNext() && i2.hasNext()) {
  if ((r = ((Comparable<? super KEY_GENERIC_CLASS>)i1.next()).compareTo(i2.next())) != 0) return r;
 }
 return i2.hasNext() ? -1 : (i1.hasNext() ? 1 : 0);
}
#endif

@Override
public void push(final KEY_GENERIC_TYPE o) {
 add(o);
}

@Override
public KEY_GENERIC_TYPE POP() {
 if (isEmpty()) throw new NoSuchElementException();
 return REMOVE_KEY(size() - 1);
}

@Override
public KEY_GENERIC_TYPE TOP() {
 if (isEmpty()) throw new NoSuchElementException();
 return GET_KEY(size() - 1);
}

@Override
public KEY_GENERIC_TYPE PEEK(final int i) {
 return GET_KEY(size() - 1 - i);
}

#if KEYS_PRIMITIVE

/** Removes a single instance of the specified element from this collection, if it is present (optional operation).
 * <p>This implementation delegates to {@code indexOf()}.
 * @see List#remove(Object)
 */
@Override
public boolean rem(final KEY_TYPE k) {
 int index = indexOf(k);
 if (index == -1) return false;
 REMOVE_KEY(index);
 return true;
}

@Override
public boolean addAll(int index, final COLLECTION c) {
 ensureIndex(index);
 final KEY_ITERATOR i = c.iterator();
 final boolean retVal = i.hasNext();
```

```
 while(i.hasNext()) add(index++, i.NEXT_KEY());
 return retVal;
 }

 /** {@inheritDoc}
  *
  * <p>This implementation delegates to the type-specific version of {@link List#addAll(int, Collection)}.
  */
 @Override
 public boolean addAll(final int index, final LIST l) {
  return addAll(index, (COLLECTION)l);
 }

 /** {@inheritDoc}
  *
  * <p>This implementation delegates to the type-specific version of {@link List#addAll(int, Collection)}.
  */
 @Override
 public boolean addAll(final COLLECTION c) {
  return addAll(size(), c);
 }

 /** {@inheritDoc}
  *
  * <p>This implementation delegates to the type-specific list version of {@link List#addAll(int, Collection)}.
  */
 @Override
 public boolean addAll(final LIST l) {
  return addAll(size(), l);
 }

#endif

 @Override
 public String toString() {
  final StringBuilder s = new StringBuilder();
  final KEY_ITERATOR KEY_GENERIC i = iterator();
  int n = size();
  KEY_GENERIC_TYPE k;
  boolean first = true;

  s.append("[");

  while(n-- != 0) {
   if (first) first = false;
   else s.append(", ");
   k = i.NEXT_KEY();
#if KEYS_REFERENCE
```

```
  if (this == k) s.append("(this list)"); else
#endif
    s.append(String.valueOf(k));
  }

  s.append("]");
  return s.toString();
  }

  /** A class implementing a sublist view. */
  public static class SUBLIST KEY_GENERIC extends ABSTRACT_LIST KEY_GENERIC implements
java.io.Serializable {
  private static final long serialVersionUID = -7046029254386353129L;
  /** The list this sublist restricts. */
  protected final LIST KEY_GENERIC l;
  /** Initial (inclusive) index of this sublist. */
  protected final int from;
  /** Final (exclusive) index of this sublist. */
  protected int to;

  public SUBLIST(final LIST KEY_GENERIC l, final int from, final int to) {
   this.l = l;
   this.from = from;
   this.to = to;
  }

  private boolean assertRange() {
   assert from <= l.size();
   assert to <= l.size();
   assert to >= from;
   return true;
  }

  @Override
  public boolean add(final KEY_GENERIC_TYPE k) {
   l.add(to, k);
   to++;
   assert assertRange();
   return true;
  }

  @Override
  public void add(final int index, final KEY_GENERIC_TYPE k) {
   ensureIndex(index);
   l.add(from + index, k);
   to++;
   assert assertRange();
  }
```

```
 @Override
 public boolean addAll(final int index, final Collection<? extends KEY_GENERIC_CLASS> c) {
  ensureIndex(index);
  to += c.size();
  return l.addAll(from + index, c);
 }

 @Override
 public KEY_GENERIC_TYPE GET_KEY(final int index) {
  ensureRestrictedIndex(index);
  return l.GET_KEY(from + index);
 }

 @Override
 public KEY_GENERIC_TYPE REMOVE_KEY(final int index) {
  ensureRestrictedIndex(index);
  to--;
  return l.REMOVE_KEY(from + index);
 }

 @Override
 public KEY_GENERIC_TYPE set(final int index, final KEY_GENERIC_TYPE k) {
  ensureRestrictedIndex(index);
  return l.set(from + index, k);
 }

 @Override
 public int size() {
  return to - from;
 }

 @Override
 public void getElements(final int from, final KEY_TYPE[] a, final int offset, final int length) {
  ensureIndex(from);
  if (from + length > size())  throw new IndexOutOfBoundsException("End index (" + from + length + ") is greater
than list size (" + size() + ")");
  l.getElements(this.from + from, a, offset, length);
 }

 @Override
 public void removeElements(final int from, final int to) {
  ensureIndex(from);
  ensureIndex(to);
  l.removeElements(this.from + from, this.from + to);
  this.to -= (to - from);
  assert assertRange();
 }
```

```
 @Override
 public void addElements(int index, final KEY_GENERIC_TYPE a[], int offset, int length) {
  ensureIndex(index);
  l.addElements(this.from + index, a, offset, length);
  this.to += length;
  assert assertRange();
 }

 @Override
 public KEY_LIST_ITERATOR KEY_GENERIC listIterator(final int index) {
  ensureIndex(index);

  return new KEY_LIST_ITERATOR KEY_GENERIC() {
    int pos = index, last = -1;

    @Override
    public boolean hasNext() { return pos < size(); }
    @Override
    public boolean hasPrevious() { return pos > 0; }
    @Override
    public KEY_GENERIC_TYPE NEXT_KEY() { if (! hasNext()) throw new NoSuchElementException(); return
l.GET_KEY(from + (last = pos++)); }
    @Override
    public KEY_GENERIC_TYPE PREV_KEY() { if (! hasPrevious()) throw new NoSuchElementException(); return
l.GET_KEY(from + (last = --pos)); }
    @Override
    public int nextIndex() { return pos; }
    @Override
    public int previousIndex() { return pos - 1; }
    @Override
    public void add(KEY_GENERIC_TYPE k) {
     if (last == -1) throw new IllegalStateException();
     SUBLIST.this.add(pos++, k);
     last = -1;
     assert assertRange();
    }
    @Override
    public void set(KEY_GENERIC_TYPE k) {
     if (last == -1) throw new IllegalStateException();
     SUBLIST.this.set(last, k);
    }
    @Override
    public void remove() {
     if (last == -1) throw new IllegalStateException();
     SUBLIST.this.REMOVE_KEY(last);
     /* If the last operation was a next(), we are removing an element *before* us, and we must decrease pos
correspondingly. */
```

```
      if (last < pos) pos--;
      last = -1;
      assert assertRange();
     }
    };
  }

  @Override
  public LIST KEY_GENERIC subList(final int from, final int to) {
   ensureIndex(from);
   ensureIndex(to);
   if (from > to) throw new IllegalArgumentException("Start index (" + from + ") is greater than end index (" + to +
")");

   return new SUBLIST KEY_GENERIC_DIAMOND(this, from, to);
  }

#if KEYS_PRIMITIVE

  @Override
  public boolean rem(final KEY_TYPE k) {
   int index = indexOf(k);
   if (index == -1) return false;
   to--;
   l.REMOVE_KEY(from + index);
   assert assertRange();
   return true;
  }

  @Override
  public boolean addAll(final int index, final COLLECTION c) {
   ensureIndex(index);
   return super.addAll(index, c);
  }

  @Override
  public boolean addAll(final int index, final LIST l) {
   ensureIndex(index);
   return super.addAll(index, l);
  }
#endif

 }

}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-

a775574/drv/AbstractList.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import java.util.Collection;
import java.util.Set;

/** A class providing static methods and objects that do useful things with type-specific sets.
*
* @see java.util.Collections
*/

public final class SETS {

 private SETS() {}

 /** An immutable class representing the empty set and implementing a type-specific set interface.
  *
  * <p>This class may be useful to implement your own in case you subclass
  * a type-specific set.
  */

 public static class EmptySet KEY_GENERIC extends COLLECTIONS.EmptyCollection KEY_GENERIC
implements SET KEY_GENERIC, java.io.Serializable, Cloneable {
  private static final long serialVersionUID = -7046029254386353129L;

  protected EmptySet() {}

  @Override
  public boolean remove(KEY_TYPE ok) { throw new UnsupportedOperationException(); }
```

```
  @Override
  public Object clone() { return EMPTY_SET; }

  @Override
  @SuppressWarnings("rawtypes")
  public boolean equals(final Object o) { return o instanceof Set && ((Set)o).isEmpty(); }

#if KEYS_PRIMITIVE
  @Deprecated
  @Override
  public boolean rem(final KEY_TYPE k) { return super.rem(k); }
#endif

  private Object readResolve() { return EMPTY_SET; }
  }


 /** An empty set (immutable). It is serializable and cloneable.
  */
 SUPPRESS_WARNINGS_KEY_RAWTYPES
 public static final EmptySet EMPTY_SET = new EmptySet();

#if KEYS_REFERENCE
 /** Returns an empty set (immutable). It is serializable and cloneable.
  *
  * <p>This method provides a typesafe access to {@link #EMPTY_SET}.
  * @return an empty set (immutable).
  */
 @SuppressWarnings("unchecked")
 public static KEY_GENERIC SET KEY_GENERIC emptySet() {
  return EMPTY_SET;
  }
#endif


 /** An immutable class representing a type-specific singleton set.
  *
  * <p>This class may be useful to implement your own in case you subclass
  * a type-specific set.
  */

 public static class Singleton KEY_GENERIC extends ABSTRACT_SET KEY_GENERIC implements
java.io.Serializable, Cloneable {

  private static final long serialVersionUID = -7046029254386353129L;

  protected final KEY_GENERIC_TYPE element;
```

```
  protected Singleton(final KEY_GENERIC_TYPE element) {
   this.element = element;
  }

  @Override
  public boolean contains(final KEY_TYPE k) { return KEY_EQUALS(k, element); }

  @Override
  public boolean remove(final KEY_TYPE k) { throw new UnsupportedOperationException(); }

  @Override
  public KEY_LIST_ITERATOR KEY_GENERIC iterator() { return ITERATORS.singleton(element); }

  @Override
  public int size() { return 1; }

  @Override
  public boolean addAll(final Collection<? extends KEY_GENERIC_CLASS> c) { throw new
UnsupportedOperationException(); }

  @Override
  public boolean removeAll(final Collection<?> c) { throw new UnsupportedOperationException(); }

  @Override
  public boolean retainAll(final Collection<?> c) { throw new UnsupportedOperationException(); }

#if KEYS_PRIMITIVE

  @Override
  public boolean addAll(final COLLECTION c) { throw new UnsupportedOperationException(); }

  @Override
  public boolean removeAll(final COLLECTION c) { throw new UnsupportedOperationException(); }

  @Override
  public boolean retainAll(final COLLECTION c) { throw new UnsupportedOperationException(); }

#endif
  @Override
  public Object clone() { return this; }
  }

  /** Returns a type-specific immutable set containing only the specified element. The returned set is serializable and
cloneable.
   *
   * @param element the only element of the returned set.
   * @return a type-specific immutable set containing just {@code element}.
```

```
 */

 public static KEY_GENERIC SET KEY_GENERIC singleton(final KEY_GENERIC_TYPE element) {
  return new Singleton KEY_GENERIC_DIAMOND(element);
 }

#if KEYS_PRIMITIVE

 /** Returns a type-specific immutable set containing only the specified element. The returned set is serializable and
cloneable.
  *
  * @param element the only element of the returned set.
  * @return a type-specific immutable set containing just {@code element}.
  */

 public static KEY_GENERIC SET KEY_GENERIC singleton(final KEY_GENERIC_CLASS element) {
  return new Singleton KEY_GENERIC_DIAMOND(KEY_CLASS2TYPE(element));
 }

#endif

 /** A synchronized wrapper class for sets. */

 public static class SynchronizedSet KEY_GENERIC extends COLLECTIONS.SynchronizedCollection
KEY_GENERIC implements SET KEY_GENERIC, java.io.Serializable {

  private static final long serialVersionUID = -7046029254386353129L;

  protected SynchronizedSet(final SET KEY_GENERIC s, final Object sync) {
   super(s, sync);
  }

  protected SynchronizedSet(final SET KEY_GENERIC s) {
   super(s);
  }

  @Override
  public boolean remove(final KEY_TYPE k) { synchronized(sync) { return collection.REMOVE(k); } }

#if KEYS_PRIMITIVE
  @Deprecated
  @Override
  public boolean rem(final KEY_TYPE k) { return super.rem(k); }
#endif
 }


 /** Returns a synchronized type-specific set backed by the given type-specific set.
```

```
 *
 * @param s the set to be wrapped in a synchronized set.
 * @return a synchronized view of the specified set.
 * @see java.util.Collections#synchronizedSet(Set)
 */
 public static KEY_GENERIC SET KEY_GENERIC synchronize(final SET KEY_GENERIC s) { return new SynchronizedSet KEY_GENERIC_DIAMOND(s); }

 /** Returns a synchronized type-specific set backed by the given type-specific set, using an assigned object to synchronize.
 *
 * @param s the set to be wrapped in a synchronized set.
 * @param sync an object that will be used to synchronize the access to the set.
 * @return a synchronized view of the specified set.
 * @see java.util.Collections#synchronizedSet(Set)
 */

 public static KEY_GENERIC SET KEY_GENERIC synchronize(final SET KEY_GENERIC s, final Object sync) { return new SynchronizedSet KEY_GENERIC_DIAMOND(s, sync); }



 /** An unmodifiable wrapper class for sets. */

 public static class UnmodifiableSet KEY_GENERIC extends COLLECTIONS.UnmodifiableCollection KEY_GENERIC implements SET KEY_GENERIC, java.io.Serializable {

  private static final long serialVersionUID = -7046029254386353129L;

  protected UnmodifiableSet(final SET KEY_GENERIC s) {
   super(s);
  }

  @Override
  public boolean remove(final KEY_TYPE k) { throw new UnsupportedOperationException(); }

  @Override
  public boolean equals(final Object o) { if (o == this) return true; return collection.equals(o); }

  @Override
  public int hashCode() { return collection.hashCode(); }

#if KEYS_PRIMITIVE
  @Deprecated
  @Override
  public boolean rem(final KEY_TYPE k) { return super.rem(k); }
#endif
 }
```

```
 /** Returns an unmodifiable type-specific set backed by the given type-specific set.
  *
  * @param s the set to be wrapped in an unmodifiable set.
  * @return an unmodifiable view of the specified set.
  * @see java.util.Collections#unmodifiableSet(Set)
  */
 public static KEY_GENERIC SET KEY_GENERIC unmodifiable(final SET KEY_GENERIC s) { return new
UnmodifiableSet KEY_GENERIC_DIAMOND(s); }



#ifdef TEST

 private static KEY_TYPE genKey() {
#if KEY_CLASS_Byte || KEY_CLASS_Short || KEY_CLASS_Character
  return (KEY_TYPE)(r.nextInt());
#elif KEYS_PRIMITIVE
  return r.NEXT_KEY();
#elif KEY_CLASS_Object
  return Integer.toBinaryString(r.nextInt());
#else
  return new java.io.Serializable() {};
#endif
 }


 private static void test() {
  int n = 100;
  int c;
  KEY_TYPE k = genKey();
  Singleton m = new Singleton(k);
  Set t = java.util.Collections.singleton(KEY2OBJ(k));

  long ms;
  boolean mThrowsIllegal, tThrowsIllegal, mThrowsNoElement, tThrowsNoElement, mThrowsIndex, tThrowsIndex,
mThrowsUnsupp, tThrowsUnsupp;
  boolean rt = false, rm = false;


  /* Now we check that m and t are equal. */
  if (!m.equals(t) || ! t.equals(m)) System.err.println("m: " + m + " t: " + t);

  ensure(m.equals(t), "Error (" + seed + "): ! m.equals(t) at start");
  ensure(t.equals(m), "Error (" + seed + "): ! t.equals(m) at start");

  /* Now we check that m actually holds that data. */
```

```
  for(java.util.Iterator i=t.iterator(); i.hasNext();) {
   ensure(m.contains(i.next()), "Error (" + seed + "): m and t differ on an entry after insertion (iterating on t)");
  }

  /* Now we check that m actually holds that data, but iterating on m. */
  for(java.util.Iterator i=m.iterator(); i.hasNext();) {
   ensure(t.contains(i.next()), "Error (" + seed + "): m and t differ on an entry after insertion (iterating on m)");
  }

  /* Now we check that inquiries about random data give the same answer in m and t. For
     m we use the polymorphic method. */

  for(int i=0; i<n;  i++) {
   KEY_TYPE T = genKey();

   mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal =
mThrowsUnsupp = tThrowsUnsupp = false;

   try {
    m.contains(T);
   }
   catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
   catch (IllegalArgumentException e) { mThrowsIllegal = true; }
   catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }

   try {
    t.contains(KEY2OBJ(T));
   }
   catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
   catch (IllegalArgumentException e) { tThrowsIllegal = true; }
   catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }

   ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + seed + "): contains() divergence in
java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ") " + m);
   ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + seed + "): contains() divergence in IllegalArgumentException
for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
   ensure(mThrowsIndex == tThrowsIndex, "Error (" + seed + "): contains() divergence in
IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
   if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex) ensure(m.contains(KEY2OBJ(T)) ==
t.contains(KEY2OBJ(T)), "Error (" + seed + "): divergence in keys between t and m (polymorphic method) " + m);
  }

  /* Again, we check that inquiries about random data give the same answer in m and t, but
     for m we use the standard method. */

  for(int i=0; i<n;  i++) {
   KEY_TYPE T = genKey();
```

```
  mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal =
mThrowsUnsupp = tThrowsUnsupp = false;

  try {
   m.contains(KEY2OBJ(T));
  }
  catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }
  catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }

  try {
   t.contains(KEY2OBJ(T));
  }
  catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
  catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + seed + "): contains() divergence in
java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ") " + m);
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + seed + "): contains() divergence in IllegalArgumentException
for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
  ensure(mThrowsIndex == tThrowsIndex, "Error (" + seed + "): contains() divergence in
IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
  ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + seed + "): contains() divergence in
UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);
  if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex && !mThrowsUnsupp)
ensure(m.contains(KEY2OBJ(T)) ==  t.contains(KEY2OBJ(T)), "Error (" + seed + "): divergence between t and m
(standard method) " + m);
 }

 /* Now we add and remove random data in m and t, checking that the result is the same. */

 for(int i=0; i<20*n;  i++) {
  KEY_TYPE T = genKey();

  mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal =
mThrowsUnsupp = tThrowsUnsupp = false;

  try {
   rm = m.add(KEY2OBJ(T));
  }
  catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }
  catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }
```

```
  try {
   rt = t.add(KEY2OBJ(T));
  }
  catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
  catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + seed + "): add() divergence in
java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ") " + m);
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + seed + "): add() divergence in IllegalArgumentException for "
+ T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
  ensure(mThrowsIndex == tThrowsIndex, "Error (" + seed + "): add() divergence in IndexOutOfBoundsException
for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
  ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + seed + "): add() divergence in
UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);
  if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex && !mThrowsUnsupp) ensure(rm == rt, "Error
(" + seed + "): divergence in add() between t and m " + m);

  T = genKey();

  mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal =
mThrowsUnsupp = tThrowsUnsupp = false;

  try {
   rm = m.remove(KEY2OBJ(T));
  }
  catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }
  catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }

  try {
   rt = t.remove(KEY2OBJ(T));
  }
  catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
  catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }

  if (! KEY_EQUALS(T, k) && mThrowsUnsupp && ! tThrowsUnsupp) mThrowsUnsupp = false; // Stupid bug in
Collections.singleton()

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + seed + "): remove() divergence in
java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ") " + m);
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + seed + "): remove() divergence in IllegalArgumentException
for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
  ensure(mThrowsIndex == tThrowsIndex, "Error (" + seed + "): remove() divergence in
```

```
IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
  ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + seed + "): remove() divergence in
UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);
  if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex && !mThrowsUnsupp) ensure(rm == rt, "Error
(" + seed + "): divergence in remove() between t and m " + m);
 }

 ensure(m.equals(t), "Error (" + seed + "): ! m.equals(t) after removal " + m);
 ensure(t.equals(m), "Error (" + seed + "): ! t.equals(m) after removal " + m);

 /* Now we add and remove random collections in m and t, checking that the result is the same. */

 for(int i=0; i<20*n;  i++) {
  KEY_TYPE T = genKey();

  mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal =
mThrowsUnsupp = tThrowsUnsupp = false;

  try {
   rm = m.addAll(java.util.Collections.singleton(KEY2OBJ(T)));
  }
  catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }
  catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }

  try {
   rt = t.addAll(java.util.Collections.singleton(KEY2OBJ(T)));
  }
  catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
  catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + seed + "): addAll() divergence in
java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ") " + m);
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + seed + "): addAll() divergence in IllegalArgumentException
for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
  ensure(mThrowsIndex == tThrowsIndex, "Error (" + seed + "): addAll() divergence in IndexOutOfBoundsException
for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
  ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + seed + "): addAll() divergence in
UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);
  if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex && !mThrowsUnsupp) ensure(rm == rt, "Error
(" + seed + "): divergence in addAll() between t and m " + m);

  T = genKey();

  mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal =
```

```
mThrowsUnsupp = tThrowsUnsupp = false;

  try {
   rm = m.removeAll(java.util.Collections.singleton(KEY2OBJ(T)));
  }
  catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }
  catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }

  try {
   rt = t.removeAll(java.util.Collections.singleton(KEY2OBJ(T)));
  }
  catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
  catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }

  if (! KEY_EQUALS(T, k) && mThrowsUnsupp && ! tThrowsUnsupp) mThrowsUnsupp = false; // Stupid bug in
Collections.singleton()

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + seed + "): removeAll() divergence in
java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ") " + m);
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + seed + "): removeAll() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
  ensure(mThrowsIndex == tThrowsIndex, "Error (" + seed + "): removeAll() divergence in
IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
  ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + seed + "): removeAll() divergence in
UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);
  if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex && !mThrowsUnsupp) ensure(rm == rt, "Error
(" + seed + "): divergence in removeAll() between t and m " + m);
  }

  ensure(m.equals(t), "Error (" + seed + "): ! m.equals(t) after set removal " + m);
  ensure(t.equals(m), "Error (" + seed + "): ! t.equals(m) after set removal " + m);

  /* Now we check that m actually holds the same data. */

  for(java.util.Iterator i=t.iterator(); i.hasNext();) {
   ensure(m.contains(i.next()), "Error (" + seed + "): m and t differ on an entry after removal (iterating on t)");
  }

  /* Now we check that m actually holds that data, but iterating on m. */

  for(java.util.Iterator i=m.iterator(); i.hasNext();) {
   ensure(t.contains(i.next()), "Error (" + seed + "): m and t differ on an entry after removal (iterating on m)");
  }
```

```
 if (m instanceof Singleton) {
  ensure(m.equals(((Singleton)m).clone()), "Error (" + seed + "): m does not equal m.clone()");
  ensure(((Singleton)m).clone().equals(m), "Error (" + seed + "): m.clone() does not equal m");
 }

 int h = m.hashCode();

 /* Now we save and read m. */

 SET m2 = null;

 try {
  java.io.File ff = new java.io.File("it.unimi.dsi.fastutil.test");
  java.io.OutputStream os = new java.io.FileOutputStream(ff);
  java.io.ObjectOutputStream oos = new java.io.ObjectOutputStream(os);

  oos.writeObject(m);
  oos.close();

  java.io.InputStream is = new java.io.FileInputStream(ff);
  java.io.ObjectInputStream ois = new java.io.ObjectInputStream(is);

  m2 = (SET)ois.readObject();
  ois.close();
  ff.delete();
 }
 catch(Exception e) {
  e.printStackTrace();
  System.exit(1);
 }

#if ! KEY_CLASS_Reference

 ensure(m2.hashCode() == h, "Error (" + seed + "): hashCode() changed after save/read");

 /* Now we check that m2 actually holds that data. */

 ensure(m2.equals(t), "Error (" + seed + "): ! m2.equals(t) after save/read");
 ensure(t.equals(m2), "Error (" + seed + "): ! t.equals(m2) after save/read");
#endif

 System.out.println("Test OK");
 return;
 }

 private static long seed = System.currentTimeMillis();
 private static java.util.Random r = new java.util.Random(seed);
```

```
 private static java.text.NumberFormat format = new java.text.DecimalFormat("#,###.00");
 private static java.text.FieldPosition fp = new java.text.FieldPosition(0);

 private static String format(double d) {
  StringBuffer s = new StringBuffer();
  return format.format(d, s, fp).toString();
 }

 private static void fatal(String msg) {
  System.out.println(msg);
  System.exit(1);
 }

 private static void ensure(boolean cond, String msg) {
  if (cond) return;
  fatal(msg);
 }

 /** This method expects as first argument a lower-cased type (e.g., "int"),
  * and as second optional argument a seed. */

 public static void main(String arg[]) throws Exception {
  if (arg.length > 1) r = new java.util.Random(seed = Long.parseLong(arg[1]));

  try {
   test();
  } catch(Throwable e) {
   e.printStackTrace(System.err);
   System.err.println("seed: " + seed);
  }
 }

#endif

}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/Sets.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2004-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
```

```
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/



/** Loads elements from a given fast buffered reader, storing them in a given array fragment.
*
* @param reader a buffered reader.
* @param array an array which will be filled with data from {@code reader}.
* @param offset the index of the first element of {@code array} to be filled.
* @param length the number of elements of {@code array} to be filled.
* @return the number of elements actually read from {@code reader} (it might be less than {@code length} if
{@code reader} ends).
*/
public static int LOAD_KEYS(final BufferedReader reader, final KEY_TYPE[] array, final int offset, final int
length) throws IOException {
 PACKAGE.ARRAYS.ensureOffsetLength(array, offset, length);
 int i = 0;
 String s;
 try {
  for(i = 0; i < length; i++)
   if ((s = reader.readLine()) != null) array[i + offset] = KEY_CLASS.PARSE_KEY(s.trim());
   else break;
 }
 catch(EOFException itsOk) {}
 return i;
}

/** Loads elements from a given buffered reader, storing them in a given array.
*
* @param reader a buffered reader.
* @param array an array which will be filled with data from {@code reader}.
* @return the number of elements actually read from {@code reader} (it might be less than the array length if
{@code reader} ends).
*/
public static int LOAD_KEYS(final BufferedReader reader, final KEY_TYPE[] array) throws IOException {
 return LOAD_KEYS(reader, array, 0, array.length);
}

/** Loads elements from a file given by a {@link File} object, storing them in a given array fragment.
*
* @param file a file.
```

```
* @param array an array which will be filled with data from the specified file.
* @param offset the index of the first element of {@code array} to be filled.
* @param length the number of elements of {@code array} to be filled.
* @return the number of elements actually read from the given file (it might be less than {@code length} if the file is too short).
*/
public static int LOAD_KEYS(final File file, final KEY_TYPE[] array, final int offset, final int length) throws IOException {
 final BufferedReader reader = new BufferedReader(new FileReader(file));
 final int result = LOAD_KEYS(reader, array, offset, length);
 reader.close();

 return result;
}

/** Loads elements from a file given by a filename, storing them in a given array fragment.
*
* @param filename a filename.
* @param array an array which will be filled with data from the specified file.
* @param offset the index of the first element of {@code array} to be filled.
* @param length the number of elements of {@code array} to be filled.
* @return the number of elements actually read from the given file (it might be less than {@code length} if the file is too short).
*/
public static int LOAD_KEYS(final CharSequence filename, final KEY_TYPE[] array, final int offset, final int length) throws IOException {
 return LOAD_KEYS(new File(filename.toString()), array, offset, length);
}

/** Loads elements from a file given by a {@link File} object, storing them in a given array.
*
* @param file a file.
* @param array an array which will be filled with data from the specified file.
* @return the number of elements actually read from the given file (it might be less than the array length if the file is too short).
*/
public static int LOAD_KEYS(final File file, final KEY_TYPE[] array) throws IOException {
 return LOAD_KEYS(file, array, 0, array.length);
}

/** Loads elements from a file given by a filename, storing them in a given array.
*
* @param filename a filename.
* @param array an array which will be filled with data from the specified file.
* @return the number of elements actually read from the given file (it might be less than the array length if the file is too short).
*/
public static int LOAD_KEYS(final CharSequence filename, final KEY_TYPE[] array) throws IOException {
```

```
 return LOAD_KEYS(filename, array, 0, array.length);
}

/** Stores an array fragment to a given print stream.
*
* @param array an array whose elements will be written to {@code stream}.
* @param offset the index of the first element of {@code array} to be written.
* @param length the number of elements of {@code array} to be written.
* @param stream a print stream.
*/
public static void STORE_KEYS(final KEY_TYPE array[], final int offset, final int length, final PrintStream
stream) {
 PACKAGE.ARRAYS.ensureOffsetLength(array, offset, length);
 for(int i = 0; i < length; i++) stream.println(array[offset + i]);
}

/** Stores an array to a given print stream.
*
* @param array an array whose elements will be written to {@code stream}.
* @param stream a print stream.
*/
public static void STORE_KEYS(final KEY_TYPE array[], final PrintStream stream) {
 STORE_KEYS(array, 0, array.length, stream);
}

/** Stores an array fragment to a file given by a {@link File} object.
*
* @param array an array whose elements will be written to {@code filename}.
* @param offset the index of the first element of {@code array} to be written.
* @param length the number of elements of {@code array} to be written.
* @param file a file.
*/
public static void STORE_KEYS(final KEY_TYPE array[], final int offset, final int length, final File file) throws
IOException {
 final PrintStream stream = new PrintStream(new FastBufferedOutputStream(new FileOutputStream(file)));
 STORE_KEYS(array, offset, length, stream);
 stream.close();
}

/** Stores an array fragment to a file given by a pathname.
*
* @param array an array whose elements will be written to {@code filename}.
* @param offset the index of the first element of {@code array} to be written.
* @param length the number of elements of {@code array} to be written.
* @param filename a filename.
*/
public static void STORE_KEYS(final KEY_TYPE array[], final int offset, final int length, final CharSequence
filename) throws IOException {
```

```
 STORE_KEYS(array, offset, length, new File(filename.toString()));
}

/** Stores an array to a file given by a {@link File} object.
*
* @param array an array whose elements will be written to {@code filename}.
* @param file a file.
*/
public static void STORE_KEYS(final KEY_TYPE array[], final File file) throws IOException {
 STORE_KEYS(array, 0, array.length, file);
}

/** Stores an array to a file given by a pathname.
*
* @param array an array whose elements will be written to {@code filename}.
* @param filename a filename.
*/
public static void STORE_KEYS(final KEY_TYPE array[], final CharSequence filename) throws IOException {
 STORE_KEYS(array, 0, array.length, filename);
}

/** Stores the element returned by an iterator to a given print stream.
*
* @param i an iterator whose output will be written to {@code stream}.
* @param stream a print stream.
*/
public static void STORE_KEYS(final KEY_ITERATOR i, final PrintStream stream) {
 while(i.hasNext()) stream.println(i.NEXT_KEY());
}

/** Stores the element returned by an iterator to a file given by a {@link File} object.
*
* @param i an iterator whose output will be written to {@code filename}.
* @param file a file.
*/
public static void STORE_KEYS(final KEY_ITERATOR i, final File file) throws IOException {
 final PrintStream stream = new PrintStream(new FastBufferedOutputStream(new FileOutputStream(file)));
 STORE_KEYS(i, stream);
 stream.close();
}

/** Stores the element returned by an iterator to a file given by a pathname.
*
* @param i an iterator whose output will be written to {@code filename}.
* @param filename a filename.
*/
public static void STORE_KEYS(final KEY_ITERATOR i, final CharSequence filename) throws IOException {
 STORE_KEYS(i, new File(filename.toString()));
```

```
}


/** Loads elements from a given fast buffered reader, storing them in a given big-array fragment.
*
* @param reader a buffered reader.
* @param array a big array which will be filled with data from {@code reader}.
* @param offset the index of the first element of {@code array} to be filled.
* @param length the number of elements of {@code array} to be filled.
* @return the number of elements actually read from {@code reader} (it might be less than {@code length} if
{@code reader} ends).
*/
public static long LOAD_KEYS(final BufferedReader reader, final KEY_TYPE[][] array, final long offset, final
long length) throws IOException {
 PACKAGE.BIG_ARRAYS.ensureOffsetLength(array, offset, length);
 long c = 0;
 String s;
 try {
  for(int i = segment(offset); i < segment(offset + length + SEGMENT_MASK); i++) {
   final KEY_TYPE[] t = array[i];
   final int l = (int)Math.min(t.length, offset + length - start(i));
   for(int d = (int)Math.max(0, offset - start(i)); d < l; d++) {
    if ((s = reader.readLine()) != null) t[d] = KEY_CLASS.PARSE_KEY(s.trim());
    else return c;
    c++;
   }
  }
 }
 catch(EOFException itsOk) {}
 return c;
}

/** Loads elements from a given buffered reader, storing them in a given array.
*
* @param reader a buffered reader.
* @param array a big array which will be filled with data from {@code reader}.
* @return the number of elements actually read from {@code reader} (it might be less than the array length if
{@code reader} ends).
*/
public static long LOAD_KEYS(final BufferedReader reader, final KEY_TYPE[][] array) throws IOException {
 return LOAD_KEYS(reader, array, 0, PACKAGE.BIG_ARRAYS.length(array));
}

/** Loads elements from a file given by a {@link File} object, storing them in a given big-array fragment.
*
* @param file a file.
* @param array a big array which will be filled with data from the specified file.
* @param offset the index of the first element of {@code array} to be filled.
```

```
* @param length the number of elements of {@code array} to be filled.
* @return the number of elements actually read from the given file (it might be less than {@code length} if the file is too short).
*/
public static long LOAD_KEYS(final File file, final KEY_TYPE[][] array, final long offset, final long length) throws IOException {
 final BufferedReader reader = new BufferedReader(new FileReader(file));
 final long result = LOAD_KEYS(reader, array, offset, length);
 reader.close();

 return result;
}

/** Loads elements from a file given by a filename, storing them in a given big-array fragment.
*
* @param filename a filename.
* @param array a big array which will be filled with data from the specified file.
* @param offset the index of the first element of {@code array} to be filled.
* @param length the number of elements of {@code array} to be filled.
* @return the number of elements actually read from the given file (it might be less than {@code length} if the file is too short).
*/
public static long LOAD_KEYS(final CharSequence filename, final KEY_TYPE[][] array, final long offset, final long length) throws IOException {
 return LOAD_KEYS(new File(filename.toString()), array, offset, length);
}

/** Loads elements from a file given by a {@link File} object, storing them in a given array.
*
* @param file a file.
* @param array a big array which will be filled with data from the specified file.
* @return the number of elements actually read from the given file (it might be less than the array length if the file is too short).
*/
public static long LOAD_KEYS(final File file, final KEY_TYPE[][] array) throws IOException {
 return LOAD_KEYS(file, array, 0, PACKAGE.BIG_ARRAYS.length(array));
}

/** Loads elements from a file given by a filename, storing them in a given array.
*
* @param filename a filename.
* @param array a big array which will be filled with data from the specified file.
* @return the number of elements actually read from the given file (it might be less than the array length if the file is too short).
*/
public static long LOAD_KEYS(final CharSequence filename, final KEY_TYPE[][] array) throws IOException {
 return LOAD_KEYS(filename, array, 0, PACKAGE.BIG_ARRAYS.length(array));
}
```

```
/** Stores a big-array fragment to a given print stream.
*
* @param array a big array whose elements will be written to {@code stream}.
* @param offset the index of the first element of {@code array} to be written.
* @param length the number of elements of {@code array} to be written.
* @param stream a print stream.
*/
public static void STORE_KEYS(final KEY_TYPE array[][], final long offset, final long length, final PrintStream
stream) {
 PACKAGE.BIG_ARRAYS.ensureOffsetLength(array, offset, length);

 for(int i = segment(offset); i < segment(offset + length + SEGMENT_MASK); i++) {
  final KEY_TYPE[] t = array[i];
  final int l = (int)Math.min(t.length, offset + length - start(i));
  for(int d = (int)Math.max(0, offset - start(i)); d < l; d++) stream.println(t[d]);
 }
}

/** Stores a big array to a given print stream.
*
* @param array a big array whose elements will be written to {@code stream}.
* @param stream a print stream.
*/
public static void STORE_KEYS(final KEY_TYPE array[][], final PrintStream stream) {
 STORE_KEYS(array, 0, PACKAGE.BIG_ARRAYS.length(array), stream);
}

/** Stores a big-array fragment to a file given by a {@link File} object.
*
* @param array a big array whose elements will be written to {@code filename}.
* @param offset the index of the first element of {@code array} to be written.
* @param length the number of elements of {@code array} to be written.
* @param file a file.
*/
public static void STORE_KEYS(final KEY_TYPE array[][], final long offset, final long length, final File file)
throws IOException {
 final PrintStream stream = new PrintStream(new FastBufferedOutputStream(new FileOutputStream(file)));
 STORE_KEYS(array, offset, length, stream);
 stream.close();
}

/** Stores a big-array fragment to a file given by a pathname.
*
* @param array a big array whose elements will be written to {@code filename}.
* @param offset the index of the first element of {@code array} to be written.
* @param length the number of elements of {@code array} to be written.
* @param filename a filename.
```

```
*/
public static void STORE_KEYS(final KEY_TYPE array[][], final long offset, final long length, final
CharSequence filename) throws IOException {
 STORE_KEYS(array, offset, length, new File(filename.toString()));
}

/** Stores a big array to a file given by a {@link File} object.
*
* @param array a big array whose elements will be written to {@code filename}.
* @param file a file.
*/
public static void STORE_KEYS(final KEY_TYPE array[][], final File file) throws IOException {
 STORE_KEYS(array, 0, PACKAGE.BIG_ARRAYS.length(array), file);
}

/** Stores a big array to a file given by a pathname.
*
* @param array a big array whose elements will be written to {@code filename}.
* @param filename a filename.
*/
public static void STORE_KEYS(final KEY_TYPE array[][], final CharSequence filename) throws IOException {
 STORE_KEYS(array, 0, PACKAGE.BIG_ARRAYS.length(array), filename);
}


/** A wrapper that exhibits the content of a reader as a type-specific iterator. */

private static final class KEY_READER_WRAPPER implements KEY_ITERATOR {
 private final BufferedReader reader;
 private boolean toAdvance = true;
 private String s;
 private KEY_TYPE next;

 public KEY_READER_WRAPPER(final BufferedReader reader) {
  this.reader = reader;
 }

 @Override
 public boolean hasNext() {
  if (! toAdvance) return s != null;

  toAdvance = false;

  try {
   s = reader.readLine();
  }
  catch(EOFException itsOk) {}
  catch(IOException rethrow) { throw new RuntimeException(rethrow); }
```

```
 if (s == null) return false;

 next = KEY_CLASS.PARSE_KEY(s.trim());
 return true;
 }

 @Override
 public KEY_TYPE NEXT_KEY() {
 if (! hasNext()) throw new NoSuchElementException();
 toAdvance = true;
 return next;
 }
}



/** Wraps the given buffered reader into an iterator.
*
* @param reader a buffered reader.
*/
public static KEY_ITERATOR AS_KEY_ITERATOR(final BufferedReader reader) {
 return new KEY_READER_WRAPPER(reader);
}

/** Wraps a file given by a {@link File} object into an iterator.
*
* @param file a file.
*/
public static KEY_ITERATOR AS_KEY_ITERATOR(final File file) throws IOException {
 return new KEY_READER_WRAPPER(new BufferedReader(new FileReader(file)));
}

/** Wraps a file given by a pathname into an iterator.
*
* @param filename a filename.
*/
public static KEY_ITERATOR AS_KEY_ITERATOR(final CharSequence filename) throws IOException {
 return AS_KEY_ITERATOR(new File(filename.toString()));
}

/** Wraps a file given by a {@link File} object into an iterable object.
*
* @param file a file.
*/
public static KEY_ITERABLE AS_KEY_ITERABLE(final File file) {
 return () -> {
  try { return AS_KEY_ITERATOR(file); }
```

```
  catch(IOException e) { throw new RuntimeException(e); }
 };
}

/** Wraps a file given by a pathname into an iterable object.
*
* @param filename a filename.
*/
public static KEY_ITERABLE AS_KEY_ITERABLE(final CharSequence filename) {
 return () -> {
  try { return AS_KEY_ITERATOR(filename); }
  catch(IOException e) { throw new RuntimeException(e); }
 };
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/TextIOFragment.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/objects/ObjectOpenCustomHashSetTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/StripedInt2IntOpenHashMapTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/SafeMath.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/IntOpenHashSetTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/Int2IntMapGenericAVLTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-

a775574/test/it/unimi/dsi/fastutil/ints/IntArrayFIFOQueueTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/doubles/DoubleOpenHashSetTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/Int2ObjectMapGenericTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/objects/Object2IntOpenHashMapTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/doubles/DoubleArraysTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/objects/ObjectHeapPriorityQueueTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/Int2IntMapGenericLinkedOpenHashTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/objects/ObjectBigArrayBigListTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/objects/AbstractObject2IntFunctionTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/io/FastBufferedOutputStreamTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/Int2IntLinkedOpenHashMapTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/objects/Reference2ReferenceArrayMapTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/Int2ObjectMapTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/objects/ObjectArraysTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/objects/Object2ObjectArrayMapTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/objects/ObjectArrayListTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/Int2ObjectMapGenericArrayTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/objects/ObjectOpenHashSetTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/IntListsTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/bytes/ByteArrayFrontCodedListTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/IntArrayPriorityQueueTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/IntArrayFrontCodedListTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/io/BinIOTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/AbstractIntCollectionTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-

a775574/test/it/unimi/dsi/fastutil/ints/IntSetsTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/Int2IntMapGenericOpenHashTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/longs/LongArrayFrontCodedListTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/bytes/ByteArraysTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/objects/ObjectAVLTreeSetTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/objects/ObjectArraySetTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/Int2ObjectMapGenericOpenHashTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/BigArraysTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/shorts/ShortArraysTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/IntOpenCustomHashSetTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/objects/ObjectRBTreeSetTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/doubles/DoubleBigArraysTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/Int2ObjectMapGenericLinkedOpenHashTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/objects/ReferenceArraySetTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/objects/ObjectArrayPriorityQueueTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/chars/CharArrayFrontCodedListTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/IntArraySetTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/Int2IntMapGenericDefaultTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/IntCollectionsTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/io/FastByteArrayOutputStreamTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/objects/ObjectBigArraysTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/IntHeapSemiIndirectPriorityQueueTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/io/FastMultiByteArrayInputStreamTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/io/InspectableFileCachedInputStreamTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-

a775574/test/it/unimi/dsi/fastutil/ints/IntLinkedOpenHashSetTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/IntSemiIndirectHeapsTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/Int2IntMapsTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/IntHeapPriorityQueueTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/objects/ObjectOpenHashBigSetTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/longs/LongArraysTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/io/TextIOTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/floats/FloatArraysTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/IntBigArrayBigListTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/AbstractInt2IntMapTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/Int2ObjectFunctionTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/Int2ObjectMapGenericAVLTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/Int2IntOpenHashMapTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ArraysTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/Int2ObjectMapGenericRBTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/longs/LongOpenHashBigSetTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/floats/FloatOpenHashSetTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/Int2IntMapGenericArrayTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/IntLinkedOpenCustomHashSetTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/Int2IntMapGenericRBTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/Int2IntOpenCustomHashMapTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/IntArraysTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/IntOpenHashBigSetTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/chars/CharArraysTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-

a775574/test/it/unimi/dsi/fastutil/shorts/ShortArrayFrontCodedListTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/Int2IntMapGenericTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/IntArrayIndirectPriorityQueueTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/IntBigArraysTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/HashCommonTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/io/FastBufferedInputStreamTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/objects/ObjectSetsTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/IntArrayListTest.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/test/it/unimi/dsi/fastutil/ints/Int2ObjectMapGenericDefaultTest.java
No license file was found, but licenses were detected in source scan.


```
package PACKAGE;

import java.util.Collection;
import java.util.Comparator;
import java.util.Iterator;
import java.util.SortedSet;
import java.util.NoSuchElementException;

/** A type-specific red-black tree set with a fast, small-footprint implementation.
*
* <p>The iterators provided by this class are type-specific {@link
* it.unimi.dsi.fastutil.BidirectionalIterator bidirectional iterators}.
```

```
* Moreover, the iterator returned by {@code iterator()} can be safely cast
* to a type-specific {@linkplain java.util.ListIterator list iterator}.
*/

public class RB_TREE_SET KEY_GENERIC extends ABSTRACT_SORTED_SET KEY_GENERIC implements
java.io.Serializable, Cloneable, SORTED_SET KEY_GENERIC {

 /** A reference to the root entry. */
 protected transient Entry KEY_GENERIC tree;

 /** Number of elements in this set. */
 protected int count;

 /** The entry of the first element of this set. */
 protected transient Entry KEY_GENERIC firstEntry;

 /** The entry of the last element of this set. */
 protected transient Entry KEY_GENERIC lastEntry;

 /** This set's comparator, as provided in the constructor. */
 protected Comparator<? super KEY_GENERIC_CLASS> storedComparator;

 /** This set's actual comparator; it may differ from {@link #storedComparator} because it is
  always a type-specific comparator, so it could be derived from the former by wrapping. */
 protected transient KEY_COMPARATOR KEY_SUPER_GENERIC actualComparator;

 private static final long serialVersionUID = -7046029254386353130L;

 {
  allocatePaths();
 }

 /** Creates a new empty tree set.
  */

 public RB_TREE_SET() {
  tree = null;
  count = 0;
 }

 /** Generates the comparator that will be actually used.
  *
  * <p>When a given {@link Comparator} is specified and stored in {@link
  * #storedComparator}, we must check whether it is type-specific.  If it is
  * so, we can used directly, and we store it in {@link #actualComparator}. Otherwise,
  * we adapt it using a helper static method.
  */
 private void setActualComparator() {
```

```
#if KEY_CLASS_Object
 actualComparator = storedComparator;
#else
 actualComparator = COMPARATORS.AS_KEY_COMPARATOR(storedComparator);
#endif
 }


 /** Creates a new empty tree set with the given comparator.
  *
  * @param c a {@link Comparator} (even better, a type-specific comparator).
  */

 public RB_TREE_SET(final Comparator<? super KEY_GENERIC_CLASS> c) {
  this();
  storedComparator = c;
  setActualComparator();
 }


 /** Creates a new tree set copying a given collection.
  *
  * @param c a collection to be copied into the new tree set.
  */

 public RB_TREE_SET(final Collection<? extends KEY_GENERIC_CLASS> c) {
  this();
  addAll(c);
 }

 /** Creates a new tree set copying a given sorted set (and its {@link Comparator}).
  *
  * @param s a {@link SortedSet} to be copied into the new tree set.
  */

 public RB_TREE_SET(final SortedSet<KEY_GENERIC_CLASS> s) {
  this(s.comparator());
  addAll(s);
 }

 /** Creates a new tree set copying a given type-specific collection.
  *
  * @param c a type-specific collection to be copied into the new tree set.
  */

 public RB_TREE_SET(final COLLECTION KEY_EXTENDS_GENERIC c) {
  this();
  addAll(c);
```

```
}

/** Creates a new tree set copying a given type-specific sorted set (and its {@link Comparator}).
 *
 * @param s a type-specific sorted set to be copied into the new tree set.
 */

public RB_TREE_SET(final SORTED_SET KEY_GENERIC s) {
 this(s.comparator());
 addAll(s);
}


/** Creates a new tree set using elements provided by a type-specific iterator.
 *
 * @param i a type-specific iterator whose elements will fill the set.
 */

public RB_TREE_SET(final STD_KEY_ITERATOR KEY_EXTENDS_GENERIC i) {
 while(i.hasNext()) add(i.NEXT_KEY());
}


#if KEYS_PRIMITIVE

/** Creates a new tree set using elements provided by an iterator.
 *
 * @param i an iterator whose elements will fill the set.
 */

SUPPRESS_WARNINGS_KEY_UNCHECKED
public RB_TREE_SET(final Iterator<?> i) {
 this(ITERATORS.AS_KEY_ITERATOR(i));
}

#endif

/** Creates a new tree set and fills it with the elements of a given array using a given {@link Comparator}.
 *
 * @param a an array whose elements will be used to fill the set.
 * @param offset the first element to use.
 * @param length the number of elements to use.
 * @param c a {@link Comparator} (even better, a type-specific comparator).
 */

public RB_TREE_SET(final KEY_GENERIC_TYPE[] a, final int offset, final int length, final Comparator<? super
KEY_GENERIC_CLASS> c) {
 this(c);
```

```
 ARRAYS.ensureOffsetLength(a, offset, length);
 for(int i = 0; i < length; i++) add(a[offset + i]);
}


/** Creates a new tree set and fills it with the elements of a given array.
 *
 * @param a an array whose elements will be used to fill the set.
 * @param offset the first element to use.
 * @param length the number of elements to use.
 */

public RB_TREE_SET(final KEY_GENERIC_TYPE[] a, final int offset, final int length) {
 this(a, offset, length, null);
}


/** Creates a new tree set copying the elements of an array.
 *
 * @param a an array to be copied into the new tree set.
 */

public RB_TREE_SET(final KEY_GENERIC_TYPE[] a) {
 this();
 int i = a.length;
 while(i-- != 0) add(a[i]);
}


/** Creates a new tree set copying the elements of an array using a given {@link Comparator}.
 *
 * @param a an array to be copied into the new tree set.
 * @param c a {@link Comparator} (even better, a type-specific comparator).
 */

public RB_TREE_SET(final KEY_GENERIC_TYPE[] a, final Comparator<? super KEY_GENERIC_CLASS> c)
{
 this(c);
 int i = a.length;
 while(i-- != 0) add(a[i]);
}




/*
 * The following methods implements some basic building blocks used by
 * all accessors.  They are (and should be maintained) identical to those used in RBTreeMap.drv.
```

```
 *
 * The add()/remove() code is derived from Ben Pfaff's GNU libavl
 * (http://www.msu.edu/~pfaffben/avl/). If you want to understand what's
 * going on, you should have a look at the literate code contained therein
 * first.
 */



/** Compares two keys in the right way.
 *
 * <p>This method uses the {@link #actualComparator} if it is non-{@code null}.
 * Otherwise, it resorts to primitive type comparisons or to {@link Comparable#compareTo(Object) compareTo()}.
 *
 * @param k1 the first key.
 * @param k2 the second key.
 * @return a number smaller than, equal to or greater than 0, as usual
 * (i.e., when k1 &lt; k2, k1 = k2 or k1 &gt; k2, respectively).
 */

SUPPRESS_WARNINGS_KEY_UNCHECKED
final int compare(final KEY_GENERIC_TYPE k1, final KEY_GENERIC_TYPE k2) {
 return actualComparator == null ? KEY_CMP(k1, k2) : actualComparator.compare(k1, k2);
}



/** Returns the entry corresponding to the given key, if it is in the tree; {@code null}, otherwise.
 *
 * @param k the key to search for.
 * @return the corresponding entry, or {@code null} if no entry with the given key exists.
 */

private Entry KEY_GENERIC findKey(final KEY_GENERIC_TYPE k) {
 Entry KEY_GENERIC e = tree;
 int cmp;

 while (e != null && (cmp = compare(k, e.key)) != 0)
  e = cmp < 0 ? e.left() : e.right();

 return e;
}

/** Locates a key.
 *
 * @param k a key.
 * @return the last entry on a search for the given key; this will be
 * the given key, if it present; otherwise, it will be either the smallest greater key or the greatest smaller key.
 */
```

```
final Entry KEY_GENERIC locateKey(final KEY_GENERIC_TYPE k) {
 Entry KEY_GENERIC e = tree, last = tree;
 int cmp = 0;

 while (e != null && (cmp = compare(k, e.key)) != 0) {
  last = e;
  e = cmp < 0 ? e.left() : e.right();
 }

 return cmp == 0 ? e : last;
}

/** This vector remembers the path and the direction followed during the
 *  current insertion. It suffices for about 2<sup>32</sup> entries. */
private transient boolean dirPath[];
private transient Entry KEY_GENERIC nodePath[];

SUPPRESS_WARNINGS_KEY_UNCHECKED_RAWTYPES
private void allocatePaths() {
 dirPath = new boolean[64];
 nodePath = new Entry[64];
}

@Override
public boolean add(final KEY_GENERIC_TYPE k) {
 int maxDepth = 0;

 if (tree == null) { // The case of the empty tree is treated separately.
  count++;
  tree = lastEntry = firstEntry = new Entry KEY_GENERIC_DIAMOND(k);
 }
 else {
  Entry KEY_GENERIC p = tree, e;
  int cmp, i = 0;

  while(true) {
   if ((cmp = compare(k, p.key)) == 0) {
    // We clean up the node path, or we could have stale references later.
    while(i-- != 0) nodePath[i] = null;
    return false;
   }

   nodePath[i] = p;

   if (dirPath[i++] = cmp > 0) {
    if (p.succ()) {
     count++;
```

```
   e = new Entry KEY_GENERIC_DIAMOND(k);

   if (p.right == null) lastEntry = e;

   e.left = p;
   e.right = p.right;

   p.right(e);

   break;
  }

  p = p.right;
 }
 else {
  if (p.pred()) {
   count++;
   e = new Entry KEY_GENERIC_DIAMOND(k);

   if (p.left == null) firstEntry = e;

   e.right = p;
   e.left = p.left;

   p.left(e);

   break;
  }

  p = p.left;
 }
}

maxDepth = i--;

while(i > 0 && ! nodePath[i].black()) {
 if (! dirPath[i - 1]) {
  Entry KEY_GENERIC y = nodePath[i - 1].right;

  if (! nodePath[i - 1].succ() && ! y.black()) {
   nodePath[i].black(true);
   y.black(true);
   nodePath[i - 1].black(false);
   i -= 2;
  }
  else {
   Entry KEY_GENERIC x;
```

```
 if (! dirPath[i]) y = nodePath[i];
 else {
  x = nodePath[i];
  y = x.right;
  x.right = y.left;
  y.left = x;
  nodePath[i - 1].left = y;

  if (y.pred()) {
   y.pred(false);
   x.succ(y);
  }
 }

 x = nodePath[i - 1];
 x.black(false);
 y.black(true);

 x.left = y.right;
 y.right = x;
 if (i < 2) tree = y;
 else {
  if (dirPath[i - 2]) nodePath[i - 2].right = y;
  else nodePath[i - 2].left = y;
 }

 if (y.succ()) {
  y.succ(false);
  x.pred(y);
 }
 break;
 }
}
else {
 Entry KEY_GENERIC y = nodePath[i - 1].left;

 if (! nodePath[i - 1].pred() && ! y.black()) {
  nodePath[i].black(true);
  y.black(true);
  nodePath[i - 1].black(false);
  i -= 2;
 }
 else {
  Entry KEY_GENERIC x;

  if (dirPath[i]) y = nodePath[i];
  else {
   x = nodePath[i];
```

```
      y = x.left;
      x.left = y.right;
      y.right = x;
      nodePath[i - 1].right = y;

      if (y.succ()) {
       y.succ(false);
       x.pred(y);
      }

     }

     x = nodePath[i - 1];
     x.black(false);
     y.black(true);

     x.right = y.left;
     y.left = x;
     if (i < 2) tree = y;
     else {
      if (dirPath[i - 2]) nodePath[i - 2].right = y;
      else nodePath[i - 2].left = y;
     }

     if (y.pred()){
      y.pred(false);
      x.succ(y);
     }

     break;
    }
   }
  }
 }
 tree.black(true);
 // We clean up the node path, or we could have stale references later.
 while(maxDepth-- != 0) nodePath[maxDepth] = null;
 return true;
}


SUPPRESS_WARNINGS_KEY_UNCHECKED
@Override
public boolean remove(final KEY_TYPE k) {
 if (tree == null) return false;

 Entry KEY_GENERIC p = tree;
 int cmp;
```

```
int i = 0;
final KEY_GENERIC_TYPE kk = KEY_GENERIC_CAST k;

while(true) {
 if ((cmp = compare(kk, p.key)) == 0) break;

 dirPath[i] = cmp > 0;
 nodePath[i] = p;

 if (dirPath[i++]) {
  if ((p = p.right()) == null) {
   // We clean up the node path, or we could have stale references later.
   while(i-- != 0) nodePath[i] = null;
   return false;
  }
 }
 else {
  if ((p = p.left()) == null) {
   // We clean up the node path, or we could have stale references later.
   while(i-- != 0) nodePath[i] = null;
   return false;
  }
 }
}

if (p.left == null) firstEntry = p.next();
if (p.right == null) lastEntry = p.prev();

if (p.succ()) {
 if (p.pred()) {
  if (i == 0) tree = p.left;
  else {
   if (dirPath[i - 1]) nodePath[i - 1].succ(p.right);
   else nodePath[i - 1].pred(p.left);
  }
 }
 else {
  p.prev().right = p.right;

  if (i == 0) tree = p.left;
  else {
   if (dirPath[i - 1]) nodePath[i - 1].right = p.left;
   else nodePath[i - 1].left = p.left;
  }
 }
}
else {
 boolean color;
```

```
Entry KEY_GENERIC r = p.right;

if (r.pred()) {
 r.left = p.left;
 r.pred(p.pred());
 if (! r.pred()) r.prev().right = r;
 if (i == 0) tree = r;
 else {
  if (dirPath[i - 1]) nodePath[i - 1].right = r;
  else nodePath[i - 1].left = r;
 }

 color = r.black();
 r.black(p.black());
 p.black(color);
 dirPath[i] = true;
 nodePath[i++] = r;
}
else {
 Entry KEY_GENERIC s;
 int j = i++;

 while(true) {
  dirPath[i] = false;
  nodePath[i++] = r;
  s = r.left;
  if (s.pred()) break;
  r = s;
 }

 dirPath[j] = true;
 nodePath[j] = s;

 if (s.succ()) r.pred(s);
 else r.left = s.right;

 s.left = p.left;

 if (! p.pred()) {
  p.prev().right =  s;
  s.pred(false);
 }

 s.right(p.right);

 color = s.black();
 s.black(p.black());
 p.black(color);
```

```
  if (j == 0) tree = s;
  else {
   if (dirPath[j - 1]) nodePath[j - 1].right = s;
   else nodePath[j - 1].left = s;
  }
 }
}

int maxDepth = i;

if (p.black()) {
 for(; i > 0; i--) {
  if (dirPath[i - 1] && ! nodePath[i - 1].succ() ||
    ! dirPath[i - 1] && ! nodePath[i - 1].pred()) {
   Entry KEY_GENERIC x = dirPath[i - 1] ? nodePath[i - 1].right : nodePath[i - 1].left;

   if (! x.black()) {
    x.black(true);
    break;
   }
  }

  if (! dirPath[i - 1]) {
   Entry KEY_GENERIC w = nodePath[i - 1].right;

   if (! w.black()) {
    w.black(true);
    nodePath[i - 1].black(false);

    nodePath[i - 1].right = w.left;
    w.left = nodePath[i - 1];

    if (i < 2) tree = w;
    else {
     if (dirPath[i - 2]) nodePath[i - 2].right = w;
     else nodePath[i - 2].left = w;
    }

    nodePath[i] = nodePath[i - 1];
    dirPath[i] = false;
    nodePath[i - 1] = w;
    if (maxDepth == i++) maxDepth++;

    w = nodePath[i - 1].right;
   }

   if ((w.pred() || w.left.black()) &&
```

```
  (w.succ() || w.right.black())) {
 w.black(false);
 }
 else {
 if (w.succ() || w.right.black()) {
  Entry KEY_GENERIC y = w.left;

  y.black (true);
  w.black(false);
  w.left = y.right;
  y.right = w;
  w = nodePath[i - 1].right = y;

  if (w.succ()) {
   w.succ(false);
   w.right.pred(w);
  }
 }

 w.black(nodePath[i - 1].black());
 nodePath[i - 1].black(true);
 w.right.black(true);

 nodePath[i - 1].right = w.left;
 w.left = nodePath[i - 1];

 if (i < 2) tree = w;
 else {
  if (dirPath[i - 2]) nodePath[i - 2].right = w;
  else nodePath[i - 2].left = w;
 }

 if (w.pred()) {
  w.pred(false);
  nodePath[i - 1].succ(w);
 }
 break;
 }
}
else {
 Entry KEY_GENERIC w = nodePath[i - 1].left;

 if (! w.black()) {
  w.black (true);
  nodePath[i - 1].black(false);

  nodePath[i - 1].left = w.right;
  w.right = nodePath[i - 1];
```

```
 if (i < 2) tree = w;
 else {
  if (dirPath[i - 2]) nodePath[i - 2].right = w;
  else nodePath[i - 2].left = w;
 }

 nodePath[i] = nodePath[i - 1];
 dirPath[i] = true;
 nodePath[i - 1] = w;
 if (maxDepth == i++) maxDepth++;

 w = nodePath[i - 1].left;
}

if ((w.pred() || w.left.black()) &&
  (w.succ() || w.right.black())) {
 w.black(false);
}
else {
 if (w.pred() || w.left.black()) {
  Entry KEY_GENERIC y = w.right;

  y.black(true);
  w.black (false);
  w.right = y.left;
  y.left = w;
  w = nodePath[i - 1].left = y;

  if (w.pred()) {
   w.pred(false);
   w.left.succ(w);
  }
 }

 w.black(nodePath[i - 1].black());
 nodePath[i - 1].black(true);
 w.left.black(true);

 nodePath[i - 1].left = w.right;
 w.right = nodePath[i - 1];

 if (i < 2) tree = w;
 else {
  if (dirPath[i - 2]) nodePath[i - 2].right = w;
  else nodePath[i - 2].left = w;
 }
```

```
      if (w.succ()) {
       w.succ(false);
       nodePath[i - 1].pred(w);
      }
      break;
     }
    }
   }

   if (tree != null) tree.black(true);
  }

  count--;
  // We clean up the node path, or we could have stale references later.
  while(maxDepth-- != 0) nodePath[maxDepth] = null;
  return true;
 }

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 @Override
 public boolean contains(final KEY_TYPE k) {
  return findKey(KEY_GENERIC_CAST k) != null;
 }

#if KEY_CLASS_Object
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public K get(final KEY_TYPE k) {
  final Entry KEY_GENERIC entry = findKey(KEY_GENERIC_CAST k);
  return entry == null ? null : entry.key;
 }
#endif

 @Override
 public void clear() {
  count = 0;
  tree = null;
  firstEntry = lastEntry = null;
 }


 /** This class represent an entry in a tree set.
  *
  * <p>We use the only "metadata", i.e., {@link Entry#info}, to store
  * information about color, predecessor status and successor status.
  *
  * <p>Note that since the class is recursive, it can be
  * considered equivalently a tree.
  */
```

```
 private static final class Entry KEY_GENERIC implements Cloneable {
  /** The the bit in this mask is true, the node is black. */
  private static final int BLACK_MASK = 1;
  /** If the bit in this mask is true, {@link #right} points to a successor. */
  private static final int SUCC_MASK = 1 << 31;
  /** If the bit in this mask is true, {@link #left} points to a predecessor. */
  private static final int PRED_MASK = 1 << 30;
  /** The key of this entry. */
  KEY_GENERIC_TYPE key;
  /** The pointers to the left and right subtrees. */
  Entry KEY_GENERIC left, right;
  /** This integers holds different information in different bits (see {@link #SUCC_MASK}, {@link
#PRED_MASK} and {@link #BLACK_MASK}). */
  int info;

  Entry() {}

  /** Creates a new red entry with the given key.
   *
   * @param k a key.
   */
  Entry(final KEY_GENERIC_TYPE k) {
   this.key = k;
   info = SUCC_MASK | PRED_MASK;
  }

  /** Returns the left subtree.
   *
   * @return the left subtree ({@code null} if the left
   * subtree is empty).
   */
  Entry KEY_GENERIC left() {
   return (info & PRED_MASK) != 0 ? null : left;
  }

  /** Returns the right subtree.
   *
   * @return the right subtree ({@code null} if the right
   * subtree is empty).
   */
  Entry KEY_GENERIC right() {
   return (info & SUCC_MASK) != 0 ? null : right;
  }

  /** Checks whether the left pointer is really a predecessor.
   * @return true if the left pointer is a predecessor.
   */
```

```
boolean pred() {
 return (info & PRED_MASK) != 0;
}

/** Checks whether the right pointer is really a successor.
 * @return true if the right pointer is a successor.
 */
boolean succ() {
 return (info & SUCC_MASK) != 0;
}

/** Sets whether the left pointer is really a predecessor.
 * @param pred if true then the left pointer will be considered a predecessor.
 */
void pred(final boolean pred) {
 if (pred) info |= PRED_MASK;
 else info &= ~PRED_MASK;
}

/** Sets whether the right pointer is really a successor.
 * @param succ if true then the right pointer will be considered a successor.
 */
void succ(final boolean succ) {
 if (succ) info |= SUCC_MASK;
 else info &= ~SUCC_MASK;
}

/** Sets the left pointer to a predecessor.
 * @param pred the predecessr.
 */
void pred(final Entry KEY_GENERIC pred) {
 info |= PRED_MASK;
 left = pred;
}

/** Sets the right pointer to a successor.
 * @param succ the successor.
 */
void succ(final Entry KEY_GENERIC succ) {
 info |= SUCC_MASK;
 right = succ;
}

/** Sets the left pointer to the given subtree.
 * @param left the new left subtree.
 */
void left(final Entry KEY_GENERIC left) {
 info &= ~PRED_MASK;
```

```
 this.left = left;
}

/** Sets the right pointer to the given subtree.
 * @param right the new right subtree.
 */
void right(final Entry KEY_GENERIC right) {
 info &= ~SUCC_MASK;
 this.right = right;
}

/** Returns whether this node is black.
 * @return true iff this node is black.
 */
boolean black() {
 return (info & BLACK_MASK) != 0;
}

/** Sets whether this node is black.
 * @param black if true, then this node becomes black; otherwise, it becomes red..
 */
void black(final boolean black) {
 if (black) info |= BLACK_MASK;
 else info &= ~BLACK_MASK;
}

/** Computes the next entry in the set order.
 *
 * @return the next entry ({@code null}) if this is the last entry).
 */

Entry KEY_GENERIC next() {
 Entry KEY_GENERIC next = this.right;
 if ((info & SUCC_MASK) == 0) while ((next.info & PRED_MASK) == 0) next = next.left;
 return next;
}

/** Computes the previous entry in the set order.
 *
 * @return the previous entry ({@code null}) if this is the first entry).
 */

Entry KEY_GENERIC prev() {
 Entry KEY_GENERIC prev = this.left;
 if ((info & PRED_MASK) == 0) while ((prev.info & SUCC_MASK) == 0) prev = prev.right;
 return prev;
}
```

```
@Override
SUPPRESS_WARNINGS_KEY_UNCHECKED
public Entry KEY_GENERIC clone() {
 Entry KEY_GENERIC c;
 try {
  c = (Entry KEY_GENERIC)super.clone();
 }
 catch(CloneNotSupportedException cantHappen) {
  throw new InternalError();
 }

 c.key = key;
 c.info = info;

 return c;
}

public boolean equals(final Object o) {
 if (!(o instanceof Entry)) return false;
 Entry KEY_GENERIC_WILDCARD e = (Entry KEY_GENERIC_WILDCARD)o;

 return KEY_EQUALS(key, e.key);
}

public int hashCode() {
 return KEY2JAVAHASH_NOT_NULL(key);
}


public String toString() {
 return String.valueOf(key);
}

/*
  public void prettyPrint() {
  prettyPrint(0);
  }


  public void prettyPrint(int level) {
  if (pred()) {
  for (int i = 0; i < level; i++)
  System.err.print("  ");
  System.err.println("pred: " + left);
  }
  else if (left != null)
  left.prettyPrint(level +1);
  for (int i = 0; i < level; i++)
```

```
  System.err.print("  ");
  System.err.println(key + " (" + (black() ? "black" : "red") + ")");
  if (succ()) {
  for (int i = 0; i < level; i++)
  System.err.print("  ");
  System.err.println("succ: " + right);
  }
  else if (right != null)
  right.prettyPrint(level + 1);
  }*/
}

/*
 public void prettyPrint() {
 System.err.println("size: " + count);
 if (tree != null) tree.prettyPrint();
 }
*/

@Override
public int size() {
 return count;
}

@Override
public boolean isEmpty() {
 return count == 0;
}

@Override
public KEY_GENERIC_TYPE FIRST() {
 if (tree == null) throw new NoSuchElementException();
 return firstEntry.key;
}

@Override
public KEY_GENERIC_TYPE LAST() {
 if (tree == null) throw new NoSuchElementException();
 return lastEntry.key;
}


/** An iterator on the whole range.
 *
 * <p>This class can iterate in both directions on a threaded tree.
 */

private class SetIterator implements KEY_LIST_ITERATOR KEY_GENERIC {
```

```
  /** The entry that will be returned by the next call to {@link java.util.ListIterator#previous()} (or {@code null} if no previous entry exists). */
  Entry KEY_GENERIC prev;
  /** The entry that will be returned by the next call to {@link java.util.ListIterator#next()} (or {@code null} if no next entry exists). */
  Entry KEY_GENERIC next;
  /** The last entry that was returned (or {@code null} if we did not iterate or used {@link #remove()}). */
  Entry KEY_GENERIC curr;
  /** The current index (in the sense of a {@link java.util.ListIterator}). Note that this value is not meaningful when this iterator has been created using the nonempty constructor.*/
  int index = 0;

  SetIterator() {
   next = firstEntry;
  }

  SetIterator(final KEY_GENERIC_TYPE k) {
   if ((next = locateKey(k)) != null) {
    if (compare(next.key, k) <= 0) {
     prev = next;
     next = next.next();
    }
    else prev = next.prev();
   }
  }

  @Override
  public boolean hasNext() { return next != null; }
  @Override
  public boolean hasPrevious() { return prev != null; }

  void updateNext() { next = next.next(); }
  void updatePrevious() { prev = prev.prev(); }

  @Override
  public KEY_GENERIC_TYPE NEXT_KEY() { return nextEntry().key; }
  @Override
  public KEY_GENERIC_TYPE PREV_KEY() { return previousEntry().key; }

  Entry KEY_GENERIC nextEntry() {
   if (! hasNext()) throw new NoSuchElementException();
   curr = prev = next;
   index++;
   updateNext();
   return curr;
  }

  Entry KEY_GENERIC previousEntry() {
```

```
  if (! hasPrevious()) throw new NoSuchElementException();
  curr = next = prev;
  index--;
  updatePrevious();
  return curr;
 }

 @Override
 public int nextIndex() { return index; }
 @Override
 public int previousIndex() { return index - 1; }

 @Override
 public void remove() {
  if (curr == null) throw new IllegalStateException();
  /* If the last operation was a next(), we are removing an entry that preceeds
     the current index, and thus we must decrement it. */
  if (curr == prev) index--;
  next = prev = curr;
  updatePrevious();
  updateNext();
  RB_TREE_SET.this.remove(curr.key);
  curr = null;
 }
}

 @Override
 public KEY_BIDI_ITERATOR KEY_GENERIC iterator() { return new SetIterator(); }

 @Override
 public KEY_BIDI_ITERATOR KEY_GENERIC iterator(final KEY_GENERIC_TYPE from) { return new
SetIterator(from); }

 @Override
 public KEY_COMPARATOR KEY_SUPER_GENERIC comparator() { return actualComparator; }

 @Override
 public SORTED_SET KEY_GENERIC headSet(final KEY_GENERIC_TYPE to) { return new
Subset(KEY_NULL, true, to, false); }

 @Override
 public SORTED_SET KEY_GENERIC tailSet(final KEY_GENERIC_TYPE from) { return new Subset(from,
false, KEY_NULL, true); }

 @Override
 public SORTED_SET KEY_GENERIC subSet(final KEY_GENERIC_TYPE from, final KEY_GENERIC_TYPE
to) { return new Subset(from, false, to, false); }
```

```
 /** A subset with given range.
  *
  * <p>This class represents a subset. One has to specify the left/right
  * limits (which can be set to -&infin; or &infin;). Since the subset is a
  * view on the set, at a given moment it could happen that the limits of
  * the range are not any longer in the main set. Thus, things such as
  * {@link java.util.SortedSet#first()} or {@link java.util.Collection#size()} must be always computed
  * on-the-fly.
  */
 private final class Subset extends ABSTRACT_SORTED_SET KEY_GENERIC implements java.io.Serializable,
SORTED_SET KEY_GENERIC {
  private static final long serialVersionUID = -7046029254386353129L;

  /** The start of the subset range, unless {@link #bottom} is true. */
  KEY_GENERIC_TYPE from;
  /** The end of the subset range, unless {@link #top} is true. */
  KEY_GENERIC_TYPE to;
  /** If true, the subset range starts from -&infin;. */
  boolean bottom;
  /** If true, the subset range goes to &infin;. */
  boolean top;

  /** Creates a new subset with given key range.
   *
   * @param from the start of the subset range.
   * @param bottom if true, the first parameter is ignored and the range starts from -&infin;.
   * @param to the end of the subset range.
   * @param top if true, the third parameter is ignored and the range goes to &infin;.
   */
  public Subset(final KEY_GENERIC_TYPE from, final boolean bottom, final KEY_GENERIC_TYPE to, final
boolean top) {
   if (! bottom && ! top && RB_TREE_SET.this.compare(from, to) > 0) throw new IllegalArgumentException("Start
element (" + from  + ") is larger than end element (" + to + ")");

   this.from = from;
   this.bottom = bottom;
   this.to = to;
   this.top = top;
  }

  @Override
  public void clear() {
   final SubsetIterator i = new SubsetIterator();
   while(i.hasNext()) {
    i.NEXT_KEY();
    i.remove();
   }
  }
```

```
 /** Checks whether a key is in the subset range.
  * @param k a key.
  * @return true if is the key is in the subset range.
  */
 final boolean in(final KEY_GENERIC_TYPE k) {
  return (bottom || RB_TREE_SET.this.compare(k, from) >= 0) &&
   (top || RB_TREE_SET.this.compare(k, to) < 0);
 }

 @Override
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public boolean contains(final KEY_TYPE k) {
  return in(KEY_GENERIC_CAST k) && RB_TREE_SET.this.contains(k);
 }

 @Override
 public boolean add(final KEY_GENERIC_TYPE  k) {
  if (! in(k)) throw new IllegalArgumentException("Element (" + k + ") out of range [" + (bottom ? "-" :
String.valueOf(from)) + ", " + (top ? "-" : String.valueOf(to)) + ")");
  return RB_TREE_SET.this.add(k);
 }

 @Override
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public boolean remove(final KEY_TYPE k) {
  if (! in(KEY_GENERIC_CAST k)) return false;
  return RB_TREE_SET.this.remove(k);
 }

 @Override
 public int size() {
  final SubsetIterator i = new SubsetIterator();
  int n = 0;

  while(i.hasNext()) {
   n++;
   i.NEXT_KEY();
  }

  return n;
 }


 @Override
 public boolean isEmpty() {
  return ! new SubsetIterator().hasNext();
 }
```

```
@Override
public KEY_COMPARATOR KEY_SUPER_GENERIC comparator() {
 return actualComparator;
}

@Override
public KEY_BIDI_ITERATOR KEY_GENERIC iterator() {
 return new SubsetIterator();
}

@Override
public KEY_BIDI_ITERATOR KEY_GENERIC iterator(final KEY_GENERIC_TYPE from) {
 return new SubsetIterator(from);
}

@Override
public SORTED_SET KEY_GENERIC headSet(final KEY_GENERIC_TYPE to) {
 if (top) return new Subset(from, bottom, to, false);
 return compare(to, this.to) < 0 ? new Subset(from, bottom, to, false) : this;
}

@Override
public SORTED_SET KEY_GENERIC tailSet(final KEY_GENERIC_TYPE from) {
 if (bottom) return new Subset(from, false, to, top);
 return compare(from, this.from) > 0 ? new Subset(from, false, to, top) : this;
}

@Override
public SORTED_SET KEY_GENERIC subSet(KEY_GENERIC_TYPE from, KEY_GENERIC_TYPE to) {
 if (top && bottom) return new Subset(from, false, to, false);
 if (! top) to = compare(to, this.to) < 0 ? to : this.to;
 if (! bottom) from = compare(from, this.from) > 0 ? from : this.from;
   if (! top && ! bottom && from == this.from && to == this.to) return this;
 return new Subset(from, false, to, false);
}


/** Locates the first entry.
 *
 * @return the first entry of this subset, or {@code null} if the subset is empty.
 */
public RB_TREE_SET.Entry KEY_GENERIC firstEntry() {
 if (tree == null) return null;
 // If this subset goes to -infinity, we return the main set first entry; otherwise, we locate the start of the set.
 RB_TREE_SET.Entry KEY_GENERIC e;
 if (bottom) e = firstEntry;
 else {
```

```
 e = locateKey(from);
 // If we find either the start or something greater we're OK.
 if (compare(e.key, from) < 0) e = e.next();
 }
 // Finally, if this subset doesn't go to infinity, we check that the resulting key isn't greater than the end.
 if (e == null || ! top && compare(e.key, to) >= 0) return null;
 return e;
 }

 /** Locates the last entry.
  *
  * @return the last entry of this subset, or {@code null} if the subset is empty.
  */
 public RB_TREE_SET.Entry KEY_GENERIC lastEntry() {
 if (tree == null) return null;
 // If this subset goes to infinity, we return the main set last entry; otherwise, we locate the end of the set.
 RB_TREE_SET.Entry KEY_GENERIC e;
 if (top) e = lastEntry;
 else {
  e = locateKey(to);
  // If we find something smaller than the end we're OK.
  if (compare(e.key, to) >= 0) e = e.prev();
 }
 // Finally, if this subset doesn't go to -infinity, we check that the resulting key isn't smaller than the start.
 if (e == null || ! bottom && compare(e.key, from) < 0) return null;
 return e;
 }


 @Override
 public KEY_GENERIC_TYPE FIRST() {
 RB_TREE_SET.Entry KEY_GENERIC e = firstEntry();
 if (e == null) throw new NoSuchElementException();
 return e.key;
 }

 @Override
 public KEY_GENERIC_TYPE LAST() {
 RB_TREE_SET.Entry KEY_GENERIC e = lastEntry();
 if (e == null) throw new NoSuchElementException();
 return e.key;
 }

 /** An iterator for subranges.
  *
  * <p>This class inherits from {@link SetIterator}, but overrides the methods that
  * update the pointer after a {@link java.util.ListIterator#next()} or {@link java.util.ListIterator#previous()}. If we
would
```

```
 * move out of the range of the subset we just overwrite the next or previous
 * entry with {@code null}.
 */
private final class SubsetIterator extends SetIterator {
 SubsetIterator() {
  next = firstEntry();
 }

 SubsetIterator(final KEY_GENERIC_TYPE k) {
  this();

  if (next != null) {
   if (! bottom && compare(k, next.key) < 0) prev = null;
   else if (! top && compare(k, (prev = lastEntry()).key) >= 0) next = null;
   else {
    next = locateKey(k);

    if (compare(next.key, k) <= 0) {
     prev = next;
     next = next.next();
    }
    else prev = next.prev();
   }
  }
 }

 @Override
 void updatePrevious() {
  prev = prev.prev();
  if (! bottom && prev != null && RB_TREE_SET.this.compare(prev.key, from) < 0) prev = null;
 }

 @Override
 void updateNext() {
  next = next.next();
  if (! top && next != null && RB_TREE_SET.this.compare(next.key, to) >= 0) next = null;
 }
}
}



/** Returns a deep copy of this tree set.
 *
 * <p>This method performs a deep copy of this tree set; the data stored in the
 * set, however, is not cloned. Note that this makes a difference only for object keys.
 *
 * @return a deep copy of this tree set.
```

```
 */
@Override
SUPPRESS_WARNINGS_KEY_UNCHECKED
public Object clone() {
 RB_TREE_SET KEY_GENERIC c;
 try {
  c = (RB_TREE_SET KEY_GENERIC)super.clone();
 }
 catch(CloneNotSupportedException cantHappen) {
  throw new InternalError();
 }

 c.allocatePaths();

 if (count != 0) {
  // Also this apparently unfathomable code is derived from GNU libavl.
  Entry KEY_GENERIC e, p, q, rp = new Entry KEY_GENERIC_DIAMOND(), rq = new Entry
KEY_GENERIC_DIAMOND();

  p = rp;
  rp.left(tree);

  q = rq;
  rq.pred(null);

  while(true) {
   if (! p.pred()) {
    e = p.left.clone();
    e.pred(q.left);
    e.succ(q);
    q.left(e);

    p = p.left;
    q = q.left;
   }
   else {
    while(p.succ()) {
     p = p.right;

     if (p == null) {
      q.right = null;
      c.tree = rq.left;

      c.firstEntry = c.tree;
      while(c.firstEntry.left != null) c.firstEntry = c.firstEntry.left;
      c.lastEntry = c.tree;
      while(c.lastEntry.right != null) c.lastEntry = c.lastEntry.right;
```

```
      return c;
     }
     q = q.right;
    }

    p = p.right;
    q = q.right;
   }

   if (! p.succ()) {
    e = p.right.clone();
    e.succ(q.right);
    e.pred(q);
    q.right(e);
   }
  }
 }

 return c;
}


private void writeObject(java.io.ObjectOutputStream s) throws java.io.IOException {
 int n = count;
 SetIterator i = new SetIterator();

 s.defaultWriteObject();
 while(n-- != 0) s.WRITE_KEY(i.NEXT_KEY());
}


/** Reads the given number of entries from the input stream, returning the corresponding tree.
 *
 * @param s the input stream.
 * @param n the (positive) number of entries to read.
 * @param pred the entry containing the key that preceeds the first key in the tree.
 * @param succ the entry containing the key that follows the last key in the tree.
 */
SUPPRESS_WARNINGS_KEY_UNCHECKED
private Entry KEY_GENERIC readTree(final java.io.ObjectInputStream s, final int n, final Entry KEY_GENERIC
pred, final Entry KEY_GENERIC succ) throws java.io.IOException, ClassNotFoundException {
 if (n == 1) {
  final Entry KEY_GENERIC top = new Entry KEY_GENERIC_DIAMOND(KEY_GENERIC_CAST
s.READ_KEY());
  top.pred(pred);
  top.succ(succ);
  top.black(true);
```

```
 return top;
 }

 if (n == 2) {
  /* We handle separately this case so that recursion will
   *always* be on nonempty subtrees. */
  final Entry KEY_GENERIC top = new Entry KEY_GENERIC_DIAMOND(KEY_GENERIC_CAST
s.READ_KEY());
  top.black(true);
  top.right(new Entry KEY_GENERIC_DIAMOND(KEY_GENERIC_CAST s.READ_KEY()));
  top.right.pred(top);
  top.pred(pred);
  top.right.succ(succ);

  return top;
 }

 // The right subtree is the largest one.
 final int rightN = n / 2, leftN = n - rightN - 1;

 final Entry KEY_GENERIC top = new Entry KEY_GENERIC_DIAMOND();

 top.left(readTree(s, leftN, pred, top));

 top.key = KEY_GENERIC_CAST s.READ_KEY();
 top.black(true);

 top.right(readTree(s, rightN, top, succ));

 if (n + 2 == ((n + 2)  & -(n + 2))) top.right.black(false); // Quick test for determining whether n + 2 is a power of 2.

 return top;
}


private void readObject(java.io.ObjectInputStream s) throws java.io.IOException, ClassNotFoundException {
 s.defaultReadObject();
 /* The storedComparator is now correctly set, but we must restore
    on-the-fly the actualComparator. */
 setActualComparator();
 allocatePaths();

 if (count != 0) {
  tree = readTree(s, count, null, null);
  Entry KEY_GENERIC e;

  e = tree;
  while(e.left() != null) e = e.left();
```

```
  firstEntry = e;

  e = tree;
  while(e.right() != null) e = e.right();
  lastEntry = e;
 }
}


#ifdef ASSERTS_CODE
 private void checkNodePath() {
  for(int i = nodePath.length; i-- != 0;) assert nodePath[i] == null : i;
 }

 private static KEY_GENERIC int checkTree(Entry KEY_GENERIC e, int d, int D) {
  if (e == null) return 0;
  if (e.black()) d++;
  if (e.left() != null) D = checkTree(e.left(), d, D);
  if (e.right() != null) D = checkTree(e.right(), d, D);
  if (e.left() == null && e.right() == null) {
   if (D == -1) D = d;
   else if (D != d) throw new AssertionError("Mismatch between number of black nodes (" + D + " and " + d + ")");
  }
  return D;
 }
#endif

#ifdef TEST

 private static long seed = System.currentTimeMillis();
 private static java.util.Random r = new java.util.Random(seed);

 private static KEY_TYPE genKey() {
#if KEY_CLASS_Byte || KEY_CLASS_Short || KEY_CLASS_Character
  return (KEY_TYPE)(r.nextInt());
#elif KEYS_PRIMITIVE
  return r.NEXT_KEY();
#else
  return Integer.toBinaryString(r.nextInt());
#endif
 }

 private static java.text.NumberFormat format = new java.text.DecimalFormat("#,###.00");
 private static java.text.FieldPosition p = new java.text.FieldPosition(0);

 private static String format(double d) {
  StringBuffer s = new StringBuffer();
  return format.format(d, s, p).toString();
```

```
}

private static void speedTest(int n, boolean comp) {
 int i, j;
 RB_TREE_SET m;
 java.util.TreeSet t;
 KEY_TYPE k[] = new KEY_TYPE[n];
 KEY_TYPE nk[] = new KEY_TYPE[n];
 long ms;

 for(i = 0; i < n; i++) {
  k[i] = genKey();
  nk[i] = genKey();
 }

 double totAdd = 0, totYes = 0, totNo = 0, totIterFor = 0, totIterBack = 0, totRemYes = 0, d, dd;

 if (comp) {
  for(j = 0; j < 20; j++) {

   t = new java.util.TreeSet();

   /* We first add all pairs to t. */
   for(i = 0; i < n;  i++) t.add(KEY2OBJ(k[i]));

   /* Then we remove the first half and put it back. */
   for(i = 0; i < n/2;  i++) t.remove(KEY2OBJ(k[i]));

   ms = System.currentTimeMillis();
   for(i = 0; i < n/2;  i++) t.add(KEY2OBJ(k[i]));
   d = System.currentTimeMillis() - ms;

   /* Then we remove the other half and put it back again. */
   ms = System.currentTimeMillis();
   for(i = n/2; i < n;  i++) t.remove(KEY2OBJ(k[i]));
   dd = System.currentTimeMillis() - ms ;

   ms = System.currentTimeMillis();
   for(i = n/2; i < n;  i++) t.add(KEY2OBJ(k[i]));
   d += System.currentTimeMillis() - ms;
   if (j > 2) totAdd += n/d;
   System.out.print("Add: " + format(n/d) +" K/s ");

   /* Then we remove again the first half. */
   ms = System.currentTimeMillis();
   for(i = 0; i < n/2;  i++) t.remove(KEY2OBJ(k[i]));
   dd += System.currentTimeMillis() - ms ;
   if (j > 2) totRemYes += n/dd;
```

```
   System.out.print("RemYes: " + format(n/dd) +" K/s ");

   /* And then we put it back. */
   for(i = 0; i < n/2;  i++) t.add(KEY2OBJ(k[i]));

   /* We check for pairs in t. */
   ms = System.currentTimeMillis();
   for(i = 0; i < n;  i++) t.contains(KEY2OBJ(k[i]));
   d = 1.0 * n / (System.currentTimeMillis() - ms);
   if (j > 2) totYes += d;
   System.out.print("Yes: " + format(d) +" K/s ");

   /* We check for pairs not in t. */
   ms = System.currentTimeMillis();
   for(i = 0; i < n;  i++) t.contains(KEY2OBJ(nk[i]));
   d = 1.0 * n / (System.currentTimeMillis() - ms);
   if (j > 2) totNo += d;
   System.out.print("No: " + format(d) +" K/s ");

   /* We iterate on t. */
   ms = System.currentTimeMillis();
   for(Iterator it = t.iterator(); it.hasNext(); it.next());
   d = 1.0 * n / (System.currentTimeMillis() - ms);
   if (j > 2) totIterFor += d;
   System.out.print("IterFor: " + format(d) +" K/s ");

   System.out.println();
   }

   System.out.println();
   System.out.println("java.util Add: " + format(totAdd/(j-3)) + " K/s RemYes: " + format(totRemYes/(j-3)) + " K/s
Yes: " + format(totYes/(j-3)) + " K/s No: " + format(totNo/(j-3)) + " K/s IterFor: " + format(totIterFor/(j-3))  + "
K/s");

   System.out.println();

   totAdd = totYes = totNo = totIterFor = totIterBack = totRemYes = 0;

  }

  for(j = 0; j < 20; j++) {

   m = new RB_TREE_SET();


   /* We first add all pairs to m. */
   for(i = 0; i < n;  i++) m.add(k[i]);
```

```
/* Then we remove the first half and put it back. */
for(i = 0; i < n/2;  i++) m.remove(k[i]);

ms = System.currentTimeMillis();
for(i = 0; i < n/2;  i++) m.add(k[i]);
d = System.currentTimeMillis() - ms;

/* Then we remove the other half and put it back again. */
ms = System.currentTimeMillis();
for(i = n/2; i < n;  i++) m.remove(k[i]);
dd = System.currentTimeMillis() - ms ;

ms = System.currentTimeMillis();
for(i = n/2; i < n;  i++) m.add(k[i]);
d += System.currentTimeMillis() - ms;
if (j > 2) totAdd += n/d;
System.out.print("Add: " + format(n/d) +" K/s ");

/* Then we remove again the first half. */
ms = System.currentTimeMillis();
for(i = 0; i < n/2;  i++) m.remove(k[i]);
dd += System.currentTimeMillis() - ms ;
if (j > 2) totRemYes += n/dd;
System.out.print("RemYes: " + format(n/dd) +" K/s ");

/* And then we put it back. */
for(i = 0; i < n/2;  i++) m.add(k[i]);

/* We check for pairs in m. */
ms = System.currentTimeMillis();
for(i = 0; i < n;  i++) m.contains(k[i]);
d = 1.0 * n / (System.currentTimeMillis() - ms);
if (j > 2) totYes += d;
System.out.print("Yes: " + format(d) +" K/s ");

/* We check for pairs not in m. */
ms = System.currentTimeMillis();
for(i = 0; i < n;  i++) m.contains(nk[i]);
d = 1.0 * n / (System.currentTimeMillis() - ms);
if (j > 2) totNo += d;
System.out.print("No: " + format(d) +" K/s ");

/* We iterate on m. */
KEY_LIST_ITERATOR it = (KEY_LIST_ITERATOR)m.iterator();
ms = System.currentTimeMillis();
for(; it.hasNext(); it.NEXT_KEY());
d = 1.0 * n / (System.currentTimeMillis() - ms);
if (j > 2) totIterFor += d;
```

```
  System.out.print("IterFor: " + format(d) +" K/s ");

  /* We iterate back on m. */
  ms = System.currentTimeMillis();
  for(; it.hasPrevious(); it.PREV_KEY());
  d = 1.0 * n / (System.currentTimeMillis() - ms);
  if (j > 2) totIterBack += d;
  System.out.print("IterBack: " + format(d) +" K/s ");

  System.out.println();
 }


 System.out.println();
 System.out.println("fastutil  Add: " + format(totAdd/(j-3)) + " K/s RemYes: " + format(totRemYes/(j-3)) + " K/s
Yes: " + format(totYes/(j-3)) + " K/s No: " + format(totNo/(j-3)) + " K/s IterFor: " + format(totIterFor/(j-3))  + " K/s
IterBack: " + format(totIterBack/(j-3)) + "K/s");

 System.out.println();
}


private static boolean valEquals(Object o1, Object o2) {
 return o1 == null ? o2 == null : o1.equals(o2);
}

private static void fatal(String msg) {
 System.out.println(msg);
 System.exit(1);
}

private static void ensure(boolean cond, String msg) {
 if (cond) return;
 fatal(msg);
}

private static Object[] k, v, nk;
private static KEY_TYPE kt[];
private static KEY_TYPE nkt[];
private static RB_TREE_SET topSet;

protected static void testSets(SORTED_SET m, SortedSet t, int n, int level) {
 long ms;
 boolean mThrowsIllegal, tThrowsIllegal, mThrowsNoElement, tThrowsNoElement;
 boolean rt = false, rm = false;

 if (level > 4) return;
```

```
  /* Now we check that both sets agree on first/last keys. */

  mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

  try {
   m.first();
  }
  catch (NoSuchElementException e) { mThrowsNoElement = true; }
  try {
   t.first();
  }
  catch (NoSuchElementException e) { tThrowsNoElement = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): first() divergence at start in
NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
  if (! mThrowsNoElement) ensure(t.first().equals(m.first()), "Error (" + level + ", " + seed + "): m and t differ at start
on their first key (" + m.first() + ", " + t.first() +")");

  mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

  try {
   m.last();
  }
  catch (NoSuchElementException e) { mThrowsNoElement = true; }
  try {
   t.last();
  }
  catch (NoSuchElementException e) { tThrowsNoElement = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): last() divergence at start in
NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");


  if (! mThrowsNoElement) ensure(t.last().equals(m.last()), "Error (" + level + ", " + seed + "): m and t differ at start
on their last key (" + m.last() + ", " + t.last() +")");


  /* Now we check that m and t are equal. */
  if (!m.equals(t) || ! t.equals(m)) System.err.println("m: " + m + " t: " + t);

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) at start");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) at start");



  /* Now we check that m actually holds that data. */
  for(Iterator i=t.iterator(); i.hasNext();) {
```

```
  ensure(m.contains(i.next()), "Error (" + level + ", " + seed + "): m and t differ on an entry after insertion (iterating on t)");
  }

  /* Now we check that m actually holds that data, but iterating on m. */
  for(Iterator i=m.iterator(); i.hasNext();) {
  ensure(t.contains(i.next()), "Error (" + level + ", " + seed + "): m and t differ on an entry after insertion (iterating on m)");
  }

  /* Now we check that inquiries about random data give the same answer in m and t. For
     m we use the polymorphic method. */

  for(int i=0; i<n;  i++) {
  KEY_TYPE T = genKey();

  mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

  try {
   m.contains(T);
  }
  catch (NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }

  try {
   t.contains(KEY2OBJ(T));
  }
  catch (NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): contains() divergence in NoSuchElementException (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): contains() divergence in IllegalArgumentException (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
  if (!mThrowsNoElement && !mThrowsIllegal) ensure(m.contains(KEY2OBJ(T)) == t.contains(KEY2OBJ(T)), "Error (" + level + ", " + seed + "): divergence in keys between t and m (polymorphic method)");
  }

  /* Again, we check that inquiries about random data give the same answer in m and t, but
     for m we use the standard method. */

  for(int i=0; i<n;  i++) {
  KEY_TYPE T = genKey();

  mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

  try {
   m.contains(KEY2OBJ(T));
```

```
  }
  catch (NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }

  try {
   t.contains(KEY2OBJ(T));
  }
  catch (NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): contains() divergence in
NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): contains() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
  if (!mThrowsNoElement && !mThrowsIllegal) ensure(m.contains(KEY2OBJ(T)) ==  t.contains(KEY2OBJ(T)),
"Error (" + level + ", " + seed + "): divergence between t and m (standard method)");
  }

  /* Now we add and remove random data in m and t, checking that the result is the same. */

  for(int i=0; i<20*n;  i++) {
   KEY_TYPE T = genKey();

  mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

  try {
   rm = m.add(KEY2OBJ(T));
  }
  catch (NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }

  try {
   rt = t.add(KEY2OBJ(T));
  }
  catch (NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }


  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): add() divergence in
NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): add() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
  if (!mThrowsNoElement && !mThrowsIllegal) ensure(rm == rt, "Error (" + level + ", " + seed + "): divergence in
add() between t and m");

  T = genKey();

  mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;
```

```
  try {
   rm = m.remove(KEY2OBJ(T));
  }
  catch (NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }

  try {
   rt = t.remove(KEY2OBJ(T));
  }
  catch (NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }


  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): remove() divergence in
NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): remove() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
  if (!mThrowsNoElement && !mThrowsIllegal) ensure(rm == rt, "Error (" + level + ", " + seed + "): divergence in
remove() between t and m");
  }

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after removal");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after removal");

  /* Now we check that m actually holds the same data. */

  for(Iterator i=t.iterator(); i.hasNext();) {
   ensure(m.contains(i.next()), "Error (" + level + ", " + seed + "): m and t differ on an entry after removal (iterating on
t)");
  }

  /* Now we check that m actually holds that data, but iterating on m. */

  for(Iterator i=m.iterator(); i.hasNext();) {
   ensure(t.contains(i.next()), "Error (" + level + ", " + seed + "): m and t differ on an entry after removal (iterating on
m)");
  }

  /* Now we check that both sets agree on first/last keys. */

  mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

  try {
   m.first();
  }
  catch (NoSuchElementException e) { mThrowsNoElement = true; }
  try {
```

```
  t.first();
  }
  catch (NoSuchElementException e) { tThrowsNoElement = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): first() divergence in
NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
  if (! mThrowsNoElement) ensure(t.first().equals(m.first()), "Error (" + level + ", " + seed + "): m and t differ on their
first key (" + m.first() + ", " + t.first() +")");

  mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

  try {
   m.last();
  }
  catch (NoSuchElementException e) { mThrowsNoElement = true; }
  try {
   t.last();
  }
  catch (NoSuchElementException e) { tThrowsNoElement = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): last() divergence in
NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");

  if (! mThrowsNoElement) ensure(t.last().equals(m.last()), "Error (" + level + ", " + seed + "): m and t differ on their
last key (" + m.last() + ", " + t.last() +")");

  /* Now we check cloning. */

  if (level == 0) {
   ensure(m.equals(((RB_TREE_SET)m).clone()), "Error (" + level + ", " + seed + "): m does not equal m.clone()");
   ensure(((RB_TREE_SET)m).clone().equals(m), "Error (" + level + ", " + seed + "): m.clone() does not equal m");
   m = (RB_TREE_SET)((RB_TREE_SET)m).clone();
  }

  /* Now we play with constructors. */
  ensure(m.equals(new RB_TREE_SET((Collection)m)), "Error (" + level + ", " + seed + "): m does not equal new
(Collection m)");
  ensure((new RB_TREE_SET((Collection)m)).equals(m), "Error (" + level + ", " + seed + "): new (Collection
m)does not equal m");
  ensure(m.equals(new RB_TREE_SET((COLLECTION)m)), "Error (" + level + ", " + seed + "): m does not equal
new (type-specific Collection m)");
  ensure((new RB_TREE_SET((COLLECTION)m)).equals(m), "Error (" + level + ", " + seed + "): new (type-specific
Collection m) does not equal m");
  ensure(m.equals(new RB_TREE_SET((SortedSet)m)), "Error (" + level + ", " + seed + "): m does not equal new
(SortedSet m)");
  ensure((new RB_TREE_SET((SortedSet)m)).equals(m), "Error (" + level + ", " + seed + "): new (SortedSet m) does
not equal m");
  ensure(m.equals(new RB_TREE_SET((SORTED_SET)m)), "Error (" + level + ", " + seed + "): m does not equal
```

```
new (type-specific SortedSet m)");
 ensure((new RB_TREE_SET((SORTED_SET)m)).equals(m), "Error (" + level + ", " + seed + "): new (type-specific SortedSet m) does not equal m");
 ensure(m.equals(new RB_TREE_SET(m.iterator())), "Error (" + level + ", " + seed + "): m does not equal new (m.iterator())");
 ensure((new RB_TREE_SET(m.iterator())).equals(m), "Error (" + level + ", " + seed + "): new (m.iterator()) does not equal m");
 ensure(m.equals(new RB_TREE_SET(m.iterator())), "Error (" + level + ", " + seed + "): m does not equal new (m.type_specific_iterator())");
 ensure((new RB_TREE_SET(m.iterator())).equals(m), "Error (" + level + ", " + seed + "): new (m.type_specific_iterator()) does not equal m");

 /* Now we play with conversion to array, wrapping and copying. */
 ensure(m.equals(new RB_TREE_SET(m.TO_KEY_ARRAY())), "Error (" + level + ", " + seed + "): m does not equal new (toArray(m))");
 ensure((new RB_TREE_SET(m.TO_KEY_ARRAY())).equals(m), "Error (" + level + ", " + seed + "): new (toArray(m)) does not equal m");



 int h = m.hashCode();


 /* Now we save and read m. */

 SORTED_SET m2 = null;

 try {
  java.io.File ff = new java.io.File("it.unimi.dsi.fastutil.test");
  java.io.OutputStream os = new java.io.FileOutputStream(ff);
  java.io.ObjectOutputStream oos = new java.io.ObjectOutputStream(os);

  oos.writeObject(m);
  oos.close();

  java.io.InputStream is = new java.io.FileInputStream(ff);
  java.io.ObjectInputStream ois = new java.io.ObjectInputStream(is);

  m2 = (SORTED_SET)ois.readObject();
  ois.close();
  ff.delete();
 }
 catch(Exception e) {
  e.printStackTrace();
  System.exit(1);
 }

 ensure(m2.hashCode() == h, "Error (" + level + ", " + seed + "): hashCode() changed after save/read");
```

```
/* Now we check that m2 actually holds that data. */

ensure(m2.equals(t), "Error (" + level + ", " + seed + "): ! m2.equals(t) after save/read");
ensure(t.equals(m2), "Error (" + level + ", " + seed + "): ! t.equals(m2) after save/read");

/* Now we take out of m everything, and check that it is empty. */

for(Iterator i=t.iterator(); i.hasNext();) m2.remove(i.next());

ensure(m2.isEmpty(), "Error (" + level + ", " + seed + "): m2 is not empty (as it should be)");

/* Now we play with iterators. */

{
 java.util.ListIterator i, j;
 Object J;
 i = (java.util.ListIterator)m.iterator();
 j = new java.util.LinkedList(t).listIterator();

 for(int k = 0; k < 2*n; k++) {
  ensure(i.hasNext() == j.hasNext(), "Error (" + level + ", " + seed + "): divergence in hasNext()");
  ensure(i.hasPrevious() == j.hasPrevious(), "Error (" + level + ", " + seed + "): divergence in hasPrevious()");

  if (r.nextFloat() < .8 && i.hasNext()) {
   ensure(i.next().equals(J = j.next()), "Error (" + level + ", " + seed + "): divergence in next()");

   if (r.nextFloat() < 0.5) {
    i.remove();
    j.remove();
    t.remove(J);
   }
  }
  else if (r.nextFloat() < .2 && i.hasPrevious()) {
   ensure(i.previous().equals(J = j.previous()), "Error (" + level + ", " + seed + "): divergence in previous()");

   if (r.nextFloat() < 0.5) {
    i.remove();
    j.remove();
    t.remove(J);
   }
  }

  ensure(i.nextIndex() == j.nextIndex(), "Error (" + level + ", " + seed + "): divergence in nextIndex()");
  ensure(i.previousIndex() == j.previousIndex(), "Error (" + level + ", " + seed + "): divergence in previousIndex()");

 }
```

```
 }

 {
  boolean badPrevious = false;
  Object previous = null;
  it.unimi.dsi.fastutil.BidirectionalIterator i;
  java.util.ListIterator j;
  Object I, J;
  KEY_TYPE from = genKey();
  j = new java.util.LinkedList(t).listIterator();
  while(j.hasNext()) {
   Object k = j.next();
   if (((Comparable)k).compareTo(KEY2OBJ(from)) > 0) {
    badPrevious = true;
    j.previous();
    break;
   }
   previous = k;
  }

  i = (it.unimi.dsi.fastutil.BidirectionalIterator)m.iterator(from);

  for(int k = 0; k < 2*n; k++) {
   ensure(i.hasNext() == j.hasNext(), "Error (" + level + ", " + seed + "): divergence in hasNext() (iterator with starting point " + from + ")");
   ensure(i.hasPrevious() == j.hasPrevious() || badPrevious && (i.hasPrevious() == (previous != null)), "Error (" + level + ", " + seed + "): divergence in hasPrevious() (iterator with starting point " + from + ")" + badPrevious);

   if (r.nextFloat() < .8 && i.hasNext()) {
    ensure((I = i.next()).equals(J = j.next()), "Error (" + level + ", " + seed + "): divergence in next() (" + I + ", " + J + ", iterator with starting point " + from + ")");
    //System.err.println("Done next " + I + " " + J + "  " + badPrevious);

    badPrevious = false;

    if (r.nextFloat() < 0.5) {
     //System.err.println("Removing in next");
     i.remove();
     j.remove();
     t.remove(J);
    }
   }
   else if (!badPrevious && r.nextFloat() < .2 && i.hasPrevious()) {
    ensure((I = i.previous()).equals(J = j.previous()), "Error (" + level + ", " + seed + "): divergence in previous() (" + I + ", " + J + ", iterator with starting point " + from + ")");

    if (r.nextFloat() < 0.5) {
     //System.err.println("Removing in prev");
```

```
    i.remove();
    j.remove();
    t.remove(J);
   }
  }
 }

}

/* Now we check that m actually holds that data. */

ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after iteration");
ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after iteration");

/* Now we select a pair of keys and create a subset. */

if (! m.isEmpty()) {
 java.util.ListIterator i;
 Object start = m.first(), end = m.first();
 for(i = (java.util.ListIterator)m.iterator(); i.hasNext() && r.nextFloat() < .3; start = end = i.next());
 for(; i.hasNext() && r.nextFloat() < .95; end = i.next());

 //System.err.println("Checking subSet from " + start + " to " + end + " (level=" + (level+1) + ")...");
 testSets((SORTED_SET)m.subSet((KEY_CLASS)start, (KEY_CLASS)end), t.subSet(start, end), n, level + 1);

 ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after subSet");
 ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after subSet");

 //System.err.println("Checking headSet to " + end + " (level=" + (level+1) + ")...");
 testSets((SORTED_SET)m.headSet((KEY_CLASS)end), t.headSet(end), n, level + 1);

 ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after headSet");
 ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after headSet");

 //System.err.println("Checking tailSet from " + start + " (level=" + (level+1) + ")...");
 testSets((SORTED_SET)m.tailSet((KEY_CLASS)start), t.tailSet(start), n, level + 1);

 ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after tailSet");
 ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after tailSet");
 }


}


private static void runTest(int n) {
 RB_TREE_SET m = new RB_TREE_SET();
 SortedSet t = new java.util.TreeSet();
```

```
  topSet = m;
  k = new Object[n];
  nk = new Object[n];
  kt = new KEY_TYPE[n];
  nkt = new KEY_TYPE[n];

  for(int i = 0; i < n; i++) {
#if KEY_CLASS_Object
   k[i] = kt[i] = genKey();
   nk[i] = nkt[i] = genKey();
#else
   k[i] = new KEY_CLASS(kt[i] = genKey());
   nk[i] = new KEY_CLASS(nkt[i] = genKey());
#endif
  }

  /* We add pairs to t. */
  for(int i = 0; i < n;  i++) t.add(k[i]);

  /* We add to m the same data */
  m.addAll(t);

  testSets(m, t, n, 0);

  System.out.println("Test OK");
  return;
 }


 public static void main(String args[]) {
  int n  = Integer.parseInt(args[1]);
  if (args.length > 2) r = new java.util.Random(seed = Long.parseLong(args[2]));

  try {
   if ("speedTest".equals(args[0]) || "speedComp".equals(args[0])) speedTest(n, "speedComp".equals(args[0]));
   else if ("test".equals(args[0])) runTest(n);
  } catch(Throwable e) {
   e.printStackTrace(System.err);
   System.err.println("seed: " + seed);
  }
 }

#endif


}
```

Found in path(s):

* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/RBTreeSet.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import VALUE_PACKAGE.VALUE_COLLECTION;

import it.unimi.dsi.fastutil.objects.ObjectSortedSet;
import it.unimi.dsi.fastutil.objects.ObjectBidirectionalIterator;

import java.util.Map;
import java.util.SortedMap;

#if KEYS_REFERENCE
import java.util.Comparator;
#endif

/** A type-specific {@link SortedMap}; provides some additional methods that use polymorphism to avoid (un)boxing.
*
* <p>Additionally, this interface strengthens {@link #entrySet()},
* {@link #keySet()}, {@link #values()},
* {@link #comparator()}, {@link SortedMap#subMap(Object,Object)}, {@link SortedMap#headMap(Object)} and {@link SortedMap#tailMap(Object)}.
*
* @see SortedMap
*/

public interface SORTED_MAP KEY_VALUE_GENERIC extends MAP KEY_VALUE_GENERIC, SortedMap<KEY_GENERIC_CLASS, VALUE_GENERIC_CLASS> {
```

```
 /** Returns a view of the portion of this sorted map whose keys range from {@code fromKey}, inclusive, to
{@code toKey}, exclusive.
  *
  * <p>Note that this specification strengthens the one given in {@link SortedMap#subMap(Object,Object)}.
  *
  * @see SortedMap#subMap(Object,Object)
  */
#if KEYS_REFERENCE
 @Override
#endif
 SORTED_MAP KEY_VALUE_GENERIC subMap(KEY_GENERIC_TYPE fromKey, KEY_GENERIC_TYPE
toKey);

 /** Returns a view of the portion of this sorted map whose keys are strictly less than {@code toKey}.
  *
  * <p>Note that this specification strengthens the one given in {@link SortedMap#headMap(Object)}.
  *
  * @see SortedMap#headMap(Object)
  */
#if KEYS_REFERENCE
 @Override
#endif
 SORTED_MAP KEY_VALUE_GENERIC headMap(KEY_GENERIC_TYPE toKey);

 /** Returns a view of the portion of this sorted map whose keys are greater than or equal to {@code fromKey}.
  *
  * <p>Note that this specification strengthens the one given in {@link SortedMap#tailMap(Object)}.
  *
  * @see SortedMap#tailMap(Object)
  */
#if KEYS_REFERENCE
 @Override
#endif
 SORTED_MAP KEY_VALUE_GENERIC tailMap(KEY_GENERIC_TYPE fromKey);

#if KEYS_PRIMITIVE

 /** Returns the first (lowest) key currently in this map.
  * @see SortedMap#firstKey()
  */
 KEY_GENERIC_TYPE FIRST_KEY();

 /** Returns the last (highest) key currently in this map.
  * @see SortedMap#lastKey()
  */
 KEY_GENERIC_TYPE LAST_KEY();
```

```
/** {@inheritDoc}
 * <p>Note that this specification strengthens the one given in {@link SortedMap#subMap(Object,Object)}.
 * @deprecated Please use the corresponding type-specific method instead.
 */
@Deprecated
@Override
default SORTED_MAP KEY_VALUE_GENERIC subMap(final KEY_GENERIC_CLASS from, final
KEY_GENERIC_CLASS to) {
 return subMap(KEY_CLASS2TYPE(from), KEY_CLASS2TYPE(to));
}

/** {@inheritDoc}
 * <p>Note that this specification strengthens the one given in {@link SortedMap#headMap(Object)}.
 * @deprecated Please use the corresponding type-specific method instead.
 */
@Deprecated
@Override
default SORTED_MAP KEY_VALUE_GENERIC headMap(final KEY_GENERIC_CLASS to) {
 return headMap(KEY_CLASS2TYPE(to));
}

/** {@inheritDoc}
 * <p>Note that this specification strengthens the one given in {@link SortedMap#tailMap(Object)}.
 * @deprecated Please use the corresponding type-specific method instead.
 */
@Deprecated
@Override
default SORTED_MAP KEY_VALUE_GENERIC tailMap(final KEY_GENERIC_CLASS from) {
 return tailMap(KEY_CLASS2TYPE(from));
}

/** {@inheritDoc}
 * @deprecated Please use the corresponding type-specific method instead.
 */
@Deprecated
@Override
default KEY_GENERIC_CLASS firstKey() {
 return KEY2OBJ(FIRST_KEY());
}

/** {@inheritDoc}
 * @deprecated Please use the corresponding type-specific method instead.
 */
@Deprecated
@Override
default KEY_GENERIC_CLASS lastKey() {
 return KEY2OBJ(LAST_KEY());
}
```

```
#endif


 /** A sorted entry set providing fast iteration.
  *
  * <p>In some cases (e.g., hash-based classes) iteration over an entry set requires the creation
  * of a large number of entry objects. Some {@code fastutil}
  * maps might return {@linkplain #entrySet() entry set} objects of type {@code FastSortedEntrySet}: in this case,
{@link #fastIterator() fastIterator()}
  * will return an iterator that is guaranteed not to create a large number of objects, <em>possibly
  * by returning always the same entry</em> (of course, mutated).
  */
 interface FastSortedEntrySet KEY_VALUE_GENERIC extends ObjectSortedSet<MAP.Entry
KEY_VALUE_GENERIC>, FastEntrySet KEY_VALUE_GENERIC {
  /** {@inheritDoc}
   */
  @Override
  ObjectBidirectionalIterator<MAP.Entry KEY_VALUE_GENERIC> fastIterator();


  /** Returns a fast iterator over this entry set, starting from a given element of the domain (optional operation);
   * the iterator might return always the same entry instance, suitably mutated.
   *
   * @param from an element to start from.
   * @return a fast iterator over this sorted entry set starting at {@code from}; the iterator might return always the
same entry object, suitably mutated.
   */
  ObjectBidirectionalIterator<MAP.Entry KEY_VALUE_GENERIC> fastIterator(MAP.Entry
KEY_VALUE_GENERIC from);
 }

#if KEYS_PRIMITIVE || VALUES_PRIMITIVE
 /** Returns a sorted-set view of the mappings contained in this map.
  *  <p>Note that this specification strengthens the one given in the
  *  corresponding type-specific unsorted map.
  *
  * @return a sorted-set view of the mappings contained in this map.
  * @see SortedMap#entrySet()
  * @deprecated Please use the corresponding type-specific method instead.
  */

 @SuppressWarnings({ "unchecked", "rawtypes" })
 @Deprecated
 @Override
 default ObjectSortedSet<Map.Entry<KEY_GENERIC_CLASS, VALUE_GENERIC_CLASS>> entrySet() {
  return (ObjectSortedSet)ENTRYSET();
 }
#else
```

```
 /** Returns a sorted-set view of the mappings contained in this map.
  *  <p>Note that this specification strengthens the one given in the
  *  corresponding type-specific unsorted map.
  *
  * @return a sorted-set view of the mappings contained in this map.
  * @see Map#entrySet()
  */

 @SuppressWarnings({ "unchecked", "rawtypes" })
 @Override
 default ObjectSortedSet<Map.Entry<KEY_GENERIC_CLASS, VALUE_GENERIC_CLASS>> entrySet() {
  return (ObjectSortedSet)ENTRYSET();
 }
#endif

 /** Returns a type-specific sorted-set view of the mappings contained in this map.
  * <p>Note that this specification strengthens the one given in the
  * corresponding type-specific unsorted map.
  *
  * @return a type-specific sorted-set view of the mappings contained in this map.
  * @see #entrySet()
  */

 @Override
 ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> ENTRYSET();

 /** Returns a type-specific sorted-set view of the keys contained in this map.
  *  <p>Note that this specification strengthens the one given in the
  *  corresponding type-specific unsorted map.
  *
  * @return a sorted-set view of the keys contained in this map.
  * @see SortedMap#keySet()
  */

 @Override
 SORTED_SET KEY_GENERIC keySet();

 /** Returns a type-specific set view of the values contained in this map.
  * <p>Note that this specification strengthens the one given in {@link Map#values()},
  * which was already strengthened in the corresponding type-specific class,
  * but was weakened by the fact that this interface extends {@link SortedMap}.
  *
  * @return a set view of the values contained in this map.
  * @see SortedMap#values()
  */

 @Override
 VALUE_COLLECTION VALUE_GENERIC values();
```

```
 /** Returns the comparator associated with this sorted set, or null if it uses its keys' natural ordering.
  *
  *  <p>Note that this specification strengthens the one given in {@link SortedMap#comparator()}.
  *
  * @see SortedMap#comparator()
  */

 @Override
 KEY_COMPARATOR KEY_SUPER_GENERIC comparator();
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/SortedMap.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

/** An abstract class providing basic methods for functions implementing a type-specific interface.
*
* <p>This class handles directly a default return
* value (including {@linkplain #defaultReturnValue() methods to access
* it}). Instances of classes inheriting from this class have just to return
* {@code defRetValue} to denote lack of a key in type-specific methods. The value
* is serialized.
*
* <p>Implementing subclasses have just to provide type-specific {@code get()},
* type-specific {@code containsKey()}, and {@code size()} methods.
*
*/
```

```
public abstract class ABSTRACT_FUNCTION KEY_VALUE_GENERIC implements FUNCTION
KEY_VALUE_GENERIC, java.io.Serializable {

 private static final long serialVersionUID = -4940583368468432370L;

 protected ABSTRACT_FUNCTION() {}

 /**
  * The default return value for {@code get()}, {@code put()} and
  * {@code remove()}.
  */

 protected VALUE_GENERIC_TYPE defRetValue;

 @Override
 public void defaultReturnValue(final VALUE_GENERIC_TYPE rv) {
  defRetValue = rv;
 }

 @Override
 public VALUE_GENERIC_TYPE defaultReturnValue() {
  return defRetValue;
 }
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/AbstractFunction.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2003-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
```

```
package PACKAGE;

/** An abstract class providing basic methods for sorted sets implementing a type-specific interface. */

public abstract class ABSTRACT_SORTED_SET KEY_GENERIC extends ABSTRACT_SET KEY_GENERIC implements SORTED_SET KEY_GENERIC {

 protected ABSTRACT_SORTED_SET() {}

 @Override
 public abstract KEY_BIDI_ITERATOR KEY_GENERIC iterator();
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/AbstractSortedSet.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import java.util.Collection;

import it.unimi.dsi.fastutil.objects.ObjectArrays;


/** A class providing static methods and objects that do useful things with type-specific collections.
*
* @see java.util.Collections
*/

public final class COLLECTIONS {
```

```
 private COLLECTIONS() {}

 /** An immutable class representing an empty type-specific collection.
  *
  * <p>This class may be useful to implement your own in case you subclass
  * a type-specific collection.
  */

 public abstract static class EmptyCollection KEY_GENERIC extends ABSTRACT_COLLECTION
KEY_GENERIC {
  protected EmptyCollection() {}

  @Override
  public boolean contains(KEY_TYPE k) { return false; }

  @Override
  public Object[] toArray() { return ObjectArrays.EMPTY_ARRAY; }

  @Override
  SUPPRESS_WARNINGS_KEY_UNCHECKED
  public KEY_BIDI_ITERATOR KEY_GENERIC iterator() { return ITERATORS.EMPTY_ITERATOR; }

  @Override
  public int size() { return 0; }

  @Override
  public void clear() {}

  @Override
  public int hashCode() { return 0; }

  @Override
  public boolean equals(Object o) {
   if (o == this) return true;
   if (! (o instanceof Collection)) return false;
   return ((Collection<?>)o).isEmpty();
  }

  @Override
  public boolean addAll(final Collection<? extends KEY_GENERIC_CLASS> c) { throw new
UnsupportedOperationException(); }

  @Override
  public boolean removeAll(final Collection<?> c) { throw new UnsupportedOperationException(); }

  @Override
  public boolean retainAll(final Collection<?> c) { throw new UnsupportedOperationException(); }
```

```
#if KEYS_PRIMITIVE

 @Override
 public boolean addAll(final COLLECTION c) { throw new UnsupportedOperationException(); }

 @Override
 public boolean removeAll(final COLLECTION c) { throw new UnsupportedOperationException(); }

 @Override
 public boolean retainAll(final COLLECTION c) { throw new UnsupportedOperationException(); }

#endif

 }


 /** A synchronized wrapper class for collections. */

 public static class SynchronizedCollection KEY_GENERIC implements COLLECTION KEY_GENERIC,
java.io.Serializable {

 private static final long serialVersionUID = -7046029254386353129L;

 protected final COLLECTION KEY_GENERIC collection;
 protected final Object sync;

 protected SynchronizedCollection(final COLLECTION KEY_GENERIC c, final Object sync) {
  if (c == null) throw new NullPointerException();
  this.collection = c;
  this.sync = sync;
 }

 protected SynchronizedCollection(final COLLECTION KEY_GENERIC c) {
  if (c == null) throw new NullPointerException();
  this.collection = c;
  this.sync = this;
 }

 @Override
 public boolean add(final KEY_GENERIC_TYPE k) { synchronized(sync) { return collection.add(k); } }

 @Override
 public boolean contains(final KEY_TYPE k) { synchronized(sync) { return collection.contains(k); } }

 @Override
 public boolean REMOVE(final KEY_TYPE k) { synchronized(sync) { return collection.REMOVE(k); } }
```

```
  @Override
  public int size() { synchronized(sync) { return collection.size(); } }

  @Override
  public boolean isEmpty() { synchronized(sync) { return collection.isEmpty(); } }

  @Override
  public KEY_TYPE[] TO_KEY_ARRAY() { synchronized(sync) { return collection.TO_KEY_ARRAY(); } }

#if KEYS_PRIMITIVE
  @Override
  public Object[] toArray() { synchronized(sync) { return collection.toArray(); } }

  /* {@inheritDoc}
   * @deprecated Please use {@code toArray()} instead&amp;mdash;this method is redundant and will be removed in the future.
   */
  @Deprecated
  @Override
  public KEY_TYPE[] TO_KEY_ARRAY(final KEY_TYPE[] a) { return toArray(a); }

  @Override
  public KEY_TYPE[] toArray(final KEY_TYPE[] a) { synchronized(sync) { return collection.toArray(a); } }

  @Override
  public boolean addAll(final COLLECTION c) { synchronized(sync) { return collection.addAll(c); } }

  @Override
  public boolean containsAll(final COLLECTION c) { synchronized(sync) { return collection.containsAll(c); } }

  @Override
  public boolean removeAll(final COLLECTION c) { synchronized(sync) { return collection.removeAll(c); } }

#ifdef JDK_PRIMITIVE_PREDICATE
  @Override
  public boolean removeIf(final JDK_PRIMITIVE_PREDICATE filter) { synchronized(sync) { return collection.removeIf(filter); } }
#endif

  @Override
  public boolean retainAll(final COLLECTION c) { synchronized(sync) { return collection.retainAll(c); } }

  @Override
  @Deprecated
  public boolean add(final KEY_GENERIC_CLASS k) { synchronized(sync) { return collection.add(k); } }

  @Override
  @Deprecated
```

```
  public boolean contains(final Object k) { synchronized(sync) { return collection.contains(k); } }

  @Override
  @Deprecated
  public boolean remove(final Object k) { synchronized(sync) { return collection.remove(k); } }
#endif

  @Override
  public <T> T[] toArray(final T[] a) { synchronized(sync) { return collection.toArray(a); } }

  @Override
  public KEY_ITERATOR KEY_GENERIC iterator() { return collection.iterator(); }

  @Override
  public boolean addAll(final Collection<? extends KEY_GENERIC_CLASS> c) { synchronized(sync) { return
collection.addAll(c); } }

  @Override
  public boolean containsAll(final Collection<?> c) { synchronized(sync) { return collection.containsAll(c); } }

  @Override
  public boolean removeAll(final Collection<?> c) { synchronized(sync) { return collection.removeAll(c); } }

  @Override
  public boolean retainAll(final Collection<?> c) { synchronized(sync) { return collection.retainAll(c); } }

  @Override
  public void clear() { synchronized(sync) { collection.clear(); } }

  @Override
  public String toString() { synchronized(sync) { return collection.toString(); } }

  @Override
  public int hashCode() { synchronized(sync) { return collection.hashCode(); } }

  @Override
  public boolean equals(final Object o) { if (o == this) return true; synchronized(sync) { return collection.equals(o); }
}

  private void writeObject(java.io.ObjectOutputStream s) throws java.io.IOException {
   synchronized(sync) { s.defaultWriteObject(); }
  }
 }



 /** Returns a synchronized collection backed by the specified collection.
  *
  * @param c the collection to be wrapped in a synchronized collection.
```

```
 * @return a synchronized view of the specified collection.
 * @see java.util.Collections#synchronizedCollection(Collection)
 */
 public static KEY_GENERIC COLLECTION KEY_GENERIC synchronize(final COLLECTION KEY_GENERIC
c) { return new SynchronizedCollection KEY_GENERIC_DIAMOND(c); }

 /** Returns a synchronized collection backed by the specified collection, using an assigned object to synchronize.
 *
 * @param c the collection to be wrapped in a synchronized collection.
 * @param sync an object that will be used to synchronize the list access.
 * @return a synchronized view of the specified collection.
 * @see java.util.Collections#synchronizedCollection(Collection)
 */

 public static KEY_GENERIC COLLECTION KEY_GENERIC synchronize(final COLLECTION KEY_GENERIC
c, final Object sync) { return new SynchronizedCollection KEY_GENERIC_DIAMOND(c, sync); }


 /** An unmodifiable wrapper class for collections. */

 public static class UnmodifiableCollection KEY_GENERIC implements COLLECTION KEY_GENERIC,
java.io.Serializable {

  private static final long serialVersionUID = -7046029254386353129L;

  protected final COLLECTION KEY_GENERIC collection;

  protected UnmodifiableCollection(final COLLECTION KEY_GENERIC c) {
   if (c == null) throw new NullPointerException();
   this.collection = c;
  }

  @Override
  public boolean add(final KEY_GENERIC_TYPE k) { throw new UnsupportedOperationException(); }

  @Override
  public boolean REMOVE(final KEY_TYPE k) { throw new UnsupportedOperationException(); }

  @Override
  public int size() { return collection.size(); }

  @Override
  public boolean isEmpty() { return collection.isEmpty(); }

  @Override
  public boolean contains(final KEY_TYPE o) { return collection.contains(o); }

  @Override
```

```
  public KEY_ITERATOR KEY_GENERIC iterator() { return ITERATORS.unmodifiable(collection.iterator()); }

  @Override
  public void clear() { throw new UnsupportedOperationException(); }

  @Override
  public <T> T[] toArray(final T[] a) { return collection.toArray(a); }

  @Override
  public Object[] toArray() { return collection.toArray(); }

  @Override
  public boolean containsAll(Collection<?> c) { return collection.containsAll(c); }

  @Override
  public boolean addAll(Collection<? extends KEY_GENERIC_CLASS> c) { throw new
UnsupportedOperationException(); }

  @Override
  public boolean removeAll(Collection<?> c) { throw new UnsupportedOperationException(); }

  @Override
  public boolean retainAll(Collection<?> c) { throw new UnsupportedOperationException(); }

#if KEYS_PRIMITIVE
  @Override
  @Deprecated
  public boolean add(final KEY_GENERIC_CLASS k) { throw new UnsupportedOperationException(); }

  @Override
  @Deprecated
  public boolean contains(final Object k) { return collection.contains(k); }

  @Override
  @Deprecated
  public boolean remove(final Object k) { throw new UnsupportedOperationException(); }

  @Override
  public KEY_TYPE[] TO_KEY_ARRAY() { return collection.TO_KEY_ARRAY(); }

  /* {@inheritDoc}
   * @deprecated Please use {@code toArray()} instead&amp;mdash;this method is redundant.
   */
  @Deprecated
  @Override
  public KEY_TYPE[] TO_KEY_ARRAY(final KEY_TYPE[] a) { return toArray(a); }

  @Override
```

```
 public KEY_TYPE[] toArray(final KEY_TYPE[] a) { return collection.toArray(a); }

 @Override
 public boolean containsAll(COLLECTION c) { return collection.containsAll(c); }

 @Override
 public boolean addAll(COLLECTION c) { throw new UnsupportedOperationException(); }

 @Override
 public boolean removeAll(COLLECTION c) { throw new UnsupportedOperationException(); }

 @Override
 public boolean retainAll(COLLECTION c) { throw new UnsupportedOperationException(); }
#endif
 @Override
 public String toString() { return collection.toString(); }

 @Override
 public int hashCode() { return collection.hashCode(); }

 @Override
 public boolean equals(final Object o) { if (o == this) return true; return collection.equals(o); }
 }


 /** Returns an unmodifiable collection backed by the specified collection.
  *
  * @param c the collection to be wrapped in an unmodifiable collection.
  * @return an unmodifiable view of the specified collection.
  * @see java.util.Collections#unmodifiableCollection(Collection)
  */
 public static KEY_GENERIC COLLECTION KEY_GENERIC unmodifiable(final COLLECTION
KEY_GENERIC c) { return new UnmodifiableCollection KEY_GENERIC_DIAMOND(c); }

 /** A collection wrapper class for iterables. */

 public static class IterableCollection KEY_GENERIC extends ABSTRACT_COLLECTION KEY_GENERIC
implements java.io.Serializable {

 private static final long serialVersionUID = -7046029254386353129L;

 protected final KEY_ITERABLE KEY_GENERIC iterable;

 protected IterableCollection(final KEY_ITERABLE KEY_GENERIC iterable) {
  if (iterable == null) throw new NullPointerException();
  this.iterable = iterable;
 }
```

```
 @Override
 public int size() {
  int c = 0;
  final KEY_ITERATOR KEY_GENERIC iterator = iterator();
  while(iterator.hasNext()) {
   iterator.NEXT_KEY();
   c++;
  }

  return c;
 }

 @Override
 public boolean isEmpty() { return ! iterable.iterator().hasNext(); }

 @Override
 public KEY_ITERATOR KEY_GENERIC iterator() { return iterable.iterator(); }
}


 /** Returns an unmodifiable collection backed by the specified iterable.
  *
  * @param iterable the iterable object to be wrapped in an unmodifiable collection.
  * @return an unmodifiable collection view of the specified iterable.
  */
 public static KEY_GENERIC COLLECTION KEY_GENERIC asCollection(final KEY_ITERABLE
KEY_GENERIC iterable) {
  if (iterable instanceof COLLECTION) return (COLLECTION KEY_GENERIC)iterable;
  return new IterableCollection KEY_GENERIC_DIAMOND(iterable);
 }

}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/Collections.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
```

```
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import it.unimi.dsi.fastutil.Hash;
import it.unimi.dsi.fastutil.HashCommon;
import static it.unimi.dsi.fastutil.HashCommon.arraySize;
import static it.unimi.dsi.fastutil.HashCommon.maxFill;

import java.util.Arrays;
import java.util.Collection;
import java.util.Iterator;
import java.util.NoSuchElementException;

#ifdef Linked

#if KEYS_REFERENCE
import java.util.Comparator;
#endif

/**  A type-specific linked hash set with with a fast, small-footprint implementation.
*
* <p>Instances of this class use a hash table to represent a set. The table is
* filled up to a specified <em>load factor</em>, and then doubled in size to
* accommodate new entries. If the table is emptied below <em>one fourth</em>
* of the load factor, it is halved in size; however, the table is never reduced to a
* size smaller than that at creation time: this approach makes it
* possible to create sets with a large capacity in which insertions and
* deletions do not cause immediately rehashing. Moreover, halving is
* not performed when deleting entries from an iterator, as it would interfere
* with the iteration process.
*
* <p>Note that {@link #clear()} does not modify the hash table size.
* Rather, a family of {@linkplain #trim() trimming
* methods} lets you control the size of the table; this is particularly useful
* if you reuse instances of this class.
*
* <p>Iterators generated by this set will enumerate elements in the same order in which they
* have been added to the set (addition of elements already present
* in the set does not change the iteration order). Note that this order has nothing in common with the natural
* order of the keys. The order is kept by means of a doubly linked list, represented
* <i>via</i> an array of longs parallel to the table.
*
```

```
* <p>This class implements the interface of a sorted set, so to allow easy
* access of the iteration order: for instance, you can get the first element
* in iteration order with {@code first()} without having to create an
* iterator; however, this class partially violates the {@link java.util.SortedSet}
* contract because all subset methods throw an exception and {@link
* #comparator()} returns always {@code null}.
*
* <p>Additional methods, such as {@code addAndMoveToFirst()}, make it easy
* to use instances of this class as a cache (e.g., with LRU policy).
*
* <p>The iterators provided by this class are type-specific {@linkplain
* java.util.ListIterator list iterators}, and can be started at any
* element <em>which is in the set</em> (if the provided element
* is not in the set, a {@link NoSuchElementException} exception will be thrown).
* If, however, the provided element is not the first or last element in the
* set, the first access to the list index will require linear time, as in the worst case
* the entire set must be scanned in iteration order to retrieve the positional
* index of the starting element. If you use just the methods of a type-specific {@link
it.unimi.dsi.fastutil.BidirectionalIterator},
* however, all operations will be performed in constant time.
*
* @see Hash
* @see HashCommon
*/

public class OPEN_HASH_SET KEY_GENERIC extends ABSTRACT_SORTED_SET KEY_GENERIC
implements java.io.Serializable, Cloneable, Hash {

#else

#ifdef Custom

/** A type-specific hash set with a fast, small-footprint implementation whose {@linkplain
it.unimi.dsi.fastutil.Hash.Strategy hashing strategy}
* is specified at creation time.
*
* <p>Instances of this class use a hash table to represent a set. The table is
* filled up to a specified <em>load factor</em>, and then doubled in size to
* accommodate new entries. If the table is emptied below <em>one fourth</em>
* of the load factor, it is halved in size; however, the table is never reduced to a
* size smaller than that at creation time: this approach makes it
* possible to create sets with a large capacity in which insertions and
* deletions do not cause immediately rehashing. Moreover, halving is
* not performed when deleting entries from an iterator, as it would interfere
* with the iteration process.
*
* <p>Note that {@link #clear()} does not modify the hash table size.
* Rather, a family of {@linkplain #trim() trimming
```

```
* methods} lets you control the size of the table; this is particularly useful
* if you reuse instances of this class.
*
* @see Hash
* @see HashCommon
*/

public class OPEN_HASH_SET KEY_GENERIC extends ABSTRACT_SET KEY_GENERIC implements
java.io.Serializable, Cloneable, Hash {

#else

/**  A type-specific hash set with with a fast, small-footprint implementation.
*
* <p>Instances of this class use a hash table to represent a set. The table is
* filled up to a specified <em>load factor</em>, and then doubled in size to
* accommodate new entries. If the table is emptied below <em>one fourth</em>
* of the load factor, it is halved in size; however, the table is never reduced to a
* size smaller than that at creation time: this approach makes it
* possible to create sets with a large capacity in which insertions and
* deletions do not cause immediately rehashing. Moreover, halving is
* not performed when deleting entries from an iterator, as it would interfere
* with the iteration process.
*
* <p>Note that {@link #clear()} does not modify the hash table size.
* Rather, a family of {@linkplain #trim() trimming
* methods} lets you control the size of the table; this is particularly useful
* if you reuse instances of this class.
*
* @see Hash
* @see HashCommon
*/

public class OPEN_HASH_SET KEY_GENERIC extends ABSTRACT_SET KEY_GENERIC implements
java.io.Serializable, Cloneable, Hash {

#endif

#endif

 private static final long serialVersionUID = 0L;
 private static final boolean ASSERTS = ASSERTS_VALUE;

 /** The array of keys. */
 protected transient KEY_GENERIC_TYPE[] key;

 /** The mask for wrapping a position counter. */
 protected transient int mask;
```

```
 /** Whether this set contains the null key. */
 protected transient boolean containsNull;

#ifdef Custom
 /** The hash strategy of this custom set. */
 protected STRATEGY KEY_SUPER_GENERIC strategy;
#endif

#ifdef Linked
 /** The index of the first entry in iteration order. It is valid iff {@link #size} is nonzero; otherwise, it contains -1. */
 protected transient int first = -1;
 /** The index of the last entry in iteration order. It is valid iff {@link #size} is nonzero; otherwise, it contains -1. */
 protected transient int last = -1;
 /** For each entry, the next and the previous entry in iteration order,
  * stored as {@code ((prev & 0xFFFFFFFFL) << 32) | (next & 0xFFFFFFFFL)}.
  * The first entry contains predecessor -1, and the last entry
  * contains successor -1. */
 protected transient long[] link;
#endif

 /** The current table size. Note that an additional element is allocated for storing the null key. */
 protected transient int n;

 /** Threshold after which we rehash. It must be the table size times {@link #f}. */
 protected transient int maxFill;

 /** We never resize below this threshold, which is the construction-time {#n}. */
 protected final transient int minN;

 /** Number of entries in the set (including the null key, if present). */
 protected int size;

 /** The acceptable load factor. */
 protected final float f;

#ifdef Custom
 /** Creates a new hash set.
  *
  * <p>The actual table size will be the least power of two greater than {@code expected}/{@code f}.
  *
  * @param expected the expected number of elements in the hash set.
  * @param f the load factor.
  * @param strategy the strategy.
  */
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public OPEN_HASH_SET(final int expected, final float f, final STRATEGY KEY_SUPER_GENERIC strategy) {
  this.strategy = strategy;
```

```
#else
 /** Creates a new hash set.
  *
  * <p>The actual table size will be the least power of two greater than {@code expected}/{@code f}.
  *
  * @param expected the expected number of elements in the hash set.
  * @param f the load factor.
  */
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public OPEN_HASH_SET(final int expected, final float f) {
#endif
  if (f <= 0 || f > 1) throw new IllegalArgumentException("Load factor must be greater than 0 and smaller than or
equal to 1");
  if (expected < 0) throw new IllegalArgumentException("The expected number of elements must be nonnegative");

  this.f = f;

  minN = n = arraySize(expected, f);
  mask = n - 1;
  maxFill = maxFill(n, f);
  key = KEY_GENERIC_ARRAY_CAST new KEY_TYPE[n + 1];
#ifdef Linked
  link = new long[n + 1];
#endif
 }


#ifdef Custom
 /** Creates a new hash set with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor.
  *
  * @param expected the expected number of elements in the hash set.
  * @param strategy the strategy.
  */

 public OPEN_HASH_SET(final int expected, final STRATEGY KEY_SUPER_GENERIC strategy) {
  this(expected, DEFAULT_LOAD_FACTOR, strategy);
 }

#else
 /** Creates a new hash set with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor.
  *
  * @param expected the expected number of elements in the hash set.
  */

 public OPEN_HASH_SET(final int expected) {
  this(expected, DEFAULT_LOAD_FACTOR);
 }
```

```
#endif


#ifdef Custom
 /** Creates a new hash set with initial expected {@link Hash#DEFAULT_INITIAL_SIZE} elements
  * and {@link Hash#DEFAULT_LOAD_FACTOR} as load factor.
  * @param strategy the strategy.
  */

 public OPEN_HASH_SET(final STRATEGY KEY_SUPER_GENERIC strategy) {
 this(DEFAULT_INITIAL_SIZE, DEFAULT_LOAD_FACTOR, strategy);
 }
#else
 /** Creates a new hash set with initial expected {@link Hash#DEFAULT_INITIAL_SIZE} elements
  * and {@link Hash#DEFAULT_LOAD_FACTOR} as load factor.
  */

 public OPEN_HASH_SET() {
  this(DEFAULT_INITIAL_SIZE, DEFAULT_LOAD_FACTOR);
 }
#endif


#ifdef Custom
 /** Creates a new hash set copying a given collection.
  *
  * @param c a {@link Collection} to be copied into the new hash set.
  * @param f the load factor.
  * @param strategy the strategy.
  */

 public OPEN_HASH_SET(final Collection<? extends KEY_GENERIC_CLASS> c, final float f, final STRATEGY
KEY_SUPER_GENERIC strategy) {
 this(c.size(), f, strategy);
  addAll(c);
 }
#else
 /** Creates a new hash set copying a given collection.
  *
  * @param c a {@link Collection} to be copied into the new hash set.
  * @param f the load factor.
  */

 public OPEN_HASH_SET(final Collection<? extends KEY_GENERIC_CLASS> c, final float f) {
  this(c.size(), f);
  addAll(c);
 }
#endif
```

```
#ifdef Custom
 /** Creates a new hash set  with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor
  * copying a given collection.
  *
  * @param c a {@link Collection} to be copied into the new hash set.
  * @param strategy the strategy.
  */

 public OPEN_HASH_SET(final Collection<? extends KEY_GENERIC_CLASS> c, final STRATEGY
KEY_SUPER_GENERIC strategy) {
  this(c, DEFAULT_LOAD_FACTOR, strategy);
 }

#else
 /** Creates a new hash set  with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor
  * copying a given collection.
  *
  * @param c a {@link Collection} to be copied into the new hash set.
  */

 public OPEN_HASH_SET(final Collection<? extends KEY_GENERIC_CLASS> c) {
  this(c, DEFAULT_LOAD_FACTOR);
 }
#endif


#ifdef Custom
 /** Creates a new hash set copying a given type-specific collection.
  *
  * @param c a type-specific collection to be copied into the new hash set.
  * @param f the load factor.
  * @param strategy the strategy.
  */

 public OPEN_HASH_SET(final COLLECTION KEY_EXTENDS_GENERIC c, final float f, STRATEGY
KEY_SUPER_GENERIC strategy) {
 this(c.size(), f, strategy);
  addAll(c);
 }
#else
 /** Creates a new hash set copying a given type-specific collection.
  *
  * @param c a type-specific collection to be copied into the new hash set.
  * @param f the load factor.
  */
```

```
 public OPEN_HASH_SET(final COLLECTION KEY_EXTENDS_GENERIC c, final float f) {
  this(c.size(), f);
  addAll(c);
 }
#endif


#ifdef Custom
 /** Creates a new hash set  with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor
  * copying a given type-specific collection.
  *
  * @param c a type-specific collection to be copied into the new hash set.
  * @param strategy the strategy.
  */

 public OPEN_HASH_SET(final COLLECTION KEY_EXTENDS_GENERIC c, final STRATEGY
KEY_SUPER_GENERIC strategy) {
  this(c, DEFAULT_LOAD_FACTOR, strategy);
 }
#else
 /** Creates a new hash set  with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor
  * copying a given type-specific collection.
  *
  * @param c a type-specific collection to be copied into the new hash set.
  */

 public OPEN_HASH_SET(final COLLECTION KEY_EXTENDS_GENERIC c) {
  this(c, DEFAULT_LOAD_FACTOR);
 }
#endif


#ifdef Custom
 /** Creates a new hash set using elements provided by a type-specific iterator.
  *
  * @param i a type-specific iterator whose elements will fill the set.
  * @param f the load factor.
  * @param strategy the strategy.
  */

 public OPEN_HASH_SET(final STD_KEY_ITERATOR KEY_EXTENDS_GENERIC i, final float f, final
STRATEGY KEY_SUPER_GENERIC strategy) {
  this(DEFAULT_INITIAL_SIZE, f, strategy);
  while(i.hasNext()) add(i.NEXT_KEY());
 }
#else
 /** Creates a new hash set using elements provided by a type-specific iterator.
```

```
 *
 * @param i a type-specific iterator whose elements will fill the set.
 * @param f the load factor.
 */

 public OPEN_HASH_SET(final STD_KEY_ITERATOR KEY_EXTENDS_GENERIC i, final float f) {
  this(DEFAULT_INITIAL_SIZE, f);
  while(i.hasNext()) add(i.NEXT_KEY());
 }
#endif

#ifdef Custom
 /** Creates a new hash set with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor using elements
provided by a type-specific iterator.
 *
 * @param i a type-specific iterator whose elements will fill the set.
 * @param strategy the strategy.
 */

 public OPEN_HASH_SET(final STD_KEY_ITERATOR KEY_EXTENDS_GENERIC i, final STRATEGY
KEY_SUPER_GENERIC strategy) {
  this(i, DEFAULT_LOAD_FACTOR, strategy);
 }
#else
 /** Creates a new hash set with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor using elements
provided by a type-specific iterator.
 *
 * @param i a type-specific iterator whose elements will fill the set.
 */

 public OPEN_HASH_SET(final STD_KEY_ITERATOR KEY_EXTENDS_GENERIC i) {
  this(i, DEFAULT_LOAD_FACTOR);
 }
#endif



#if KEYS_PRIMITIVE

#ifdef Custom
 /** Creates a new hash set using elements provided by an iterator.
 *
 * @param i an iterator whose elements will fill the set.
 * @param f the load factor.
 * @param strategy the strategy.
 */

 public OPEN_HASH_SET(final Iterator<?> i, final float f, final STRATEGY KEY_SUPER_GENERIC strategy) {
```

```
 this(ITERATORS.AS_KEY_ITERATOR(i), f, strategy);
 }
#else
 /** Creates a new hash set using elements provided by an iterator.
  *
  * @param i an iterator whose elements will fill the set.
  * @param f the load factor.
  */

 public OPEN_HASH_SET(final Iterator<?> i, final float f) {
  this(ITERATORS.AS_KEY_ITERATOR(i), f);
 }

#endif


#ifdef Custom
 /** Creates a new hash set with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor using elements
provided by an iterator.
  *
  * @param i an iterator whose elements will fill the set.
  * @param strategy the strategy.
  */

 public OPEN_HASH_SET(final Iterator<?> i, final STRATEGY KEY_SUPER_GENERIC strategy) {
  this(ITERATORS.AS_KEY_ITERATOR(i), strategy);
 }
#else
 /** Creates a new hash set with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor using elements
provided by an iterator.
  *
  * @param i an iterator whose elements will fill the set.
  */

 public OPEN_HASH_SET(final Iterator<?> i) {
  this(ITERATORS.AS_KEY_ITERATOR(i));
 }
#endif

#endif



#ifdef Custom
 /** Creates a new hash set and fills it with the elements of a given array.
  *
  * @param a an array whose elements will be used to fill the set.
  * @param offset the first element to use.
```

```
  * @param length the number of elements to use.
  * @param f the load factor.
  * @param strategy the strategy.
  */

 public OPEN_HASH_SET(final KEY_GENERIC_TYPE[] a, final int offset, final int length, final float f, final
STRATEGY KEY_SUPER_GENERIC strategy) {
 this(length < 0 ? 0 : length, f, strategy);
  ARRAYS.ensureOffsetLength(a, offset, length);
  for(int i = 0; i < length; i++) add(a[offset + i]);
 }
#else
 /** Creates a new hash set and fills it with the elements of a given array.
  *
  * @param a an array whose elements will be used to fill the set.
  * @param offset the first element to use.
  * @param length the number of elements to use.
  * @param f the load factor.
  */

 public OPEN_HASH_SET(final KEY_GENERIC_TYPE[] a, final int offset, final int length, final float f) {
  this(length < 0 ? 0 : length, f);
  ARRAYS.ensureOffsetLength(a, offset, length);
  for(int i = 0; i < length; i++) add(a[offset + i]);
 }
#endif

#ifdef Custom
 /** Creates a new hash set with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor and fills it with the
elements of a given array.
  *
  * @param a an array whose elements will be used to fill the set.
  * @param offset the first element to use.
  * @param length the number of elements to use.
  * @param strategy the strategy.
  */

 public OPEN_HASH_SET(final KEY_GENERIC_TYPE[] a, final int offset, final int length, final STRATEGY
KEY_SUPER_GENERIC strategy) {
  this(a, offset, length, DEFAULT_LOAD_FACTOR, strategy);
 }
#else
 /** Creates a new hash set with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor and fills it with the
elements of a given array.
  *
  * @param a an array whose elements will be used to fill the set.
  * @param offset the first element to use.
  * @param length the number of elements to use.
```

```
 */

 public OPEN_HASH_SET(final KEY_GENERIC_TYPE[] a, final int offset, final int length) {
  this(a, offset, length, DEFAULT_LOAD_FACTOR);
 }
#endif

#ifdef Custom
 /** Creates a new hash set copying the elements of an array.
  *
  * @param a an array to be copied into the new hash set.
  * @param f the load factor.
  * @param strategy the strategy.
  */

 public OPEN_HASH_SET(final KEY_GENERIC_TYPE[] a, final float f, final STRATEGY
KEY_SUPER_GENERIC strategy) {
  this(a, 0, a.length, f, strategy);
 }
#else
 /** Creates a new hash set copying the elements of an array.
  *
  * @param a an array to be copied into the new hash set.
  * @param f the load factor.
  */

 public OPEN_HASH_SET(final KEY_GENERIC_TYPE[] a, final float f) {
  this(a, 0, a.length, f);
 }
#endif

#ifdef Custom
 /** Creates a new hash set with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor
  * copying the elements of an array.
  *
  * @param a an array to be copied into the new hash set.
  * @param strategy the strategy.
  */

 public OPEN_HASH_SET(final KEY_GENERIC_TYPE[] a, final STRATEGY KEY_SUPER_GENERIC
strategy) {
  this(a, DEFAULT_LOAD_FACTOR, strategy);
 }
#else
 /** Creates a new hash set with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor
  * copying the elements of an array.
  *
  * @param a an array to be copied into the new hash set.
```

```
 */

 public OPEN_HASH_SET(final KEY_GENERIC_TYPE[] a) {
  this(a, DEFAULT_LOAD_FACTOR);
 }
#endif


#ifdef Custom
 /** Returns the hashing strategy.
  *
  * @return the hashing strategy of this custom hash set.
  */

 public STRATEGY KEY_SUPER_GENERIC strategy() {
  return strategy;
 }
#endif

 private int realSize() {
  return containsNull ? size - 1 : size;
 }

 private void ensureCapacity(final int capacity) {
  final int needed = arraySize(capacity, f);
  if (needed > n) rehash(needed);
 }

 private void tryCapacity(final long capacity) {
  final int needed = (int)Math.min(1 << 30, Math.max(2, HashCommon.nextPowerOfTwo((long)Math.ceil(capacity / f))));
  if (needed > n) rehash(needed);
 }

#if KEYS_PRIMITIVE
 @Override
 public boolean addAll(COLLECTION c) {
  if (f <= .5) ensureCapacity(c.size()); // The resulting collection will be sized for c.size() elements
  else tryCapacity(size() + c.size()); // The resulting collection will be tentatively sized for size() + c.size() elements
  return super.addAll(c);
 }
#endif

 @Override
 public boolean addAll(Collection<? extends KEY_GENERIC_CLASS> c) {
  // The resulting collection will be at least c.size() big
  if (f <= .5) ensureCapacity(c.size()); // The resulting collection will be sized for c.size() elements
  else tryCapacity(size() + c.size()); // The resulting collection will be tentatively sized for size() + c.size() elements
```

```
 return super.addAll(c);
 }


 @Override
 public boolean add(final KEY_GENERIC_TYPE k) {
  int pos;

  if (KEY_EQUALS_NULL(k)) {
   if (containsNull) return false;
#ifdef Linked
   pos = n;
#endif
   containsNull = true;
#ifdef Custom
   key[n] = k;
#endif
  }
  else {
   KEY_GENERIC_TYPE curr;
   final KEY_GENERIC_TYPE[] key = this.key;

   // The starting point.
   if (! KEY_IS_NULL(curr = key[pos = KEY2INTHASH(k) & mask])) {
    if (KEY_EQUALS_NOT_NULL(curr, k)) return false;
    while(! KEY_IS_NULL(curr = key[pos = (pos + 1) & mask]))
     if (KEY_EQUALS_NOT_NULL(curr, k)) return false;
   }
   key[pos] = k;
  }

#ifdef Linked
  if (size == 0) {
   first = last = pos;
   // Special case of SET_UPPER_LOWER(link[pos], -1, -1);
   link[pos] = -1L;
  }
  else {
   SET_NEXT(link[last], pos);
   SET_UPPER_LOWER(link[pos], last, -1);
   last = pos;
  }
#endif

  if (size++ >= maxFill) rehash(arraySize(size + 1, f));
  if (ASSERTS) checkTable();
  return true;
 }
```

```
#if KEY_CLASS_Object
 /** Add a random element if not present, get the existing value if already present.
  *
  * This is equivalent to (but faster than) doing a:
  * <pre>
  * K exist = set.get(k);
  * if (exist == null) {
  *   set.add(k);
  *   exist = k;
  * }
  * </pre>
  */
 public KEY_GENERIC_TYPE addOrGet(final KEY_GENERIC_TYPE k) {
  int pos;

  if (KEY_EQUALS_NULL(k)) {
   if (containsNull) return key [n];
#ifdef Linked
   pos = n;
#endif
   containsNull = true;
#ifdef Custom
   key [n] = k;
#endif
  }
  else {
   KEY_GENERIC_TYPE curr;
   final KEY_GENERIC_TYPE[] key = this.key;

   // The starting point.
   if (! KEY_IS_NULL(curr = key[pos = KEY2INTHASH(k) & mask])) {
    if (KEY_EQUALS_NOT_NULL(curr, k)) return curr;
    while(! KEY_IS_NULL(curr = key[pos = (pos + 1) & mask]))
     if (KEY_EQUALS_NOT_NULL(curr, k)) return curr;
   }
   key[pos] = k;
  }

#ifdef Linked
  if (size == 0) {
   first = last = pos;
   // Special case of SET_UPPER_LOWER(link[pos], -1, -1);
   link[pos] = -1L;
  }
  else {
   SET_NEXT(link[last], pos);
   SET_UPPER_LOWER(link[pos], last, -1);
```

```
   last = pos;
  }
#endif

  if (size++ >= maxFill) rehash(arraySize(size + 1, f));
  if (ASSERTS) checkTable();
  return k;
 }
#endif

 /** Shifts left entries with the specified hash code, starting at the specified position,
  * and empties the resulting free entry.
  *
  * @param pos a starting position.
  */
 protected final void shiftKeys(int pos) {
  // Shift entries with the same hash.
  int last, slot;
  KEY_GENERIC_TYPE curr;
  final KEY_GENERIC_TYPE[] key = this.key;

  for(;;) {
   pos = ((last = pos) + 1) & mask;

   for(;;) {
    if (KEY_IS_NULL(curr = key[pos])) {
     key[last] = KEY_NULL;
     return;
    }
    slot = KEY2INTHASH(curr) & mask;
    if (last <= pos ? last >= slot || slot > pos : last >= slot && slot > pos) break;
    pos = (pos + 1) & mask;
   }

   key[last] = curr;
#ifdef Linked
   fixPointers(pos, last);
#endif
  }
 }

 private boolean removeEntry(final int pos) {
  size--;
#ifdef Linked
  fixPointers(pos);
#endif
  shiftKeys(pos);
  if (n > minN && size < maxFill / 4 && n > DEFAULT_INITIAL_SIZE) rehash(n / 2);
```

```
 return true;
 }

 private boolean removeNullEntry() {
  containsNull = false;
  key[n] = KEY_NULL;
  size--;
#ifdef Linked
  fixPointers(n);
#endif
  if (n > minN && size < maxFill / 4 && n > DEFAULT_INITIAL_SIZE) rehash(n / 2);
  return true;
 }

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 @Override
 public boolean remove(final KEY_TYPE k) {
  if (KEY_EQUALS_NULL(KEY_GENERIC_CAST k)) {
   if (containsNull) return removeNullEntry();
   return false;
  }

  KEY_GENERIC_TYPE curr;
  final KEY_GENERIC_TYPE[] key = this.key;
  int pos;

  // The starting point.
  if (KEY_IS_NULL(curr = key[pos = KEY2INTHASH_CAST(k) & mask])) return false;
  if (KEY_EQUALS_NOT_NULL_CAST(k, curr)) return removeEntry(pos);

  while(true) {
   if (KEY_IS_NULL(curr = key[pos = (pos + 1) & mask])) return false;
   if (KEY_EQUALS_NOT_NULL_CAST(k, curr)) return removeEntry(pos);
  }
 }

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 @Override
 public boolean contains(final KEY_TYPE k) {
  if (KEY_EQUALS_NULL(KEY_GENERIC_CAST k)) return containsNull;

  KEY_GENERIC_TYPE curr;
  final KEY_GENERIC_TYPE[] key = this.key;
  int pos;

  // The starting point.
  if (KEY_IS_NULL(curr = key[pos = KEY2INTHASH_CAST(k) & mask])) return false;
  if (KEY_EQUALS_NOT_NULL_CAST(k, curr)) return true;
```

```
  while(true) {
   if (KEY_IS_NULL(curr = key[pos = (pos + 1) & mask])) return false;
   if (KEY_EQUALS_NOT_NULL_CAST(k, curr)) return true;
  }
 }

#if KEY_CLASS_Object
 /** Returns the element of this set that is equal to the given key, or {@code null}.
  * @return the element of this set that is equal to the given key, or {@code null}.
  */
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public K get(final Object k) {
  if (KEY_EQUALS_NULL(KEY_GENERIC_CAST k)) return key[n]; // This is correct independently of the value
of containsNull and of the set being custom

  KEY_GENERIC_TYPE curr;
  final KEY_GENERIC_TYPE[] key = this.key;
  int pos;

  // The starting point.
  if (KEY_IS_NULL(curr = key[pos = KEY2INTHASH_CAST(k) & mask])) return null;
  if (KEY_EQUALS_NOT_NULL_CAST(k, curr)) return curr;
  // There's always an unused entry.
  while(true) {
   if (KEY_IS_NULL(curr = key[pos = (pos + 1) & mask])) return null;
   if (KEY_EQUALS_NOT_NULL_CAST(k, curr)) return curr;
  }
 }
#endif

#ifdef Linked

 /** Removes the first key in iteration order.
  * @return the first key.
  * @throws NoSuchElementException is this set is empty.
  */
 public KEY_GENERIC_TYPE REMOVE_FIRST_KEY() {
  if (size == 0) throw new NoSuchElementException();
  final int pos = first;
  // Abbreviated version of fixPointers(pos)
  first = GET_NEXT(link[pos]);
  if (0 <= first) {
   // Special case of SET_PREV(link[first], -1)
   link[first] |= (-1 & 0xFFFFFFFFL) << 32;
  }
  final KEY_GENERIC_TYPE k = key[pos];
  size--;
```

```
 if (KEY_EQUALS_NULL(k)) {
  containsNull = false;
  key[n] = KEY_NULL;
 }
 else shiftKeys(pos);
 if (n > minN && size < maxFill / 4 && n > DEFAULT_INITIAL_SIZE) rehash(n / 2);
 return k;
}

/** Removes the the last key in iteration order.
 * @return the last key.
 * @throws NoSuchElementException is this set is empty.
 */
public KEY_GENERIC_TYPE REMOVE_LAST_KEY() {
 if (size == 0) throw new NoSuchElementException();
 final int pos = last;
 // Abbreviated version of fixPointers(pos)
 last = GET_PREV(link[pos]);
 if (0 <= last) {
  // Special case of SET_NEXT(link[last], -1)
  link[last] |= -1 & 0xFFFFFFFFL;
 }
 final KEY_GENERIC_TYPE k = key[pos];
 size--;
 if (KEY_EQUALS_NULL(k)) {
  containsNull = false;
  key[n] = KEY_NULL;
 }
 else shiftKeys(pos);
 if (n > minN && size < maxFill / 4 && n > DEFAULT_INITIAL_SIZE) rehash(n / 2);
 return k;
}

private void moveIndexToFirst(final int i) {
 if (size == 1 || first == i) return;
 if (last == i) {
  last = GET_PREV(link[i]);
  // Special case of SET_NEXT(link[last], -1);
  link[last] |= -1 & 0xFFFFFFFFL;
 }
 else {
  final long linki = link[i];
  final int prev = GET_PREV(linki);
  final int next = GET_NEXT(linki);
  COPY_NEXT(link[prev], linki);
  COPY_PREV(link[next], linki);
 }
 SET_PREV(link[first], i);
```

```
 SET_UPPER_LOWER(link[i], -1, first);
 first = i;
}

private void moveIndexToLast(final int i) {
 if (size == 1 ||  last == i) return;
 if (first == i) {
  first = GET_NEXT(link[i]);
  // Special case of SET_PREV(link[first], -1);
  link[first] |= (-1 & 0xFFFFFFFFL) << 32;
 }
 else {
  final long linki = link[i];
  final int prev = GET_PREV(linki);
  final int next = GET_NEXT(linki);
  COPY_NEXT(link[prev], linki);
  COPY_PREV(link[next], linki);
 }
 SET_NEXT(link[last], i);
 SET_UPPER_LOWER(link[i], last, -1);
 last = i;
}

/** Adds a key to the set; if the key is already present, it is moved to the first position of the iteration order.
 *
 * @param k the key.
 * @return true if the key was not present.
 */
public boolean addAndMoveToFirst(final KEY_GENERIC_TYPE k) {
 int pos;

 if (KEY_EQUALS_NULL(k)) {
  if (containsNull) {
   moveIndexToFirst(n);
   return false;
  }
  containsNull = true;
  pos = n;
 }
 else {
  // The starting point.
  final KEY_GENERIC_TYPE key[] = this.key;
  pos = KEY2INTHASH(k) & mask;

  // There's always an unused entry. TODO
  while(! KEY_IS_NULL(key[pos])) {
   if (KEY_EQUALS_NOT_NULL(k, key[pos])) {
    moveIndexToFirst(pos);
```

```
   return false;
  }

  pos = (pos + 1) & mask;
 }
}

key[pos] = k;

if (size == 0) {
 first = last = pos;
 // Special case of SET_UPPER_LOWER(link[pos], -1, -1);
 link[pos] = -1L;
}
else {
 SET_PREV(link[first], pos);
 SET_UPPER_LOWER(link[pos], -1, first);
 first = pos;
}

if (size++ >= maxFill) rehash(arraySize(size, f));
if (ASSERTS) checkTable();
return true;
}

/** Adds a key to the set; if the key is already present, it is moved to the last position of the iteration order.
 *
 * @param k the key.
 * @return true if the key was not present.
 */
public boolean addAndMoveToLast(final KEY_GENERIC_TYPE k) {
 int pos;

 if (KEY_EQUALS_NULL(k)) {
  if (containsNull) {
   moveIndexToLast(n);
   return false;
  }
  containsNull = true;
  pos = n;
 }
 else {
  // The starting point.
  final KEY_GENERIC_TYPE key[] = this.key;
  pos = KEY2INTHASH(k) & mask;

  // There's always an unused entry.
  while(! KEY_IS_NULL(key[pos])) {
```

```
   if (KEY_EQUALS_NOT_NULL(k, key[pos])) {
    moveIndexToLast(pos);
    return false;
   }

   pos = (pos + 1) & mask;
  }
 }

 key[pos] = k;

 if (size == 0) {
  first = last = pos;
  // Special case of SET_UPPER_LOWER(link[pos], -1, -1);
  link[pos] = -1L;
 }
 else {
  SET_NEXT(link[last], pos);
  SET_UPPER_LOWER(link[pos], last, -1);
  last = pos;
 }

 if (size++ >= maxFill) rehash(arraySize(size, f));
 if (ASSERTS) checkTable();
 return true;
}

#endif

 /* Removes all elements from this set.
  *
  * <p>To increase object reuse, this method does not change the table size.
  * If you want to reduce the table size, you must use {@link #trim()}.
  *
  */
 @Override
 public void clear() {
  if (size == 0) return;
  size = 0;
  containsNull = false;
  Arrays.fill(key, KEY_NULL);
#ifdef Linked
  first = last = -1;
#endif
 }


 @Override
```

```
 public int size() {
  return size;
 }

 @Override
 public boolean isEmpty() {
  return size == 0;
 }

#ifdef Linked

 /** Modifies the {@link #link} vector so that the given entry is removed.
  * This method will complete in constant time.
  *
  * @param i the index of an entry.
  */
 protected void fixPointers(final int i) {
  if (size == 0) {
   first = last = -1;
   return;
  }
  if (first == i) {
   first = GET_NEXT(link[i]);
   if (0 <= first) {
    // Special case of SET_PREV(link[first], -1)
    link[first] |= (-1 & 0xFFFFFFFFL) << 32;
   }
   return;
  }
  if (last == i) {
   last = GET_PREV(link[i]);
   if (0 <= last) {
    // Special case of SET_NEXT(link[last], -1)
    link[last] |= -1 & 0xFFFFFFFFL;
   }
   return;
  }
  final long linki = link[i];
  final int prev = GET_PREV(linki);
  final int next = GET_NEXT(linki);
  COPY_NEXT(link[prev], linki);
  COPY_PREV(link[next], linki);
 }


 /** Modifies the {@link #link} vector for a shift from s to d.
  * This method will complete in constant time.
  *
```

```
 * @param s the source position.
 * @param d the destination position.
 */
protected void fixPointers(int s, int d) {
 if (size == 1) {
  first = last = d;
  // Special case of SET(link[d], -1, -1)
  link[d] = -1L;
  return;
 }
 if (first == s) {
  first = d;
  SET_PREV(link[GET_NEXT(link[s])], d);
  link[d] = link[s];
  return;
 }
 if (last == s) {
  last = d;
  SET_NEXT(link[GET_PREV(link[s])], d);
  link[d] = link[s];
  return;
 }
 final long links = link[s];
 final int prev = GET_PREV(links);
 final int next = GET_NEXT(links);
 SET_NEXT(link[prev], d);
 SET_PREV(link[next], d);
 link[d] = links;
}


/** Returns the first element of this set in iteration order.
 *
 * @return the first element in iteration order.
 */
@Override
public KEY_GENERIC_TYPE FIRST() {
 if (size == 0) throw new NoSuchElementException();
 return key[first];
}

/** Returns the last element of this set in iteration order.
 *
 * @return the last element in iteration order.
 */
@Override
public KEY_GENERIC_TYPE LAST() {
 if (size == 0) throw new NoSuchElementException();
```

```
 return key[last];
 }

 /** {@inheritDoc}
  * <p>This implementation just throws an {@link UnsupportedOperationException}.*/
 @Override
 public SORTED_SET KEY_GENERIC tailSet(KEY_GENERIC_TYPE from) { throw new
UnsupportedOperationException(); }

 /** {@inheritDoc}
  * <p>This implementation just throws an {@link UnsupportedOperationException}.*/
 @Override
 public SORTED_SET KEY_GENERIC headSet(KEY_GENERIC_TYPE to) { throw new
UnsupportedOperationException(); }

 /** {@inheritDoc}
  * <p>This implementation just throws an {@link UnsupportedOperationException}.*/
 @Override
 public SORTED_SET KEY_GENERIC subSet(KEY_GENERIC_TYPE from, KEY_GENERIC_TYPE to) { throw
new UnsupportedOperationException(); }

 /** {@inheritDoc}
  * <p>This implementation just returns {@code null}.*/
 @Override
 public KEY_COMPARATOR KEY_SUPER_GENERIC comparator() { return null; }


 /** A list iterator over a linked set.
  *
  * <p>This class provides a list iterator over a linked hash set. The constructor runs in constant time.
  */
 private class SetIterator implements KEY_LIST_ITERATOR KEY_GENERIC {
  /** The entry that will be returned by the next call to {@link java.util.ListIterator#previous()} (or {@code null} if
no previous entry exists). */
  int prev = -1;
  /** The entry that will be returned by the next call to {@link java.util.ListIterator#next()} (or {@code null} if no
next entry exists). */
  int next = -1;
  /** The last entry that was returned (or -1 if we did not iterate or used {@link #remove()}). */
  int curr = -1;
  /** The current index (in the sense of a {@link java.util.ListIterator}). When -1, we do not know the current
index.*/
  int index = -1;

  SetIterator() {
   next = first;
   index = 0;
  }
```

```
SetIterator(KEY_GENERIC_TYPE from) {
 if (KEY_EQUALS_NULL(from)) {
  if (OPEN_HASH_SET.this.containsNull) {
   next = GET_NEXT(link[n]);
   prev = n;
   return;
  }
  else throw new NoSuchElementException("The key " + from + " does not belong to this set.");
 }

 if (KEY_EQUALS(key[last], from)) {
  prev = last;
  index = size;
  return;
 }

 // The starting point.
 final KEY_GENERIC_TYPE key[] = OPEN_HASH_SET.this.key;
 int pos = KEY2INTHASH(from) & mask;

 // There's always an unused entry.
 while(! KEY_IS_NULL(key[pos])) {
  if (KEY_EQUALS_NOT_NULL(key[pos], from)) {
   // Note: no valid index known.
   next = GET_NEXT(link[pos]);
   prev = pos;
   return;
  }
  pos = (pos + 1) & mask;
 }
 throw new NoSuchElementException("The key " + from + " does not belong to this set.");
}

@Override
public boolean hasNext() { return next != -1; }
@Override
public boolean hasPrevious() { return prev != -1; }

@Override
public KEY_GENERIC_TYPE NEXT_KEY() {
 if (! hasNext()) throw new NoSuchElementException();

 curr = next;
 next = GET_NEXT(link[curr]);
 prev = curr;

 if (index >= 0) index++;
```

```
 if (ASSERTS) assert curr == n || ! KEY_IS_NULL(key[curr]) : "Position " + curr + " is not used";
 return key[curr];
}

@Override
public KEY_GENERIC_TYPE PREV_KEY() {
 if (! hasPrevious()) throw new NoSuchElementException();

 curr = prev;
 prev = GET_PREV(link[curr]);
 next = curr;

 if (index >= 0) index--;

 return key[curr];
}

private final void ensureIndexKnown() {
 if (index >= 0) return;
 if (prev == -1) {
  index = 0;
  return;
 }
 if (next == -1) {
  index = size;
  return;
 }
 int pos = first;
 index = 1;
 while(pos != prev) {
  pos = GET_NEXT(link[pos]);
  index++;
 }
}

@Override
public int nextIndex() {
 ensureIndexKnown();
 return index;
}

@Override
public int previousIndex() {
 ensureIndexKnown();
 return index - 1;
}

@Override
```

```
public void remove() {
 ensureIndexKnown();
 if (curr == -1) throw new IllegalStateException();

 if (curr == prev) {
  /* If the last operation was a next(), we are removing an entry that preceeds
     the current index, and thus we must decrement it. */
  index--;
  prev = GET_PREV(link[curr]);
 }
 else
  next = GET_NEXT(link[curr]);

 size--;
 /* Now we manually fix the pointers. Because of our knowledge of next
    and prev, this is going to be faster than calling fixPointers(). */
 if (prev == -1) first = next;
 else
  SET_NEXT(link[prev], next);
 if (next == -1) last = prev;
 else
  SET_PREV(link[next], prev);

 int last, slot, pos = curr;
 curr = -1;

 if (pos == n) {
  OPEN_HASH_SET.this.containsNull = false;
  OPEN_HASH_SET.this.key[n] = KEY_NULL;
 }
 else {
  KEY_GENERIC_TYPE curr;
  final KEY_GENERIC_TYPE[] key = OPEN_HASH_SET.this.key;
  // We have to horribly duplicate the shiftKeys() code because we need to update next/prev.
  for(;;) {
   pos = ((last = pos) + 1) & mask;
   for(;;) {
    if (KEY_IS_NULL(curr = key[pos])) {
     key[last] = KEY_NULL;
     return;
    }
    slot = KEY2INTHASH(curr) & mask;
    if (last <= pos ? last >= slot || slot > pos : last >= slot && slot > pos) break;
    pos = (pos + 1) & mask;
   }

   key[last] = curr;
   if (next == pos) next = last;
```

```
    if (prev == pos) prev = last;
    fixPointers(pos, last);
   }
  }
 }
}


/** Returns a type-specific list iterator on the elements in this set, starting from a given element of the set.
 * Please see the class documentation for implementation details.
 *
 * @param from an element to start from.
 * @return a type-specific list iterator starting at the given element.
 * @throws IllegalArgumentException if {@code from} does not belong to the set.
 */
@Override
public KEY_LIST_ITERATOR KEY_GENERIC iterator(KEY_GENERIC_TYPE from) {
 return new SetIterator(from);
}

/** Returns a type-specific list iterator on the elements in this set, starting from the first element.
 * Please see the class documentation for implementation details.
 *
 * @return a type-specific list iterator starting at the first element.
 */
@Override
public KEY_LIST_ITERATOR KEY_GENERIC iterator() {
 return new SetIterator();
}

#else

/** An iterator over a hash set. */

private class SetIterator implements KEY_ITERATOR KEY_GENERIC {
 /** The index of the last entry returned, if positive or zero; initially, {@link #n}. If negative, the last
  element returned was that of index {@code - pos - 1} from the {@link #wrapped} list. */
 int pos = n;
 /** The index of the last entry that has been returned (more precisely, the value of {@link #pos} if {@link #pos} is
positive,
  or {@link Integer#MIN_VALUE} if {@link #pos} is negative). It is -1 if either
  we did not return an entry yet, or the last returned entry has been removed. */
 int last = -1;
 /** A downward counter measuring how many entries must still be returned. */
 int c = size;
 /** A boolean telling us whether we should return the null key. */
 boolean mustReturnNull = OPEN_HASH_SET.this.containsNull;
 /** A lazily allocated list containing elements that have wrapped around the table because of removals. */
```

```
ARRAY_LIST KEY_GENERIC wrapped;

@Override
public boolean hasNext() {
 return c != 0;
}

@Override
public KEY_GENERIC_TYPE NEXT_KEY() {
 if (! hasNext()) throw new NoSuchElementException();
 c--;
 if (mustReturnNull) {
  mustReturnNull = false;
  last = n;
  return key[n];
 }
 final KEY_GENERIC_TYPE key[] = OPEN_HASH_SET.this.key;
 for(;;) {
  if (--pos < 0) {
   // We are just enumerating elements from the wrapped list.
   last = Integer.MIN_VALUE;
   return wrapped.GET_KEY(- pos - 1);
  }
  if (! KEY_IS_NULL(key[pos])) return key[last = pos];
 }
}


/** Shifts left entries with the specified hash code, starting at the specified position,
 * and empties the resulting free entry.
 *
 * @param pos a starting position.
 */
private final void shiftKeys(int pos) {
 // Shift entries with the same hash.
 int last, slot;
 KEY_GENERIC_TYPE curr;
 final KEY_GENERIC_TYPE[] key = OPEN_HASH_SET.this.key;

 for(;;) {
  pos = ((last = pos) + 1) & mask;

  for(;;) {
   if (KEY_IS_NULL(curr = key[pos])) {
    key[last] = KEY_NULL;
    return;
   }
   slot = KEY2INTHASH(curr) & mask;
```

```
    if (last <= pos ? last >= slot || slot > pos : last >= slot && slot > pos) break;
    pos = (pos + 1) & mask;
   }

   if (pos < last) { // Wrapped entry.
    if (wrapped == null) wrapped = new ARRAY_LIST KEY_GENERIC_DIAMOND(2);
    wrapped.add(key[pos]);
   }

   key[last] = curr;
  }
 }

 @Override
 public void remove() {
  if (last == -1) throw new IllegalStateException();
  if (last == n) {
   OPEN_HASH_SET.this.containsNull = false;
   OPEN_HASH_SET.this.key[n] = KEY_NULL;
  }
  else if (pos >= 0) shiftKeys(last);
  else {
   // We're removing wrapped entries.
#if KEYS_REFERENCE
   OPEN_HASH_SET.this.remove(wrapped.set(- pos - 1, null));
#else
   OPEN_HASH_SET.this.remove(wrapped.GET_KEY(- pos - 1));
#endif
   last = -1; // Note that we must not decrement size
   return;
  }

  size--;
  last = -1; // You can no longer remove this entry.
  if (ASSERTS) checkTable();
 }
}

@Override
public KEY_ITERATOR KEY_GENERIC iterator() {
 return new SetIterator();
}

#endif


/** Rehashes this set, making the table as small as possible.
 *
```

```
 * <p>This method rehashes the table to the smallest size satisfying the
 * load factor. It can be used when the set will not be changed anymore, so
 * to optimize access speed and size.
 *
 * <p>If the table size is already the minimum possible, this method
 * does nothing.
 *
 * @return true if there was enough memory to trim the set.
 * @see #trim(int)
 */

public boolean trim() {
 final int l = arraySize(size, f);
 if (l >= n || size > maxFill(l, f)) return true;
 try {
  rehash(l);
 }
 catch(OutOfMemoryError cantDoIt) { return false; }
 return true;
}


/** Rehashes this set if the table is too large.
 *
 * <p>Let <var>N</var> be the smallest table size that can hold
 * <code>max(n,{@link #size()})</code> entries, still satisfying the load factor. If the current
 * table size is smaller than or equal to <var>N</var>, this method does
 * nothing. Otherwise, it rehashes this set in a table of size
 * <var>N</var>.
 *
 * <p>This method is useful when reusing sets.  {@linkplain #clear() Clearing a
 * set} leaves the table size untouched. If you are reusing a set
 * many times, you can call this method with a typical
 * size to avoid keeping around a very large table just
 * because of a few large transient sets.
 *
 * @param n the threshold for the trimming.
 * @return true if there was enough memory to trim the set.
 * @see #trim()
 */

public boolean trim(final int n) {
 final int l = HashCommon.nextPowerOfTwo((int)Math.ceil(n / f));
 if (l >= n || size > maxFill(l, f)) return true;
 try {
  rehash(l);
 }
 catch(OutOfMemoryError cantDoIt) { return false; }
```

```
 return true;
}

/** Rehashes the set.
 *
 * <p>This method implements the basic rehashing strategy, and may be
 * overriden by subclasses implementing different rehashing strategies (e.g.,
 * disk-based rehashing). However, you should not override this method
 * unless you understand the internal workings of this class.
 *
 * @param newN the new size
 */

SUPPRESS_WARNINGS_KEY_UNCHECKED
protected void rehash(final int newN) {
 final KEY_GENERIC_TYPE key[] = this.key;

 final int mask = newN - 1; // Note that this is used by the hashing macro
 final KEY_GENERIC_TYPE newKey[] = KEY_GENERIC_ARRAY_CAST new KEY_TYPE[newN + 1];

#ifdef Linked
 int i = first, prev = -1, newPrev = -1, t, pos;
 final long link[] = this.link;
 final long newLink[] = new long[newN + 1];
 first = -1;

 for(int j = size; j-- != 0;) {
  if (KEY_EQUALS_NULL(key[i])) pos = newN;
  else {
   pos = KEY2INTHASH(key[i]) & mask;
   while (! KEY_IS_NULL(newKey[pos])) pos = (pos + 1) & mask;
  }

  newKey[pos] = key[i];

  if (prev != -1) {
   SET_NEXT(newLink[newPrev], pos);
   SET_PREV(newLink[pos], newPrev);
   newPrev = pos;
  }
  else {
   newPrev = first = pos;
   // Special case of SET(newLink[pos], -1, -1);
   newLink[pos] = -1L;
  }

  t = i;
  i = GET_NEXT(link[i]);
```

```
  prev = t;
  }

  this.link = newLink;
  this.last = newPrev;
  if (newPrev != -1)
   // Special case of SET_NEXT(newLink[newPrev], -1);
   newLink[newPrev] |= -1 & 0xFFFFFFFFL;
#else
  int i = n, pos;

  for(int j = realSize(); j-- != 0;) {
   while(KEY_IS_NULL(key[--i]));
   if (! KEY_IS_NULL(newKey[pos = KEY2INTHASH(key[i]) & mask]))
    while (! KEY_IS_NULL(newKey[pos = (pos + 1) & mask]));
   newKey[pos] = key[i];
  }
#endif

  n = newN;
  this.mask = mask;
  maxFill = maxFill(n, f);
  this.key = newKey;
 }


 /** Returns a deep copy of this set.
  *
  * <p>This method performs a deep copy of this hash set; the data stored in the
  * set, however, is not cloned. Note that this makes a difference only for object keys.
  *
  *  @return a deep copy of this set.
  */
 @Override
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public OPEN_HASH_SET KEY_GENERIC clone() {
  OPEN_HASH_SET KEY_GENERIC c;
  try {
   c = (OPEN_HASH_SET KEY_GENERIC)super.clone();
  }
  catch(CloneNotSupportedException cantHappen) {
   throw new InternalError();
  }
  c.key = key.clone();
  c.containsNull = containsNull;
#ifdef Linked
  c.link = link.clone();
#endif
```

```
#ifdef Custom
 c.strategy = strategy;
#endif
 return c;
 }

 /** Returns a hash code for this set.
  *
  * This method overrides the generic method provided by the superclass.
  * Since {@code equals()} is not overriden, it is important
  * that the value returned by this method is the same value as
  * the one returned by the overriden method.
  *
  * @return a hash code for this set.
  */
 @Override
 public int hashCode() {
  int h = 0;
  for(int j = realSize(), i = 0; j-- != 0;) {
   while(KEY_IS_NULL(key[i])) i++;
#if KEYS_REFERENCE
   if (this != key[i])
#endif
    h += KEY2JAVAHASH_NOT_NULL(key[i]);
   i++;
  }

  // Zero / null have hash zero.
  return h;
 }


 private void writeObject(java.io.ObjectOutputStream s) throws java.io.IOException {
  final KEY_ITERATOR KEY_GENERIC i = iterator();
  s.defaultWriteObject();
  for(int j = size; j-- != 0;) s.WRITE_KEY(i.NEXT_KEY());
 }


 SUPPRESS_WARNINGS_KEY_UNCHECKED
 private void readObject(java.io.ObjectInputStream s) throws java.io.IOException, ClassNotFoundException {
  s.defaultReadObject();

  n = arraySize(size, f);
  maxFill = maxFill(n, f);
  mask = n - 1;

  final KEY_GENERIC_TYPE key[] = this.key = KEY_GENERIC_ARRAY_CAST new KEY_TYPE[n + 1];
```

```
#ifdef Linked
  final long link[] = this.link = new long[n + 1];
  int prev = -1;
  first = last = -1;
#endif

  KEY_GENERIC_TYPE k;

  for(int i = size, pos; i-- != 0;) {

   k = KEY_GENERIC_CAST s.READ_KEY();

   if (KEY_EQUALS_NULL(k)) {
    pos = n;
    containsNull = true;
   }
   else {
    if (! KEY_IS_NULL(key[pos = KEY2INTHASH(k) & mask]))
     while (! KEY_IS_NULL(key[pos = (pos + 1) & mask]));
   }

   key[pos] = k;

#ifdef Linked
   if (first != -1) {
    SET_NEXT(link[prev], pos);
    SET_PREV(link[pos], prev);
    prev = pos;
   }
   else {
    prev = first = pos;
    // Special case of SET_PREV(newLink[pos], -1);
    link[pos] |= (-1L & 0xFFFFFFFFL) << 32;
   }
#endif
  }

#ifdef Linked
  last = prev;
  if (prev != -1)
   // Special case of SET_NEXT(link[prev], -1);
   link[prev] |= -1 & 0xFFFFFFFFL;
#endif

  if (ASSERTS) checkTable();
 }
```

```
#ifdef ASSERTS_CODE
 private void checkTable() {
  assert (n & -n) == n : "Table length is not a power of two: " + n;
  assert n == key.length - 1;
  int n = key.length - 1;
  while(n-- != 0)
   if (! KEY_IS_NULL(key[n]) && ! contains(key[n]))
    throw new AssertionError("Hash table has key " + key[n] + " marked as occupied, but the key does not belong to the table");

#if KEYS_PRIMITIVE
  java.util.HashSet<KEY_GENERIC_CLASS> s = new java.util.HashSet<KEY_GENERIC_CLASS> ();
#else
  java.util.HashSet<Object> s = new java.util.HashSet<Object>();
#endif

  for(int i = key.length - 1; i-- != 0;)
   if (! KEY_IS_NULL(key[i]) && ! s.add(key[i])) throw new AssertionError("Key " + key[i] + " appears twice at position " + i);

#ifdef Linked
  KEY_LIST_ITERATOR KEY_GENERIC i = iterator();
  KEY_GENERIC_TYPE k;
  n = size();
  while(n-- != 0)
   if (! contains(k = i.NEXT_KEY()))
    throw new AssertionError("Linked hash table forward enumerates key " + k + ", but the key does not belong to the table");

  if (i.hasNext()) throw new AssertionError("Forward iterator not exhausted");

  n = size();
  if (n > 0) {
   i = iterator(LAST());
   while(n-- != 0)
    if (! contains(k = i.PREV_KEY()))
     throw new AssertionError("Linked hash table backward enumerates key " + k + ", but the key does not belong to the table");

   if (i.hasPrevious()) throw new AssertionError("Previous iterator not exhausted");
  }
#endif
 }
#else
 private void checkTable() {}
#endif

#ifdef TEST
```

```
 private static long seed = System.currentTimeMillis();
 private static java.util.Random r = new java.util.Random(seed);

 private static KEY_TYPE genKey() {
#if KEY_CLASS_Byte || KEY_CLASS_Short || KEY_CLASS_Character
  return (KEY_TYPE)(r.nextInt());
#elif KEYS_PRIMITIVE
  return r.NEXT_KEY();
#elif KEY_CLASS_Object
#ifdef Custom
  int i = r.nextInt(3);
  byte a[] = new byte[i];
  while(i-- != 0) a[i] = (byte)r.nextInt();
  return a;
#else
  return Integer.toBinaryString(r.nextInt());
#endif
#else
  return new java.io.Serializable() {};
#endif
 }


 private static final class ArrayComparator implements java.util.Comparator {
  public int compare(Object a, Object b) {
   byte[] aa = (byte[])a;
   byte[] bb = (byte[])b;
   int length = Math.min(aa.length, bb.length);
   for(int i = 0; i < length; i++) {
    if (aa[i] < bb[i]) return -1;
    if (aa[i] > bb[i]) return 1;
   }
   return aa.length == bb.length ? 0 : (aa.length < bb.length ? -1 : 1);
  }
 }

 private static final class MockSet extends java.util.TreeSet {
  private java.util.List list = new java.util.ArrayList();

  public MockSet(java.util.Comparator c) { super(c); }

  public boolean add(Object k) {
   if (! contains(k)) list.add(k);
   return super.add(k);
  }

  public boolean addAll(Collection c) {
```

```
 java.util.Iterator i = c.iterator();
 boolean result = false;
 while(i.hasNext()) result |= add(i.next());
 return result;
 }

 public boolean removeAll(Collection c) {
  java.util.Iterator i = c.iterator();
  boolean result = false;
  while(i.hasNext()) result |= remove(i.next());
  return result;
 }

 public boolean remove(Object k) {
  if (contains(k)) {
   int i = list.size();
   while(i-- != 0) if (comparator().compare(list.get(i), k) == 0) {
    list.remove(i);
    break;
   }
  }
  return super.remove(k);
 }

 private void justRemove(Object k) { super.remove(k); }

 public java.util.Iterator iterator() {
  return new java.util.Iterator() {
    final java.util.Iterator iterator = list.iterator();
    Object curr;
    public Object next() { return curr = iterator.next(); }
    public boolean hasNext() { return iterator.hasNext(); }
    public void remove() {
     justRemove(curr);
     iterator.remove();
    }
   };
  }
 }

 private static java.text.NumberFormat format = new java.text.DecimalFormat("#,###.00");
 private static java.text.FieldPosition fp = new java.text.FieldPosition(0);

 private static String format(double d) {
  StringBuffer s = new StringBuffer();
  return format.format(d, s, fp).toString();
 }
```

```
 private static void speedTest(int n, float f, boolean comp) {
#ifndef Custom
  int i, j;
  OPEN_HASH_SET m;
#ifdef Linked
  java.util.LinkedHashSet t;
#else
  java.util.HashSet t;
#endif

  KEY_TYPE k[] = new KEY_TYPE[n];
  KEY_TYPE nk[] = new KEY_TYPE[n];
  long ns;

  for(i = 0; i < n; i++) {
   k[i] = genKey();
   nk[i] = genKey();
  }

  double totAdd = 0, totYes = 0, totNo = 0, totIter = 0, totRemYes = 0, totRemNo = 0, d;

  if (comp) { for(j = 0; j < 20; j++) {

#ifdef Linked
   t = new java.util.LinkedHashSet(16);
#else
   t = new java.util.HashSet(16);
#endif

   /* We add pairs to t. */
   ns = System.nanoTime();
   for(i = 0; i < n;  i++) t.add(KEY2OBJ(k[i]));
   d = (System.nanoTime() - ns) / (double)n;
   if (j > 2) totAdd += d;
   System.out.print("Add: " + format(d) + "ns ");

   /* We check for pairs in t. */
   ns = System.nanoTime();
   for(i = 0; i < n;  i++) t.contains(KEY2OBJ(k[i]));
   d = (System.nanoTime() - ns) / (double)n;
   if (j > 2) totYes += d;
   System.out.print("Yes: " + format(d) + "ns ");

   /* We check for pairs not in t. */
   ns = System.nanoTime();
   for(i = 0; i < n;  i++) t.contains(KEY2OBJ(nk[i]));
   d = (System.nanoTime() - ns) / (double)n;
   if (j > 2) totNo += d;
```

```
  System.out.print("No: " + format(d) + "ns ");

  /* We iterate on t. */
  ns = System.nanoTime();
  for(java.util.Iterator it = t.iterator(); it.hasNext(); it.next());
  d = (System.nanoTime() - ns) / (double)n;
  if (j > 2) totIter += d;
  System.out.print("Iter: " + format(d) + "ns ");

  // Too expensive in the linked case
#ifndef Linked
  /* We delete pairs not in t. */
  ns = System.nanoTime();
  for(i = 0; i < n;  i++) t.remove(KEY2OBJ(nk[i]));
  d = (System.nanoTime() - ns) / (double)n;
  if (j > 2) totRemNo += d;
  System.out.print("RemNo: " + format(d) + "ns ");

  /* We delete pairs in t. */
  ns = System.nanoTime();
  for(i = 0; i < n;  i++) t.remove(KEY2OBJ(k[i]));
  d = (System.nanoTime() - ns) / (double)n;
  if (j > 2) totRemYes += d;
  System.out.print("RemYes: " + format(d) + "ns ");
#endif

  System.out.println();
  }

  System.out.println();
  System.out.println("java.util Add: " + format(totAdd/(j-3)) + "ns Yes: " + format(totYes/(j-3)) + "ns No: " +
format(totNo/(j-3)) + "ns Iter: " + format(totIter/(j-3)) + "ns RemNo: " + format(totRemNo/(j-3)) + "ns RemYes: " +
format(totRemYes/(j-3)) + "ns");

  System.out.println();

  totAdd = totYes = totNo = totIter = totRemYes = totRemNo = 0;
  }

  for(j = 0; j < 20; j++) {

  m = new OPEN_HASH_SET(16, f);

  /* We add pairs to m. */
  ns = System.nanoTime();
  for(i = 0; i < n;  i++) m.add(k[i]);
  d = (System.nanoTime() - ns) / (double)n;
  if (j > 2) totAdd += d;
```

```
  System.out.print("Add: " + format(d) + "ns ");

  /* We check for pairs in m. */
  ns = System.nanoTime();
  for(i = 0; i < n;  i++) m.contains(k[i]);
  d = (System.nanoTime() - ns) / (double)n;
  if (j > 2) totYes += d;
  System.out.print("Yes: " + format(d) + "ns ");

  /* We check for pairs not in m. */
  ns = System.nanoTime();
  for(i = 0; i < n;  i++) m.contains(nk[i]);
  d = (System.nanoTime() - ns) / (double)n;
  if (j > 2) totNo += d;
  System.out.print("No: " + format(d) + "ns ");

  /* We iterate on m. */
  ns = System.nanoTime();
  for(KEY_ITERATOR it = (KEY_ITERATOR)m.iterator(); it.hasNext(); it.NEXT_KEY());
  d = (System.nanoTime() - ns) / (double)n;
  if (j > 2) totIter += d;
  System.out.print("Iter: " + format(d) + "ns ");

  // Too expensive in the linked case
#ifndef Linked
  /* We delete pairs not in m. */
  ns = System.nanoTime();
  for(i = 0; i < n;  i++) m.remove(nk[i]);
  d = (System.nanoTime() - ns) / (double)n;
  if (j > 2) totRemNo += d;
  System.out.print("RemNo: " + format(d) + "ns ");

  /* We delete pairs in m. */
  ns = System.nanoTime();
  for(i = 0; i < n;  i++) m.remove(k[i]);
  d = (System.nanoTime() - ns) / (double)n;
  if (j > 2) totRemYes += d;
  System.out.print("RemYes: " + format(d) + "ns ");
#endif

  System.out.println();
  }


  System.out.println();
  System.out.println("fastutil  Add: " + format(totAdd/(j-3)) + "ns Yes: " + format(totYes/(j-3)) + "ns No: " +
format(totNo/(j-3)) + "ns Iter: " + format(totIter/(j-3)) + "ns RemNo: " + format(totRemNo/(j-3)) + "ns RemYes: " +
format(totRemYes/(j-3)) + "ns");
```

```
 System.out.println();
#endif
 }


 private static void fatal(String msg) {
  System.out.println(msg);
  System.exit(1);
 }

 private static void ensure(boolean cond, String msg) {
  if (cond) return;
  fatal(msg);
 }


 private static void printProbes(OPEN_HASH_SET m) {
  long totProbes = 0;
  double totSquareProbes = 0;
  int maxProbes = 0;
  final double f = (double)m.size / m.n;
  for(int i = 0, c = 0; i < m.n; i++) {
   if (! KEY_IS_NULL(m.key[i])) c++;
   else {
    if (c != 0) {
     final long p = (c + 1) * (c + 2) / 2;
     totProbes += p;
     totSquareProbes += (double)p * p;
    }
    maxProbes = Math.max(c, maxProbes);
    c = 0;
    totProbes++;
    totSquareProbes++;
   }
  }

  final double expected = (double)totProbes / m.n;
  System.err.println("Expected probes: " + (
   3 * Math.sqrt(3) * (f / ((1 - f) * (1 - f))) + 4 / (9 * f) - 1
  ) + "; actual: " + expected + "; stddev: " + Math.sqrt(totSquareProbes / m.n - expected * expected)  + "; max probes:
" + maxProbes);
 }


 private static void runTest(int n, float f) {
#if !defined(Custom) || KEYS_REFERENCE
```

```
  int c;
#ifdef Custom
  OPEN_HASH_SET m = new OPEN_HASH_SET(Hash.DEFAULT_INITIAL_SIZE, f,
it.unimi.dsi.fastutil.bytes.ByteArrays.HASH_STRATEGY);
#else
  OPEN_HASH_SET m = new OPEN_HASH_SET(Hash.DEFAULT_INITIAL_SIZE, f);
#endif
#ifdef Linked
#ifdef Custom
  java.util.Set t = new MockSet(new ArrayComparator());
#else
  java.util.Set t = new java.util.LinkedHashSet();
#endif
#else
#ifdef Custom
  java.util.Set t = new java.util.TreeSet(new ArrayComparator());
#else
  java.util.Set t = new java.util.HashSet();
#endif
#endif

  /* First of all, we fill t with random data. */

  for(int i=0; i<f * n;  i++) t.add(KEY2OBJ(genKey()));

  /* Now we add to m the same data */

  m.addAll(t);

  if (!m.equals(t)) System.out.println("Error (" + seed + "): !m.equals(t) after insertion");
  if (!t.equals(m)) System.out.println("Error (" + seed + "): !t.equals(m) after insertion");
  printProbes(m);

  /* Now we check that m actually holds that data. */

  for(java.util.Iterator i=t.iterator(); i.hasNext();) {
   Object e = i.next();
   if (!m.contains(e)) {
    System.out.println("Error (" + seed + "): m and t differ on a key ("+e+") after insertion (iterating on t)");
    System.exit(1);
   }
  }

  /* Now we check that m actually holds that data, but iterating on m. */

  c = 0;
  for(java.util.Iterator i=m.iterator(); i.hasNext();) {
   Object e = i.next();
```

```
 c++;
 if (!t.contains(e)) {
  System.out.println("Error (" + seed + "): m and t differ on a key ("+e+") after insertion (iterating on m)");
  System.exit(1);
 }
}

if (c != t.size()) {
 System.out.println("Error (" + seed + "): m has only " + c + " keys instead of " + t.size() + " after insertion (iterating
on m)");
 System.exit(1);
}
/* Now we check that inquiries about random data give the same answer in m and t. For
   m we use the polymorphic method. */

for(int i=0; i<n;  i++) {
 KEY_TYPE T = genKey();
 if (m.contains(T) != t.contains(KEY2OBJ(T))) {
  System.out.println("Error (" + seed + "): divergence in keys between t and m (polymorphic method)");
  System.exit(1);
 }
}

/* Again, we check that inquiries about random data give the same answer in m and t, but
   for m we use the standard method. */

for(int i=0; i<n;  i++) {
 KEY_TYPE T = genKey();
 if (m.contains(KEY2OBJ(T)) != t.contains(KEY2OBJ(T))) {
  System.out.println("Error (" + seed + "): divergence between t and m (standard method)");
  System.exit(1);
 }
}


/* Now we put and remove random data in m and t, checking that the result is the same. */

for(int i=0; i<20*n;  i++) {
 KEY_TYPE T = genKey();
 if (m.add(KEY2OBJ(T)) != t.add(KEY2OBJ(T))) {
  System.out.println("Error (" + seed + "): divergence in add() between t and m");
  System.exit(1);
 }
 T = genKey();
 if (m.remove(KEY2OBJ(T)) != t.remove(KEY2OBJ(T))) {
  System.out.println("Error (" + seed + "): divergence in remove() between t and m");
  System.exit(1);
 }
```

```
  }

  if (!m.equals(t)) System.out.println("Error (" + seed + "): !m.equals(t) after removal");
  if (!t.equals(m)) System.out.println("Error (" + seed + "): !t.equals(m) after removal");

  /* Now we check that m actually holds that data. */

  for(java.util.Iterator i=t.iterator(); i.hasNext();) {
   Object e = i.next();
   if (!m.contains(e)) {
    System.out.println("Error (" + seed + "): m and t differ on a key ("+e+") after removal (iterating on t)");
    System.exit(1);
   }
  }

  /* Now we check that m actually holds that data, but iterating on m. */

  for(java.util.Iterator i=m.iterator(); i.hasNext();) {
   Object e = i.next();
   if (!t.contains(e)) {
    System.out.println("Error (" + seed + "): m and t differ on a key ("+e+") after removal (iterating on m)");
    System.exit(1);
   }
  }

  printProbes(m);

  /* Now we make m into an array, make it again a set and check it is OK. */
  KEY_TYPE a[] = m.TO_KEY_ARRAY();

#ifdef Custom
  if (!new OPEN_HASH_SET(a, m.strategy()).equals(m))
   System.out.println("Error (" + seed + "): toArray() output (or array-based constructor) is not OK");
#else
  if (!new OPEN_HASH_SET(a).equals(m))
   System.out.println("Error (" + seed + "): toArray() output (or array-based constructor) is not OK");
#endif

  /* Now we check cloning. */

  ensure(m.equals(((OPEN_HASH_SET)m).clone()), "Error (" + seed + "): m does not equal m.clone()");
  ensure(((OPEN_HASH_SET)m).clone().equals(m), "Error (" + seed + "): m.clone() does not equal m");

  int h = m.hashCode();

  /* Now we save and read m. */

  try {
```

```
  java.io.File ff = new java.io.File("it.unimi.dsi.fastutil.test");
  java.io.OutputStream os = new java.io.FileOutputStream(ff);
  java.io.ObjectOutputStream oos = new java.io.ObjectOutputStream(os);

  oos.writeObject(m);
  oos.close();

  java.io.InputStream is = new java.io.FileInputStream(ff);
  java.io.ObjectInputStream ois = new java.io.ObjectInputStream(is);

  m = (OPEN_HASH_SET)ois.readObject();
  ois.close();
  ff.delete();
 }
 catch(Exception e) {
  e.printStackTrace();
  System.exit(1);
 }

#if !KEY_CLASS_Reference
 if (m.hashCode() != h) System.out.println("Error (" + seed + "): hashCode() changed after save/read");

 printProbes(m);

 /* Now we check that m actually holds that data, but iterating on m. */

 for(java.util.Iterator i=m.iterator(); i.hasNext();) {
  Object e = i.next();
  if (!t.contains(e)) {
   System.out.println("Error (" + seed + "): m and t differ on a key ("+e+") after save/read");
   System.exit(1);
  }
 }
#else
 m.clear();
 m.addAll(t);
#endif

 /* Now we put and remove random data in m and t, checking that the result is the same. */

 for(int i=0; i<20*n;  i++) {
  KEY_TYPE T = genKey();
  if (m.add(KEY2OBJ(T)) != t.add(KEY2OBJ(T))) {
   System.out.println("Error (" + seed + "): divergence in add() between t and m after save/read");
   System.exit(1);
  }
  T = genKey();
  if (m.remove(KEY2OBJ(T)) != t.remove(KEY2OBJ(T))) {
```

```
   System.out.println("Error (" + seed + "): divergence in remove() between t and m after save/read");
   System.exit(1);
  }
 }

 if (!m.equals(t)) System.out.println("Error (" + seed + "): !m.equals(t) after post-save/read removal");
 if (!t.equals(m)) System.out.println("Error (" + seed + "): !t.equals(m) after post-save/read removal");


#ifdef Linked


 /* Now we play with iterators, but only in the linked case. */

 {
  java.util.ListIterator i, j;
  Object I, J;
  i = (java.util.ListIterator)m.iterator();
  j = new java.util.LinkedList(t).listIterator();

  for(int k = 0; k < 2*n; k++) {
   ensure(i.hasNext() == j.hasNext(), "Error (" + seed + "): divergence in hasNext()");
   ensure(i.hasPrevious() == j.hasPrevious(), "Error (" + seed + "): divergence in hasPrevious()");

   if (r.nextFloat() < .8 && i.hasNext()) {
#ifdef Custom
    ensure(m.strategy().equals(i.next(), J = j.next()), "Error (" + seed + "): divergence in next()");
#else
    ensure(i.next().equals(J = j.next()), "Error (" + seed + "): divergence in next()");
#endif
    if (r.nextFloat() < 0.5) {
     i.remove();
     j.remove();
     t.remove(J);
    }
   }
   else if (r.nextFloat() < .2 && i.hasPrevious()) {
#ifdef Custom
    ensure(m.strategy().equals(i.previous(), J = j.previous()), "Error (" + seed + "): divergence in previous()");
#else
    ensure(i.previous().equals(J = j.previous()), "Error (" + seed + "): divergence in previous()");
#endif
    if (r.nextFloat() < 0.5) {
     i.remove();
     j.remove();
     t.remove(J);
    }
   }
```

```
    ensure(i.nextIndex() == j.nextIndex(), "Error (" + seed + "): divergence in nextIndex()");
    ensure(i.previousIndex() == j.previousIndex(), "Error (" + seed + "): divergence in previousIndex()");

   }

  }

  if (t.size() > 0) {
   java.util.ListIterator i, j;
   Object J;
   j = new java.util.LinkedList(t).listIterator();
   int e = r.nextInt(t.size());
   Object from;
   do from = j.next(); while(e-- != 0);

   i = (java.util.ListIterator)m.iterator(KEY_OBJ2TYPE(from));

   for(int k = 0; k < 2*n; k++) {
    ensure(i.hasNext() == j.hasNext(), "Error (" + seed + "): divergence in hasNext() (iterator with starting point " +
from + ")");
    ensure(i.hasPrevious() == j.hasPrevious(), "Error (" + seed + "): divergence in hasPrevious() (iterator with starting
point " + from + ")");

    if (r.nextFloat() < .8 && i.hasNext()) {
#ifdef Custom
     ensure(m.strategy().equals(i.next(), J = j.next()), "Error (" + seed + "): divergence in next() (iterator with starting
point " + from + ")");
#else
     ensure(i.next().equals(J = j.next()), "Error (" + seed + "): divergence in next() (iterator with starting point " + from +
")");
#endif

     if (r.nextFloat() < 0.5) {
      i.remove();
      j.remove();
      t.remove(J);
     }
    }
    else if (r.nextFloat() < .2 && i.hasPrevious()) {
#ifdef Custom
     ensure(m.strategy().equals(i.previous(), J = j.previous()), "Error (" + seed + "): divergence in previous() (iterator
with starting point " + from + ")");
#else
     ensure(i.previous().equals(J = j.previous()), "Error (" + seed + "): divergence in previous() (iterator with starting
point " + from + ")");
#endif
```

```
     if (r.nextFloat() < 0.5) {
      i.remove();
      j.remove();
      t.remove(J);
     }
    }

    ensure(i.nextIndex() == j.nextIndex(), "Error (" + seed + "): divergence in nextIndex() (iterator with starting point "
+ from + ")");
    ensure(i.previousIndex() == j.previousIndex(), "Error (" + seed + "): divergence in previousIndex() (iterator with
starting point " + from + ")");

   }

  }

  /* Now we check that m actually holds that data. */

  ensure(m.equals(t), "Error (" + seed + "): ! m.equals(t) after iteration");
  ensure(t.equals(m), "Error (" + seed + "): ! t.equals(m) after iteration");



#endif

  /* Now we take out of m everything, and check that it is empty. */

  for(java.util.Iterator i=m.iterator(); i.hasNext();) { i.next(); i.remove();}

  if (!m.isEmpty())  {
   System.out.println("Error (" + seed + "): m is not empty (as it should be)");
   System.exit(1);
  }

#if KEY_CLASS_Integer || KEY_CLASS_Long
  m = new OPEN_HASH_SET(n, f);
  t.clear();
  int x;

  /* Now we torture-test the hash table. This part is implemented only for integers and longs. */

  int p = m.key.length - 1;

  for(int i=0; i<p; i++) {
   for (int j=0; j<20; j++) {
    m.add(i+(r.nextInt() % 10)*p);
    m.remove(i+(r.nextInt() % 10)*p);
   }
```

```
  for (int j=-10; j<10; j++) m.remove(i+j*p);
  }

  t.addAll(m);

  /* Now all table entries are REMOVED. */

  int k = 0;
  for(int i=0; i<(p*f)/10; i++) {
   for (int j=0; j<10; j++) {
    k++;
    x = i+(r.nextInt() % 10)*p;
    if (m.add(x) != t.add(KEY2OBJ(x)))
     System.out.println("Error (" + seed + "): m and t differ on a key during torture-test insertion.");
   }
  }

  if (!m.equals(t)) System.out.println("Error (" + seed + "): !m.equals(t) after torture-test insertion");
  if (!t.equals(m)) System.out.println("Error (" + seed + "): !t.equals(m) after torture-test insertion");

  for(int i=0; i<(p*f)/10; i++) {
   for (int j=0; j<10; j++) {
    x = i+(r.nextInt() % 10)*p;
    if (m.remove(x) != t.remove(KEY2OBJ(x)))
     System.out.println("Error (" + seed + "): m and t differ on a key during torture-test removal.");
   }
  }

  if (!m.equals(t)) System.out.println("Error (" + seed + "): !m.equals(t) after torture-test removal");
  if (!t.equals(m)) System.out.println("Error (" + seed + "): !t.equals(m) after torture-test removal");

  if (!m.equals(m.clone())) System.out.println("Error (" + seed + "): !m.equals(m.clone()) after torture-test removal");
  if (!((OPEN_HASH_SET)m.clone()).equals(m)) System.out.println("Error (" + seed + "): !m.clone().equals(m) after
torture-test removal");

  m.trim();

  if (!m.equals(t)) System.out.println("Error (" + seed + "): !m.equals(t) after trim()");
  if (!t.equals(m)) System.out.println("Error (" + seed + "): !t.equals(m) after trim()");
#endif

  System.out.println("Test OK");
  return;
#endif
  }
```

```
 public static void main(String args[]) {
  float f = Hash.DEFAULT_LOAD_FACTOR;
  int n  = Integer.parseInt(args[1]);
  if (args.length>2) f = Float.parseFloat(args[2]);
  if (args.length > 3) r = new java.util.Random(seed = Long.parseLong(args[3]));

  try {
   if ("speedTest".equals(args[0]) || "speedComp".equals(args[0])) speedTest(n, f, "speedComp".equals(args[0]));
   else if ("test".equals(args[0])) runTest(n, f);
  } catch(Throwable e) {
   e.printStackTrace(System.err);
   System.err.println("seed: " + seed);
  }
 }

#endif

}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/OpenHashSet.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import java.util.Set;

/**  An abstract class providing basic methods for sets implementing a type-specific interface.
*
* <p>Note that the type-specific {@link Set} interface adds a type-specific {@code remove()}
```

```
* method, as it is no longer harmful for subclasses. Thus, concrete subclasses of this class
* must implement {@code remove()} (the {@code rem()} implementation of this
* class just delegates to {@code remove()}).
*/

public abstract class ABSTRACT_SET KEY_GENERIC extends ABSTRACT_COLLECTION KEY_GENERIC
implements Cloneable, SET KEY_GENERIC {

 protected ABSTRACT_SET() {}

 @Override
 public abstract KEY_ITERATOR KEY_GENERIC iterator();

 @Override
 public boolean equals(final Object o) {
  if (o == this) return true;
  if (!(o instanceof Set)) return false;

  Set<?> s = (Set<?>) o;
  if (s.size() != size()) return false;
  return containsAll(s);
 }


 /** Returns a hash code for this set.
  *
  * The hash code of a set is computed by summing the hash codes of
  * its elements.
  *
  * @return a hash code for this set.
  */
 @Override
 public int hashCode() {
  int h = 0, n = size();
  KEY_ITERATOR KEY_GENERIC i = iterator();
  KEY_GENERIC_TYPE k;

  while(n-- != 0) {
   k = i.NEXT_KEY(); // We need k because KEY2JAVAHASH() is a macro with repeated evaluation.
   h += KEY2JAVAHASH(k);
  }
  return h;
 }


#if KEYS_PRIMITIVE
 /** {@inheritDoc}
  * Delegates to the type-specific {@code rem()} method
```

```
 * implemented by type-specific abstract {@link java.util.Collection} superclass.
 */
@Override
public boolean remove(KEY_TYPE k) {
 return super.rem(k);
}

/** {@inheritDoc}
 * Delegates to the type-specific {@code remove()} method
 * specified in the type-specific {@link Set} interface.
 * @deprecated Please use {@code remove()} instead.
 */
@Deprecated
@Override
public boolean rem(KEY_TYPE k) {
 return remove(k);
}
#endif

}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/AbstractSet.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import java.util.Set;

/** A type-specific {@link Set}; provides some additional methods that use polymorphism to avoid (un)boxing.
```

```
*
* <p>Additionally, this interface strengthens (again) {@link #iterator()}.
*
* @see Set
*/

public interface SET KEY_GENERIC extends COLLECTION KEY_GENERIC, Set<KEY_GENERIC_CLASS> {

 /** Returns a type-specific iterator on the elements of this set.
  *
  * <p>Note that this specification strengthens the one given in {@link java.lang.Iterable#iterator()},
  * which was already strengthened in the corresponding type-specific class,
  * but was weakened by the fact that this interface extends {@link Set}.
  *
  * @return a type-specific iterator on the elements of this set.
  */
 @Override
 KEY_ITERATOR KEY_GENERIC iterator();

#if KEYS_PRIMITIVE
 /** Removes an element from this set.
  *
  * <p>Note that the corresponding method of a type-specific collection is {@code rem()}.
  * This unfortunate situation is caused by the clash
  * with the similarly named index-based method in the {@link java.util.List} interface.
  *
  * @see java.util.Collection#remove(Object)
  */
 boolean remove(KEY_TYPE k);

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead.
  */
 @SuppressWarnings("deprecation")
 @Deprecated
 @Override
 default boolean remove(final Object o) {
  return COLLECTION.super.remove(o);
 }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead.
  */
 @SuppressWarnings("deprecation")
 @Deprecated
 @Override
 default boolean add(final KEY_GENERIC_CLASS o) {
  return COLLECTION.super.add(o);
```

```
}

/** {@inheritDoc}
 * @deprecated Please use the corresponding type-specific method instead.
 */
@SuppressWarnings("deprecation")
@Deprecated
@Override
default boolean contains(final Object o) {
 return COLLECTION.super.contains(o);
}

/** Removes an element from this set.
 *
 * <p>This method is inherited from the type-specific collection this
 * type-specific set is based on, but it should not used as
 * this interface reinstates {@code remove()} as removal method.
 *
 * @deprecated Please use {@code remove()} instead.
 */
@Deprecated
@Override
default boolean rem(KEY_TYPE k) {
 return remove(k);
}

#endif
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/Set.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2003-2017 Paolo Boldi and Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
```

```
*/


package PACKAGE;

#if KEY_CLASS_Object
import java.util.Comparator;
#endif

/** A class providing static methods and objects that do useful things with heaps.
*
* <p>The static methods of this class allow to treat arrays as 0-based heaps. They
* are used in the implementation of heap-based queues, but they may be also used
* directly.
*
*/

public final class HEAPS {

 private HEAPS() {}

 /** Moves the given element down into the heap until it reaches the lowest possible position.
  *
  * @param heap the heap (starting at 0).
  * @param size the number of elements in the heap.
  * @param i the index of the element that must be moved down.
  * @param c a type-specific comparator, or {@code null} for the natural order.
  * @return the new position of the element of index {@code i}.
  */

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public static KEY_GENERIC int downHeap(final KEY_GENERIC_TYPE[] heap, final int size, int i, final
KEY_COMPARATOR KEY_SUPER_GENERIC c) {
  assert i < size;

  final KEY_GENERIC_TYPE e = heap[i];
  int child;

  if (c == null)
   while ((child = (i << 1) + 1) < size) {
    KEY_GENERIC_TYPE t = heap[child];
    final int right = child + 1;
    if (right < size && KEY_LESS(heap[right], t)) t = heap[child = right];
    if (KEY_LESSEQ(e, t)) break;
    heap[i] = t;
    i = child;
   }
  else
```

```
 while ((child = (i << 1) + 1) < size) {
  KEY_GENERIC_TYPE t = heap[child];
  final int right = child + 1;
  if (right < size && c.compare(heap[right], t) < 0) t = heap[child = right];
  if (c.compare(e, t) <= 0) break;
  heap[i] = t;
  i = child;
 }

 heap[i] = e;

 return i;
}

/** Moves the given element up in the heap until it reaches the highest possible position.
 *
 * @param heap the heap (starting at 0).
 * @param size the number of elements in the heap.
 * @param i the index of the element that must be moved up.
 * @param c a type-specific comparator, or {@code null} for the natural order.
 * @return the new position of the element of index {@code i}.
 */

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public static KEY_GENERIC int upHeap(final KEY_GENERIC_TYPE[] heap, final int size, int i, final
KEY_COMPARATOR KEY_GENERIC c) {
 assert i < size;

 final KEY_GENERIC_TYPE e = heap[i];

 if (c == null)
  while (i != 0) {
   final int parent = (i - 1) >>> 1;
   final KEY_GENERIC_TYPE t = heap[parent];
   if (KEY_LESSEQ(t, e)) break;
   heap[i] = t;
   i = parent;
  }
 else
  while (i != 0) {
   final int parent = (i - 1) >>> 1;
   final KEY_GENERIC_TYPE t = heap[parent];
   if (c.compare(t, e) <= 0) break;
   heap[i] = t;
   i = parent;
  }

 heap[i] = e;
```

```
 return i;
 }

 /** Makes an array into a heap.
  *
  * @param heap the heap (starting at 0).
  * @param size the number of elements in the heap.
  * @param c a type-specific comparator, or {@code null} for the natural order.
  */

 public static KEY_GENERIC void makeHeap(final KEY_GENERIC_TYPE[] heap, final int size, final
KEY_COMPARATOR KEY_GENERIC c) {
  int i = size >>> 1;
  while(i-- != 0) downHeap(heap, size, i, c);
 }
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/Heaps.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2003-2017 Paolo Boldi and Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

#if KEY_CLASS_Object
import java.util.Arrays;
import java.util.Comparator;
import it.unimi.dsi.fastutil.PriorityQueue;
#endif
```

```
import java.util.NoSuchElementException;

/** A type-specific array-based priority queue.
*
* <p>Instances of this class represent a priority queue using a backing
* array&amp;mdash;all operations are performed directly on the array. The array is
* enlarged as needed, but it is never shrunk. Use the {@link #trim()} method
* to reduce its size, if necessary.
*
* <p>This implementation is extremely inefficient, but it is difficult to beat
* when the size of the queue is very small.
*/

public class ARRAY_PRIORITY_QUEUE KEY_GENERIC implements PRIORITY_QUEUE KEY_GENERIC,
java.io.Serializable {
 private static final long serialVersionUID = 1L;

 /** The backing array. */
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 protected transient KEY_GENERIC_TYPE array[] = KEY_GENERIC_ARRAY_CAST
ARRAYS.EMPTY_ARRAY;

 /** The number of elements in this queue. */
 protected int size;

 /** The type-specific comparator used in this queue. */
 protected KEY_COMPARATOR KEY_SUPER_GENERIC c;

 /** The first index, cached, if {@link #firstIndexValid} is true. */
 protected transient int firstIndex;

 /** Whether {@link #firstIndex} contains a valid value. */
 protected transient boolean firstIndexValid;

 /** Creates a new empty queue with a given capacity and comparator.
  *
  * @param capacity the initial capacity of this queue.
  * @param c the comparator used in this queue, or {@code null} for the natural order.
  */
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public ARRAY_PRIORITY_QUEUE(int capacity, KEY_COMPARATOR KEY_SUPER_GENERIC c) {
  if (capacity > 0) this.array = KEY_GENERIC_ARRAY_CAST new KEY_TYPE[capacity];
  this.c = c;
 }

 /** Creates a new empty queue with a given capacity and using the natural order.
  *
  * @param capacity the initial capacity of this queue.
```

```
 */
public ARRAY_PRIORITY_QUEUE(int capacity) {
 this(capacity, null);
}

/** Creates a new empty queue with a given comparator.
 *
 * @param c the comparator used in this queue, or {@code null} for the natural order.
 */
public ARRAY_PRIORITY_QUEUE(KEY_COMPARATOR KEY_SUPER_GENERIC c) {
 this(0, c);
}

/** Creates a new empty queue using the natural order.
 */
public ARRAY_PRIORITY_QUEUE() {
 this(0, null);
}

/** Wraps a given array in a queue using a given comparator.
 *
 * <p>The queue returned by this method will be backed by the given array.
 *
 * @param a an array.
 * @param size the number of elements to be included in the queue.
 * @param c the comparator used in this queue, or {@code null} for the natural order.
 */
public ARRAY_PRIORITY_QUEUE(final KEY_GENERIC_TYPE[] a, int size, final KEY_COMPARATOR
KEY_SUPER_GENERIC c) {
 this(c);
 this.array = a;
 this.size = size;
}


/** Wraps a given array in a queue using a given comparator.
 *
 * <p>The queue returned by this method will be backed by the given array.
 *
 * @param a an array.
 * @param c the comparator used in this queue, or {@code null} for the natural order.
 */
public ARRAY_PRIORITY_QUEUE(final KEY_GENERIC_TYPE[] a, final KEY_COMPARATOR
KEY_SUPER_GENERIC c) {
 this(a, a.length, c);
}

/** Wraps a given array in a queue using the natural order.
```

```
 *
 * <p>The queue returned by this method will be backed by the given array.
 *
 * @param a an array.
 * @param size the number of elements to be included in the queue.
 */
public ARRAY_PRIORITY_QUEUE(final KEY_GENERIC_TYPE[] a, int size) {
 this(a, size, null);
}


/** Wraps a given array in a queue using the natural order.
 *
 * <p>The queue returned by this method will be backed by the given array.
 *
 * @param a an array.
 */
public ARRAY_PRIORITY_QUEUE(final KEY_GENERIC_TYPE[] a) {
 this(a, a.length);
}



/** Returns the index of the smallest element. */

SUPPRESS_WARNINGS_KEY_UNCHECKED
private int findFirst() {
 if (firstIndexValid) return this.firstIndex;
 firstIndexValid = true;
 int i = size;
 int firstIndex = --i;
 KEY_GENERIC_TYPE first = array[firstIndex];

 if (c == null) { while(i-- != 0) if (KEY_LESS(array[i], first)) first = array[firstIndex = i]; }
 else while(i-- != 0) { if (c.compare(array[i], first) < 0) first = array[firstIndex = i]; }

 return this.firstIndex = firstIndex;
}

private void ensureNonEmpty() {
 if (size == 0) throw new NoSuchElementException();
}

@Override
SUPPRESS_WARNINGS_KEY_UNCHECKED
public void enqueue(KEY_GENERIC_TYPE x) {
 if (size == array.length) array = ARRAYS.grow(array, size + 1);
 if (firstIndexValid) {
```

```
  if (c == null) { if (KEY_LESS(x, array[firstIndex])) firstIndex = size; }
  else if (c.compare(x, array[firstIndex]) < 0) firstIndex = size;
 }
 else firstIndexValid = false;
 array[size++] = x;
}

@Override
public KEY_GENERIC_TYPE DEQUEUE() {
 ensureNonEmpty();
 final int first = findFirst();
 final KEY_GENERIC_TYPE result = array[first];
 System.arraycopy(array, first + 1, array, first, --size - first);
#if KEY_CLASS_Object
 array[size] = null;
#endif
 firstIndexValid = false;
 return result;
}

@Override
public KEY_GENERIC_TYPE FIRST() {
 ensureNonEmpty();
 return array[findFirst()];
}

@Override
public void changed() {
 ensureNonEmpty();
 firstIndexValid = false;
}

@Override
public int size() { return size; }

@Override
public void clear() {
#if KEY_CLASS_Object
 Arrays.fill(array, 0, size, null);
#endif
 size = 0;
 firstIndexValid = false;
}

/** Trims the underlying array so that it has exactly {@link #size()} elements. */

public void trim() {
 array = ARRAYS.trim(array, size);
```

```
 }

 @Override
 public KEY_COMPARATOR KEY_SUPER_GENERIC comparator() { return c; }

 private void writeObject(java.io.ObjectOutputStream s) throws java.io.IOException {
  s.defaultWriteObject();
  s.writeInt(array.length);
  for(int i = 0; i < size; i++) s.WRITE_KEY(array[i]);
 }

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 private void readObject(java.io.ObjectInputStream s) throws java.io.IOException, ClassNotFoundException {
  s.defaultReadObject();
  array = KEY_GENERIC_ARRAY_CAST new KEY_TYPE[s.readInt()];
  for(int i = 0; i < size; i++) array[i] = KEY_GENERIC_CAST s.READ_KEY();
 }
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/ArrayPriorityQueue.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2003-2017 Paolo Boldi and Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import java.util.Comparator;

/** A class providing static methods and objects that do useful things with comparators.
*/
```

```
public final class COMPARATORS {

 private COMPARATORS() {}

 /** A type-specific comparator mimicking the natural order. */
#if KEYS_REFERENCE
 SUPPRESS_WARNINGS_KEY_UNCHECKED_RAWTYPES
 protected static class NaturalImplicitComparator implements Comparator, java.io.Serializable {
#else
 protected static class NaturalImplicitComparator implements KEY_COMPARATOR KEY_GENERIC,
java.io.Serializable {
#endif
  private static final long serialVersionUID = 1L;

  @Override
  public final int compare(final KEY_TYPE a, final KEY_TYPE b) {
#if KEYS_PRIMITIVE
   return KEY_CMP(a, b);
#else
   return ((Comparable)a).compareTo(b);
#endif
  }
  private Object readResolve() { return NATURAL_COMPARATOR; }
 };

 SUPPRESS_WARNINGS_KEY_RAWTYPES
 public static final KEY_COMPARATOR NATURAL_COMPARATOR = new NaturalImplicitComparator();

 /** A type-specific comparator mimicking the opposite of the natural order. */
#if KEYS_REFERENCE
 SUPPRESS_WARNINGS_KEY_UNCHECKED_RAWTYPES
 protected static class OppositeImplicitComparator implements Comparator, java.io.Serializable {
#else
 protected static class OppositeImplicitComparator implements KEY_COMPARATOR KEY_GENERIC,
java.io.Serializable {
#endif
  private static final long serialVersionUID = 1L;

  @Override
  public final int compare(final KEY_TYPE a, final KEY_TYPE b) {
#if KEYS_PRIMITIVE
   return - KEY_CMP(a, b);
#else
   return ((Comparable)b).compareTo(a);
#endif
  }
  private Object readResolve() { return OPPOSITE_COMPARATOR; }
 };
```

```
 SUPPRESS_WARNINGS_KEY_RAWTYPES
 public static final KEY_COMPARATOR OPPOSITE_COMPARATOR = new OppositeImplicitComparator();

#if KEYS_REFERENCE
 protected static class OppositeComparator KEY_GENERIC implements Comparator KEY_GENERIC,
java.io.Serializable {
#else
 protected static class OppositeComparator KEY_GENERIC implements KEY_COMPARATOR KEY_GENERIC,
java.io.Serializable {
#endif
  private static final long serialVersionUID = 1L;

  private final KEY_COMPARATOR KEY_GENERIC comparator;

  protected OppositeComparator(final KEY_COMPARATOR KEY_GENERIC c) { comparator = c; }

  @Override
  public final int compare(final KEY_GENERIC_TYPE a, final KEY_GENERIC_TYPE b) { return
comparator.compare(b, a); }
 };

 /** Returns a comparator representing the opposite order of the given comparator.
  *
  * @param c a comparator.
  * @return a comparator representing the opposite order of {@code c}.
  */
 public static KEY_GENERIC KEY_COMPARATOR KEY_GENERIC oppositeComparator(final
KEY_COMPARATOR KEY_GENERIC c) { return new OppositeComparator KEY_GENERIC_DIAMOND(c); }

#if KEYS_PRIMITIVE
 /** Returns a type-specific comparator that is equivalent to the given comparator.
  *
  * @param c a comparator, or {@code null}.
  * @return a type-specific comparator representing the order of {@code c}.
  */
 public static KEY_COMPARATOR AS_KEY_COMPARATOR(final Comparator<? super
KEY_GENERIC_CLASS> c) {
  if (c == null || c instanceof KEY_COMPARATOR) return (KEY_COMPARATOR) c;

  return new KEY_COMPARATOR() {
    @Override
    public int compare(KEY_GENERIC_TYPE x, KEY_GENERIC_TYPE y) { return c.compare(KEY2OBJ(x),
KEY2OBJ(y)); }
    @SuppressWarnings("deprecation")
    @Override
    public int compare(KEY_GENERIC_CLASS x, KEY_GENERIC_CLASS y) { return c.compare(x, y); }
   };
```

```
 }
#endif
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/Comparators.drv
No license file was found, but licenses were detected in source scan.


```
package PACKAGE;

#if KEY_CLASS_Object
import java.util.Comparator;
#endif

import it.unimi.dsi.fastutil.ints.IntArrays;

import java.util.Arrays;
import java.util.NoSuchElementException;

/** A type-specific heap-based indirect priority queue.
*
* <p>Instances of this class use an additional <em>inversion array</em>, of the same length of the reference array,
* to keep track of the heap position containing a given element of the reference array. The priority queue is
* represented using a heap. The heap is enlarged as needed, but it is never
* shrunk. Use the {@link #trim()} method to reduce its size, if necessary.
*
* <p>This implementation does <em>not</em> allow one to enqueue several times the same index.
*/

public class HEAP_INDIRECT_PRIORITY_QUEUE KEY_GENERIC extends
```

```
HEAP_SEMI_INDIRECT_PRIORITY_QUEUE KEY_GENERIC {

 /** The inversion array. */
 protected final int inv[];

 /** Creates a new empty queue with a given capacity and comparator.
  *
  * @param refArray the reference array.
  * @param capacity the initial capacity of this queue.
  * @param c the comparator used in this queue, or {@code null} for the natural order.
  */
 public HEAP_INDIRECT_PRIORITY_QUEUE(KEY_GENERIC_TYPE[] refArray, int capacity,
KEY_COMPARATOR KEY_SUPER_GENERIC c) {
  super(refArray, capacity, c);
  if (capacity > 0) this.heap = new int[capacity];

  this.c = c;

  this.inv = new int[refArray.length];
  Arrays.fill(inv, -1);
 }

 /** Creates a new empty queue with a given capacity and using the natural order.
  *
  * @param refArray the reference array.
  * @param capacity the initial capacity of this queue.
  */
 public HEAP_INDIRECT_PRIORITY_QUEUE(KEY_GENERIC_TYPE[] refArray, int capacity) {
  this(refArray, capacity, null);
 }

 /** Creates a new empty queue with capacity equal to the length of the reference array and a given comparator.
  *
  * @param refArray the reference array.
  * @param c the comparator used in this queue, or {@code null} for the natural order.
  */
 public HEAP_INDIRECT_PRIORITY_QUEUE(KEY_GENERIC_TYPE[] refArray, KEY_COMPARATOR
KEY_SUPER_GENERIC c) {
  this(refArray, refArray.length, c);
 }

 /** Creates a new empty queue with capacity equal to the length of the reference array and using the natural order.
  * @param refArray the reference array.
  */
 public HEAP_INDIRECT_PRIORITY_QUEUE(KEY_GENERIC_TYPE[] refArray) {
  this(refArray, refArray.length, null);
 }
```

```
 /** Wraps a given array in a queue using a given comparator.
  *
  * <p>The queue returned by this method will be backed by the given array.
  * The first {@code size} element of the array will be rearranged so to form a heap (this is
  * more efficient than enqueing the elements of {@code a} one by one).
  *
  * @param refArray the reference array.
  * @param a an array of indices into {@code refArray}.
  * @param size the number of elements to be included in the queue.
  * @param c the comparator used in this queue, or {@code null} for the natural order.
  */
 public HEAP_INDIRECT_PRIORITY_QUEUE(final KEY_GENERIC_TYPE[] refArray, final int[] a, final int
size, final KEY_COMPARATOR KEY_SUPER_GENERIC c) {
  this(refArray, 0, c);
  this.heap = a;
  this.size = size;
  int i = size;
  while(i-- != 0) {
   if (inv[a[i]] != -1) throw new IllegalArgumentException("Index " + a[i] + " appears twice in the heap");
   inv[a[i]] = i;
  }
  INDIRECT_HEAPS.makeHeap(refArray, a, inv, size, c);
 }


 /** Wraps a given array in a queue using a given comparator.
  *
  * <p>The queue returned by this method will be backed by the given array.
  * The elements of the array will be rearranged so to form a heap (this is
  * more efficient than enqueing the elements of {@code a} one by one).
  *
  * @param refArray the reference array.
  * @param a an array of indices into {@code refArray}.
  * @param c the comparator used in this queue, or {@code null} for the natural order.
  */
 public HEAP_INDIRECT_PRIORITY_QUEUE(final KEY_GENERIC_TYPE[] refArray, final int[] a, final
KEY_COMPARATOR KEY_SUPER_GENERIC c) {
  this(refArray, a, a.length, c);
 }

 /** Wraps a given array in a queue using the natural order.
  *
  * <p>The queue returned by this method will be backed by the given array.
  * The first {@code size} element of the array will be rearranged so to form a heap (this is
  * more efficient than enqueing the elements of {@code a} one by one).
  *
  * @param refArray the reference array.
```

```
 * @param a an array of indices into {@code refArray}.
 * @param size the number of elements to be included in the queue.
 */
public HEAP_INDIRECT_PRIORITY_QUEUE(final KEY_GENERIC_TYPE[] refArray, final int[] a, int size) {
 this(refArray, a, size, null);
}


/** Wraps a given array in a queue using the natural order.
 *
 * <p>The queue returned by this method will be backed by the given array.
 * The elements of the array will be rearranged so to form a heap (this is
 * more efficient than enqueing the elements of {@code a} one by one).
 *
 * @param refArray the reference array.
 * @param a an array of indices into {@code refArray}.
 */
public HEAP_INDIRECT_PRIORITY_QUEUE(final KEY_GENERIC_TYPE[] refArray, final int[] a) {
 this(refArray, a, a.length);
}

@Override
public void enqueue(final int x) {
 if (inv[x] >= 0) throw new IllegalArgumentException("Index " + x + " belongs to the queue");

 if (size == heap.length) heap = IntArrays.grow(heap, size + 1);

 inv[heap[size] = x] = size++;

 INDIRECT_HEAPS.upHeap(refArray, heap, inv, size, size - 1, c);
}

@Override
public boolean contains(final int index) {
 return inv[index] >= 0;
}

@Override
public int dequeue() {
 if (size == 0) throw new NoSuchElementException();
 final int result = heap[0];
 if (--size != 0) inv[heap[0] = heap[size]] = 0;
 inv[result] = -1;

 if (size != 0) INDIRECT_HEAPS.downHeap(refArray, heap, inv, size, 0, c);
 return result;
}
```

```
@Override
public void changed() {
 INDIRECT_HEAPS.downHeap(refArray, heap, inv, size, 0, c);
}

@Override
public void changed(final int index) {
 final int pos = inv[index];
 if (pos < 0) throw new IllegalArgumentException("Index " + index + " does not belong to the queue");
 final int newPos = INDIRECT_HEAPS.upHeap(refArray, heap, inv, size, pos, c);
 INDIRECT_HEAPS.downHeap(refArray, heap, inv, size, newPos, c);
}

/** Rebuilds this queue in a bottom-up fashion (in linear time). */
@Override
public void allChanged() { INDIRECT_HEAPS.makeHeap(refArray, heap, inv, size, c); }

@Override
public boolean remove(final int index) {
 final int result = inv[index];
 if (result < 0) return false;
 inv[index] = -1;

 if (result < --size) {
  inv[heap[result] = heap[size]] = result;
  final int newPos = INDIRECT_HEAPS.upHeap(refArray, heap, inv, size, result, c);
  INDIRECT_HEAPS.downHeap(refArray, heap, inv, size, newPos, c);
 }

 return true;
}

@Override
public void clear() {
 size = 0;
 Arrays.fill(inv, -1);
}

#ifdef TEST

 /** The original class, now just used for testing. */

 private static class TestQueue {

  /** The reference array */
  private KEY_TYPE refArray[];
  /** Its length */
  private int N;
```

```
 /** The number of elements in the heaps */
 private int n;
 /** The two comparators */
 private KEY_COMPARATOR primaryComp, secondaryComp;
 /** Two indirect heaps are used, called {@code primary} and {@code secondary}. Each of them contains
  a permutation of {@code n} among the indices 0, 1, ..., {@code N}-1 in such a way that the corresponding
  objects be sorted with respect to the two comparators.
  We also need an array {@code inSec[]} so that {@code inSec[k]} is the index of {@code secondary}
  containing {@code k}.
 */
 private int primary[], secondary[], inSec[];

 /** Builds a double indirect priority queue.
  *  @param refArray The reference array.
  *  @param primaryComp The primary comparator.
  *  @param secondaryComp The secondary comparator.
  */
 public TestQueue(KEY_TYPE refArray[], KEY_COMPARATOR primaryComp, KEY_COMPARATOR
secondaryComp) {
  this.refArray = refArray;
  this.N = refArray.length;
  assert this.N != 0;
  this.n = 0;
  this.primaryComp = primaryComp;
  this.secondaryComp = secondaryComp;
  this.primary = new int[N];
  this.secondary = new int[N];
  this.inSec = new int[N];
  java.util.Arrays.fill(inSec, -1);
 }

 /** Adds an index to the queue. Notice that the index should not be already present in the queue.
  *  @param i The index to be added
  */
 public void add(int i) {
  if (i < 0 || i >= refArray.length) throw new IndexOutOfBoundsException();
  if (inSec[i] >= 0) throw new IllegalArgumentException();
  primary[n] = i;
  secondary[n] = i; inSec[i] = n;
  n++;
  swimPrimary(n-1);
  swimSecondary(n-1);
 }

 /** Heapify the primary heap.
  *  @param i The index of the heap to be heapified.
  */
 private void heapifyPrimary(int i) {
```

```
  int dep = primary[i];
  int child;

  while ((child = 2*i+1) < n) {
   if (child+1 < n && primaryComp.compare(refArray[primary[child+1]], refArray[primary[child]]) < 0) child++;
   if (primaryComp.compare(refArray[dep], refArray[primary[child]]) <= 0) break;
   primary[i] = primary[child];
   i = child;
  }
  primary[i] = dep;
 }

 /** Heapify the secondary heap.
  *  @param i The index of the heap to be heapified.
  */
 private void heapifySecondary(int i) {
  int dep = secondary[i];
  int child;

  while ((child = 2*i+1) < n) {
   if (child+1 < n && secondaryComp.compare(refArray[secondary[child+1]], refArray[secondary[child]]) < 0)
child++;
   if (secondaryComp.compare(refArray[dep], refArray[secondary[child]]) <= 0) break;
   secondary[i] = secondary[child]; inSec[secondary[i]] = i;
   i = child;
  }
  secondary[i] = dep; inSec[secondary[i]] = i;
 }

 /** Swim and heapify the primary heap.
  *  @param i The index to be moved.
  */
 private void swimPrimary(int i) {
  int dep = primary[i];
  int parent;

  while (i != 0 && (parent = (i - 1) / 2) >= 0) {
   if (primaryComp.compare(refArray[primary[parent]], refArray[dep]) <= 0) break;
   primary[i] = primary[parent];
   i = parent;
  }
  primary[i] = dep;
  heapifyPrimary(i);
 }

 /** Swim and heapify the secondary heap.
  *  @param i The index to be moved.
  */
```

```
private void swimSecondary(int i) {
 int dep = secondary[i];
 int parent;

 while (i != 0 && (parent = (i - 1) / 2) >= 0) {
  if (secondaryComp.compare(refArray[secondary[parent]], refArray[dep]) <= 0) break;
  secondary[i] = secondary[parent]; inSec[secondary[i]] = i;
  i = parent;
 }
 secondary[i] = dep; inSec[secondary[i]] = i;
 heapifySecondary(i);
}

/** Returns the minimum element with respect to the primary comparator.
 @return the minimum element.
*/
public int top() {
 if (n == 0) throw new java.util.NoSuchElementException();
 return primary[0];
}

/** Returns the minimum element with respect to the secondary comparator.
 @return the minimum element.
*/
public int secTop() {
 if (n == 0) throw new java.util.NoSuchElementException();
 return secondary[0];
}

/** Removes the minimum element with respect to the primary comparator.
 *  @return the removed element.
 */
public boolean remove() {
 if (n == 0) throw new java.util.NoSuchElementException();
 if (inSec[primary[0]] == -1) return false;
 int result = primary[0];
 int ins = inSec[result];
 inSec[result] = -1;
 // Copy a leaf
 primary[0] = primary[n-1];
 if (ins == n-1) {
  n--;
  heapifyPrimary(0);
  return true;
 }
 secondary[ins] = secondary[n-1];
 inSec[secondary[ins]] = ins;
 // Heapify
```

```
 n--;
 heapifyPrimary(0);
 swimSecondary(ins);
 return true;
}

public void clear() {
 while(size() != 0) remove();
}

public void remove(int index) {
 if (index >= refArray.length) throw new IndexOutOfBoundsException();
 if (inSec[index] == -1) return;
 int ins = inSec[index];
 inSec[index] = -1;
 // Copy a leaf
 primary[ins] = primary[n-1];
 if (ins == n-1) {
  n--;
  swimPrimary(ins);
  return;
 }
 secondary[ins] = secondary[n-1];
 inSec[secondary[ins]] = ins;
 // Heapify
 n--;
 swimPrimary(ins);
 swimSecondary(ins);
}

/** Signals that the minimum element with respect to the comparator has changed.
 */
public void change() {
 if (n == 0) throw new java.util.NoSuchElementException();
 if (inSec[primary[0]] == -1) throw new IllegalArgumentException();
 int ins = inSec[primary[0]];
 heapifyPrimary(0);
 swimSecondary(ins);
}

public void change(int index) {
 if (index >= refArray.length) throw new IndexOutOfBoundsException();
 if (inSec[index] == -1) throw new IllegalArgumentException();
 if (n == 0) throw new java.util.NoSuchElementException();
 int ins = inSec[index];
 swimPrimary(ins);
 swimSecondary(ins);
}
```

```
 /** Returns the number of elements in the queue.
  *  @return the size of the queue
  */
 public int size() {
  return n;
 }



 public String toString() {
  String s = "[";
  for (int i = 0; i < n; i++)
   s += refArray[primary[i]]+", ";
  return s+ "]";
 }
}

 private static long seed = System.currentTimeMillis();
 private static java.util.Random r = new java.util.Random(seed);

 private static KEY_TYPE genKey() {
#if KEY_CLASS_Byte || KEY_CLASS_Short || KEY_CLASS_Character
  return (KEY_TYPE)(r.nextInt());
#elif KEYS_PRIMITIVE
  return r.NEXT_KEY();
#elif KEY_CLASS_Object
  return Integer.toBinaryString(r.nextInt());
#else
  return new java.io.Serializable() {};
#endif
 }

 private static java.text.NumberFormat format = new java.text.DecimalFormat("#,###.00");
 private static java.text.FieldPosition p = new java.text.FieldPosition(0);

 private static String format(double d) {
  StringBuffer s = new StringBuffer();
  return format.format(d, s, p).toString();
 }

 private static void speedTest(int n, boolean comp) {
  System.out.println("There are presently no speed tests for this class.");
 }


 private static void fatal(String msg) {
  System.out.println(msg);
```

```
 System.exit(1);
}

private static void ensure(boolean cond, String msg) {
 if (cond) return;
 fatal(msg);
}

private static boolean heapEqual(int[] a, int[] b, int sizea, int sizeb) {
 if (sizea != sizeb) return false;
 while(sizea-- != 0) if (a[sizea] != b[sizea]) return false;
 return true;
}

private static boolean invEqual(int inva[], int[] invb) {
 int i = inva.length;
 while(i-- != 0) if (inva[i] != invb[i]) return false;
 return true;
}

protected static void runTest(int n) {
 long ms;
 Exception mThrowsIllegal, tThrowsIllegal, mThrowsOutOfBounds, tThrowsOutOfBounds, mThrowsNoElement,
tThrowsNoElement;
 int rm = 0, rt = 0;
 KEY_TYPE[] refArray = new KEY_TYPE[n];

 for(int i = 0; i < n; i++) refArray[i] = genKey();

 HEAP_INDIRECT_PRIORITY_QUEUE m = new HEAP_INDIRECT_PRIORITY_QUEUE(refArray,
COMPARATORS.NATURAL_COMPARATOR);
 TestQueue t = new TestQueue(refArray, COMPARATORS.NATURAL_COMPARATOR,
COMPARATORS.NATURAL_COMPARATOR);

 /* We add pairs to t. */
 for(int i = 0; i < n / 2;  i++) {
  t.add(i);
  m.enqueue(i);
 }

 ensure(heapEqual(m.heap, t.primary, m.size(), t.size()), "Error (" + seed + "): m and t differ after creation (" + m + ",
" + t + ")");
 ensure(invEqual(m.inv, t.inSec), "Error (" + seed + "): m and t differ in inversion arrays after creation (" +
java.util.Arrays.toString(m.inv) + ", " + java.util.Arrays.toString(t.inSec) + ")");

 /* Now we add and remove random data in m and t, checking that the result is the same. */

 for(int i=0; i<2*n;  i++) {
```

```
  if (r.nextDouble() < 0.01) {
   t.clear();
   m.clear();
   for(int j = 0; j < n / 2;  j++) {
    t.add(j);
    m.enqueue(j);
   }
  }

  int T = r.nextInt(2 * n);

  mThrowsNoElement = tThrowsNoElement = mThrowsOutOfBounds = tThrowsOutOfBounds = mThrowsIllegal = tThrowsIllegal = null;

  try {
   m.enqueue(T);
  }
  catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }
  catch (IllegalArgumentException e) { mThrowsIllegal = e; }

  try {
   t.add(T);
  }
  catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }
  catch (IllegalArgumentException e) { tThrowsIllegal = e; }

  ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + seed + "): enqueue() divergence in IndexOutOfBoundsException for " + T + " (" + mThrowsOutOfBounds + ", " + tThrowsOutOfBounds + ")");
  ensure((mThrowsIllegal == null) == (tThrowsIllegal == null), "Error (" + seed + "): enqueue() divergence in IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");

  ensure(heapEqual(m.heap, t.primary, m.size(), t.size()), "Error (" + seed + "): m and t differ after enqueue (" + m + ", " + t + ")");
  ensure(invEqual(m.inv, t.inSec), "Error (" + seed + "): m and t differ in inversion arrays after enqueue (" + java.util.Arrays.toString(m.inv) + ", " + java.util.Arrays.toString(t.inSec) + ")");

  if (m.size() != 0) {
   ensure(m.first() == t.top(), "Error (" + seed + "): m and t differ in first element after enqueue (" + m.first() + ", " + t.top() + ")");
  }

  mThrowsNoElement = tThrowsNoElement = mThrowsOutOfBounds = tThrowsOutOfBounds = mThrowsIllegal = tThrowsIllegal = null;

  try {
   rm = m.dequeue();
  }
```

```
  catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }
  catch (IllegalArgumentException e) { mThrowsIllegal = e; }
  catch (java.util.NoSuchElementException e) { mThrowsNoElement = e; }

  try {
   rt = t.top();
   t.remove();
  }
  catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }
  catch (IllegalArgumentException e) { tThrowsIllegal = e; }
  catch (java.util.NoSuchElementException e) { tThrowsNoElement = e; }

  ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + seed + "): dequeue()
divergence in IndexOutOfBoundsException (" + mThrowsOutOfBounds + ", " + tThrowsOutOfBounds + ")");
  ensure((mThrowsIllegal == null) == (tThrowsIllegal == null), "Error (" + seed + "): dequeue() divergence in
IllegalArgumentException  (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
  ensure((mThrowsNoElement == null) == (tThrowsNoElement == null), "Error (" + seed + "): dequeue() divergence
in java.util.NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
  if (mThrowsOutOfBounds == null) ensure(rt == rm , "Error (" + seed + "): divergence in dequeue() between t and m
(" + rt + ", " + rm + ")");

  ensure(heapEqual(m.heap, t.primary, m.size(), t.size()), "Error (" + seed + "): m and t differ after dequeue (" + m +
", " + t + ")");
  ensure(invEqual(m.inv, t.inSec), "Error (" + seed + "): m and t differ in inversion arrays after dequeue (" +
java.util.Arrays.toString(m.inv) + ", " + java.util.Arrays.toString(t.inSec) + ")");

  if (m.size() != 0) {
   ensure(m.first() == t.top(), "Error (" + seed + "): m and t differ in first element after dequeue (" + m.first() + ", " +
t.top() + ")");
  }


  int pos = r.nextInt(n * 2);

  try {
   m.remove(pos);
  }
  catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }
  catch (IllegalArgumentException e) { mThrowsIllegal = e; }
  catch (java.util.NoSuchElementException e) { mThrowsNoElement = e; }

  try {
   t.remove(pos);
  }
  catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }
  catch (IllegalArgumentException e) { tThrowsIllegal = e; }
  catch (java.util.NoSuchElementException e) { tThrowsNoElement = e; }
```

```
  ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + seed + "): remove(int)
divergence in IndexOutOfBoundsException (" + mThrowsOutOfBounds + ", " + tThrowsOutOfBounds + ")");
  ensure((mThrowsIllegal == null) == (tThrowsIllegal == null), "Error (" + seed + "): remove(int) divergence in
IllegalArgumentException  (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
  ensure((mThrowsNoElement == null) == (tThrowsNoElement == null), "Error (" + seed + "): remove(int)
divergence in java.util.NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
  if (mThrowsOutOfBounds == null) ensure(rt == rm , "Error (" + seed + "): divergence in remove(int) between t and
m (" + rt + ", " + rm + ")");

  ensure(heapEqual(m.heap, t.primary, m.size(), t.size()), "Error (" + seed + "): m and t differ after remove(int) (" + m
+ ", " + t + ")");
  ensure(invEqual(m.inv, t.inSec), "Error (" + seed + "): m and t differ in inversion arrays after remove(int) (" +
java.util.Arrays.toString(m.inv) + ", " + java.util.Arrays.toString(t.inSec) + ")");

  if (m.size() != 0) {
   ensure(m.first() == t.top(), "Error (" + seed + "): m and t differ in first element after remove(int) (" + m.first() + ", "
+ t.top() + ")");
  }



  pos = r.nextInt(n * 2);

  try {
   m.changed(pos);
  }
  catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }
  catch (IllegalArgumentException e) { mThrowsIllegal = e; }
  catch (java.util.NoSuchElementException e) { mThrowsNoElement = e; }

  try {
   t.change(pos);
  }
  catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }
  catch (IllegalArgumentException e) { tThrowsIllegal = e; }
  catch (java.util.NoSuchElementException e) { tThrowsNoElement = e; }

  ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + seed + "): change(int)
divergence in IndexOutOfBoundsException (" + mThrowsOutOfBounds + ", " + tThrowsOutOfBounds + ")");
  ensure((mThrowsIllegal == null) == (tThrowsIllegal == null), "Error (" + seed + "): change(int) divergence in
IllegalArgumentException  (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
  ensure((mThrowsNoElement == null) == (tThrowsNoElement == null), "Error (" + seed + "): change(int)
divergence in java.util.NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
  if (mThrowsOutOfBounds == null) ensure(rt == rm , "Error (" + seed + "): divergence in change(int) between t and
m (" + rt + ", " + rm + ")");

  ensure(heapEqual(m.heap, t.primary, m.size(), t.size()), "Error (" + seed + "): m and t differ after change(int) (" + m
+ ", " + t + ")");
  ensure(invEqual(m.inv, t.inSec), "Error (" + seed + "): m and t differ in inversion arrays after change(int) (" +
```

```
java.util.Arrays.toString(m.inv) + ", " + java.util.Arrays.toString(t.inSec) + ")");

   if (m.size() != 0) {
    ensure(m.first() == t.top(), "Error (" + seed + "): m and t differ in first element after change(int) (" + m.first() + ", " +
t.top() + ")");
   }

   if (m.size() != 0) {

    refArray[m.first()] = genKey();

    m.changed();
    t.change();

    ensure(heapEqual(m.heap, t.primary, m.size(), t.size()), "Error (" + seed + "): m and t differ after change (" + m + ",
" + t + ")");
    ensure(invEqual(m.inv, t.inSec), "Error (" + seed + "): m and t differ in inversion arrays after change (" +
java.util.Arrays.toString(m.inv) + ", " + java.util.Arrays.toString(t.inSec) + ")");

    if (m.size() != 0) {
     ensure(m.first() == t.top(), "Error (" + seed + "): m and t differ in first element after change (" + m.first() + ", " +
t.top() + ")");
    }
   }
  }


  /* Now we check that m actually holds the same data. */

  m.clear();
  ensure(m.isEmpty(), "Error (" + seed + "): m is not empty after clear()");

  System.out.println("Test OK");
 }



 public static void main(String args[]) {
  int n  = Integer.parseInt(args[1]);
  if (args.length > 2) r = new java.util.Random(seed = Long.parseLong(args[2]));


  try {
   if ("speedTest".equals(args[0]) || "speedComp".equals(args[0])) speedTest(n, "speedComp".equals(args[0]));
   else if ("test".equals(args[0])) runTest(n);
  } catch(Throwable e) {
   e.printStackTrace(System.err);
   System.err.println("seed: " + seed);
```

```
  }
 }

#endif

}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/HeapIndirectPriorityQueue.drv
No license file was found, but licenses were detected in source scan.

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a

copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object
form, that is based on (or derived from) the Work and for which the
editorial revisions, annotations, elaborations, or other modifications
represent, as a whole, an original work of authorship. For the purposes
of this License, Derivative Works shall not include works that remain
separable from, or merely link (or bind by name) to the interfaces of,
the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity
on behalf of whom a Contribution has been received by Licensor and
subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
copyright license to reproduce, prepare Derivative Works of,
publicly display, publicly perform, sublicense, and distribute the
Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
(except as stated in this section) patent license to make, have made,
use, offer to sell, sell, import, and otherwise transfer the Work,
where such license applies only to those patent claims licensable
by such Contributor that are necessarily infringed by their
Contribution(s) alone or by combination of their Contribution(s)
with the Work to which such Contribution(s) was submitted. If You
institute patent litigation against any entity (including a
cross-claim or counterclaim in a lawsuit) alleging that the Work
or a Contribution incorporated within the Work constitutes direct

or contributory patent infringement, then any patent licenses
granted to You under this License for that Work shall terminate
as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
Work or Derivative Works thereof in any medium, with or without
modifications, and in Source or Object form, provided that You
meet the following conditions:

(a) You must give any other recipients of the Work or
Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices
stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works
that You distribute, all copyright, patent, trademark, and
attribution notices from the Source form of the Work,
excluding those notices that do not pertain to any part of
the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its
distribution, then any Derivative Works that You distribute must
include a readable copy of the attribution notices contained
within such NOTICE file, excluding those notices that do not
pertain to any part of the Derivative Works, in at least one
of the following places: within a NOTICE text file distributed
as part of the Derivative Works; within the Source form or
documentation, if provided along with the Derivative Works; or,
within a display generated by the Derivative Works, if and
wherever such third-party notices normally appear. The contents
of the NOTICE file are for informational purposes only and
do not modify the License. You may add Your own attribution
notices within Derivative Works that You distribute, alongside
or as an addendum to the NOTICE text from the Work, provided
that such additional attribution notices cannot be construed
as modifying the License.

You may add Your own copyright statement to Your modifications and
may provide additional or different license terms and conditions
for use, reproduction, or distribution of Your modifications, or
for any such Derivative Works as a whole, provided Your use,
reproduction, and distribution of the Work otherwise complies with
the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
any Contribution intentionally submitted for inclusion in the Work
by You to the Licensor shall be under the terms and conditions of

this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify
the terms of any separate license agreement you may have executed
with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
names, trademarks, service marks, or product names of the Licensor,
except as required for reasonable and customary use in describing the
origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
agreed to in writing, Licensor provides the Work (and each
Contributor provides its Contributions) on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied, including, without limitation, any warranties or conditions
of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
PARTICULAR PURPOSE. You are solely responsible for determining the
appropriateness of using or redistributing the Work and assume any
risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
whether in tort (including negligence), contract, or otherwise,
unless required by applicable law (such as deliberate and grossly
negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special,
incidental, or consequential damages of any character arising as a
result of this License or out of the use or inability to use the
Work (including but not limited to damages for loss of goodwill,
work stoppage, computer failure or malfunction, or any and all
other commercial damages or losses), even if such Contributor
has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
the Work or Derivative Works thereof, You may choose to offer,
and charge a fee for, acceptance of support, warranty, indemnity,
or other liability obligations and/or rights consistent with this
License. However, in accepting such obligations, You may act only
on Your own behalf and on Your sole responsibility, not on behalf
of any other Contributor, and only if You agree to indemnify,
defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following

boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/LICENSE-2.0
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import it.unimi.dsi.fastutil.objects.ObjectIterator;
import it.unimi.dsi.fastutil.objects.ObjectIterable;
```

```
import it.unimi.dsi.fastutil.objects.ObjectSet;
import it.unimi.dsi.fastutil.objects.ObjectSets;

import VALUE_PACKAGE.VALUE_COLLECTION;
import VALUE_PACKAGE.VALUE_COLLECTIONS;
#if ! VALUE_CLASS_Object
import VALUE_PACKAGE.VALUE_SETS;
#endif

import java.util.Map;
import java.util.function.Consumer;
import PACKAGE.MAP.FastEntrySet;

/** A class providing static methods and objects that do useful things with type-specific maps.
*
* @see java.util.Collections
*/

public final class MAPS {

 private MAPS() {}

 /** Returns an iterator that will be {@linkplain FastEntrySet fast}, if possible, on the {@linkplain Map#entrySet() entry set} of the provided {@code map}.
  * @param map a map from which we will try to extract a (fast) iterator on the entry set.
  * @return an iterator on the entry set of the given map that will be fast, if possible.
  * @since 8.0.0
  */
 SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
 public static KEY_VALUE_GENERIC ObjectIterator<MAP.Entry KEY_VALUE_GENERIC> fastIterator(MAP KEY_VALUE_GENERIC map) {
  final ObjectSet<MAP.Entry KEY_VALUE_GENERIC> entries = map.ENTRYSET();
  return entries instanceof MAP.FastEntrySet ? ((MAP.FastEntrySet KEY_VALUE_GENERIC) entries).fastIterator() : entries.iterator();
 }

 /** Iterates {@linkplain FastEntrySet#fastForEach(Consumer) quickly}, if possible, on the {@linkplain Map#entrySet() entry set} of the provided {@code map}.
  * @param map a map on which we will try to iterate {@linkplain FastEntrySet#fastForEach(Consumer) quickly}.
  * @param consumer the consumer that will be passed to  {@link FastEntrySet#fastForEach(Consumer)}, if possible, or to {@link Iterable#forEach(Consumer)}.
  * @since 8.1.0
  */
 SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
 public static KEY_VALUE_GENERIC void fastForEach(MAP KEY_VALUE_GENERIC map, final Consumer<? super MAP.Entry KEY_VALUE_GENERIC> consumer) {
  final ObjectSet<MAP.Entry KEY_VALUE_GENERIC> entries = map.ENTRYSET();
  if (entries instanceof MAP.FastEntrySet) ((MAP.FastEntrySet KEY_VALUE_GENERIC)
```

```
entries).fastForEach(consumer);
  else entries.forEach(consumer);
 }

 /** Returns an iterable yielding an iterator that will be {@linkplain FastEntrySet fast}, if possible, on the
{@linkplain Map#entrySet() entry set} of the provided {@code map}.
  * @param map a map from which we will try to extract an iterable yielding a (fast) iterator on the entry set.
  * @return an iterable  yielding an iterator on the entry set of the given map that will be
  * fast, if possible.
  * @since 8.0.0
  */
 SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
 public static KEY_VALUE_GENERIC ObjectIterable<MAP.Entry KEY_VALUE_GENERIC> fastIterable(MAP
KEY_VALUE_GENERIC map) {
  final ObjectSet<MAP.Entry KEY_VALUE_GENERIC> entries = map.ENTRYSET();
  return entries instanceof MAP.FastEntrySet ? new ObjectIterable<MAP.Entry KEY_VALUE_GENERIC>() {
   public ObjectIterator<MAP.Entry KEY_VALUE_GENERIC> iterator() { return ((MAP.FastEntrySet
KEY_VALUE_GENERIC)entries).fastIterator(); }
   public void forEach(final Consumer<? super MAP.Entry KEY_VALUE_GENERIC> consumer) {
((MAP.FastEntrySet KEY_VALUE_GENERIC)entries).fastForEach(consumer); }
  } : entries;
 }

 /** An immutable class representing an empty type-specific map.
  *
  * <p>This class may be useful to implement your own in case you subclass
  * a type-specific map.
  */

 public static class EmptyMap KEY_VALUE_GENERIC extends FUNCTIONS.EmptyFunction
KEY_VALUE_GENERIC implements MAP KEY_VALUE_GENERIC, java.io.Serializable, Cloneable {

  private static final long serialVersionUID = -7046029254386353129L;

  protected EmptyMap() {}

  @Override
  public boolean containsValue(final VALUE_TYPE v) { return false; }

#if VALUES_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public boolean containsValue(final Object ov) { return false; }
#endif

  @Override
```

```
 public void putAll(final Map<? extends KEY_GENERIC_CLASS, ? extends VALUE_GENERIC_CLASS> m) {
throw new UnsupportedOperationException(); }

 @SuppressWarnings("unchecked")
 @Override
 public ObjectSet<MAP.Entry KEY_VALUE_GENERIC> ENTRYSET() { return ObjectSets.EMPTY_SET; }

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 @Override
 public SET KEY_GENERIC keySet() { return SETS.EMPTY_SET; }

 SUPPRESS_WARNINGS_VALUE_UNCHECKED
 @Override
 public VALUE_COLLECTION VALUE_GENERIC values() { return VALUE_SETS.EMPTY_SET; }

 @Override
 public Object clone() { return EMPTY_MAP; }

 @Override
 public boolean isEmpty() { return true; }

 @Override
 public int hashCode() { return 0; }

 @Override
 public boolean equals(final Object o) {
  if (! (o instanceof Map)) return false;
  return ((Map<?,?>)o).isEmpty();
 }

 @Override
 public String toString() { return "{}"; }
}


 /** An empty type-specific map (immutable). It is serializable and cloneable.
  */
 SUPPRESS_WARNINGS_KEY_VALUE_RAWTYPES
 public static final EmptyMap EMPTY_MAP = new EmptyMap();

#if KEYS_REFERENCE || VALUES_REFERENCE
 /** Returns an empty map (immutable). It is serializable and cloneable.
  *
  * <p>This method provides a typesafe access to {@link #EMPTY_MAP}.
  * @return an empty map (immutable).
  */
 @SuppressWarnings("unchecked")
 public static KEY_VALUE_GENERIC MAP KEY_VALUE_GENERIC emptyMap() {
```

```
 return EMPTY_MAP;
 }
#endif

 /** An immutable class representing a type-specific singleton map.
  *
  * <p>This class may be useful to implement your own in case you subclass
  * a type-specific map.
  */

 public static class Singleton KEY_VALUE_GENERIC extends FUNCTIONS.Singleton KEY_VALUE_GENERIC
implements MAP KEY_VALUE_GENERIC, java.io.Serializable, Cloneable {

  private static final long serialVersionUID = -7046029254386353129L;

  protected transient ObjectSet<MAP.Entry KEY_VALUE_GENERIC> entries;
  protected transient SET KEY_GENERIC keys;
  protected transient VALUE_COLLECTION VALUE_GENERIC values;

  protected Singleton(final KEY_GENERIC_TYPE key, final VALUE_GENERIC_TYPE value) {
   super(key, value);
  }

  @Override
  public boolean containsValue(final VALUE_TYPE v) { return VALUE_EQUALS(value, v); }
#if VALUES_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public boolean containsValue(final Object ov) { return VALUE_EQUALS(VALUE_OBJ2TYPE(ov), value); }
#endif

  @Override
  public void putAll(final Map<? extends KEY_GENERIC_CLASS, ? extends VALUE_GENERIC_CLASS> m) {
throw new UnsupportedOperationException(); }

  @Override
  public ObjectSet<MAP.Entry KEY_VALUE_GENERIC> ENTRYSET() { if (entries == null) entries =
ObjectSets.singleton(new ABSTRACT_MAP.BasicEntry KEY_VALUE_GENERIC_DIAMOND(key, value));
return entries; }

#if KEYS_PRIMITIVE || VALUES_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
#else
  /** {@inheritDoc} */
```

```
#endif
  @Override
  @SuppressWarnings({ "rawtypes", "unchecked" })
  public ObjectSet<Map.Entry<KEY_GENERIC_CLASS, VALUE_GENERIC_CLASS>> entrySet() { return
(ObjectSet)ENTRYSET(); }

  @Override
  public SET KEY_GENERIC keySet() { if (keys == null) keys = SETS.singleton(key); return keys; }

  @Override
  public VALUE_COLLECTION VALUE_GENERIC values() { if (values == null) values =
VALUE_SETS.singleton(value); return values; }

  @Override
  public boolean isEmpty() { return false; }

  @Override
  public int hashCode() { return KEY2JAVAHASH(key) ^ VALUE2JAVAHASH(value); }

  @Override
  public boolean equals(final Object o) {
   if (o == this) return true;
   if (! (o instanceof Map)) return false;

   Map<?,?> m = (Map<?,?>)o;
   if (m.size() != 1) return false;
   return m.entrySet().iterator().next().equals(entrySet().iterator().next());
  }

  @Override
  public String toString() { return "{" + key + "=>" + value + "}"; }
 }

 /** Returns a type-specific immutable map containing only the specified pair. The returned map is serializable and
cloneable.
  *
  * <p>Note that albeit the returned map is immutable, its default return value may be changed.
  *
  * @param key the only key of the returned map.
  * @param value the only value of the returned map.
  * @return a type-specific immutable map containing just the pair {@code &lt;key,value&gt;}.
  */

 public static KEY_VALUE_GENERIC MAP KEY_VALUE_GENERIC singleton(final KEY_GENERIC_TYPE
key, VALUE_GENERIC_TYPE value) { return new Singleton KEY_VALUE_GENERIC_DIAMOND(key, value);
}

#if KEYS_PRIMITIVE || VALUES_PRIMITIVE
```

```
 /** Returns a type-specific immutable map containing only the specified pair. The returned map is serializable and cloneable.
  *
  * <p>Note that albeit the returned map is immutable, its default return value may be changed.
  *
  * @param key the only key of the returned map.
  * @param value the only value of the returned map.
  * @return a type-specific immutable map containing just the pair {@code &lt;key,value&gt;}.
  */

 public static KEY_VALUE_GENERIC MAP KEY_VALUE_GENERIC singleton(final KEY_GENERIC_CLASS key, final VALUE_GENERIC_CLASS value) { return new Singleton KEY_VALUE_GENERIC_DIAMOND(KEY_CLASS2TYPE(key), VALUE_CLASS2TYPE(value)); }

#endif


 /** A synchronized wrapper class for maps. */

 public static class SynchronizedMap KEY_VALUE_GENERIC extends FUNCTIONS.SynchronizedFunction KEY_VALUE_GENERIC implements MAP KEY_VALUE_GENERIC, java.io.Serializable {

  private static final long serialVersionUID = -7046029254386353129L;

  protected final MAP KEY_VALUE_GENERIC map;

  protected transient ObjectSet<MAP.Entry KEY_VALUE_GENERIC> entries;
  protected transient SET KEY_GENERIC keys;
  protected transient VALUE_COLLECTION VALUE_GENERIC values;

  protected SynchronizedMap(final MAP KEY_VALUE_GENERIC m, final Object sync) {
   super(m, sync);
   this.map = m;
  }

  protected SynchronizedMap(final MAP KEY_VALUE_GENERIC m) {
   super(m);
   this.map = m;
  }

  @Override
  public boolean containsValue(final VALUE_TYPE v) { synchronized(sync) { return map.containsValue(v); } }

#if VALUES_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
```

```
  @Override
  public boolean containsValue(final Object ov) { synchronized(sync) { return map.containsValue(ov); } }
#endif

  @Override
  public void putAll(final Map<? extends KEY_GENERIC_CLASS, ? extends VALUE_GENERIC_CLASS> m) {
synchronized(sync) { map.putAll(m); } }

  @Override
  public ObjectSet<MAP.Entry KEY_VALUE_GENERIC> ENTRYSET() { synchronized(sync) { if (entries == null)
entries = ObjectSets.synchronize(map.ENTRYSET(), sync); return entries; } }

#if KEYS_PRIMITIVE || VALUES_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
#else
  /** {@inheritDoc} */
#endif
  @Override
  @SuppressWarnings({ "unchecked", "rawtypes" })
  public ObjectSet<Map.Entry<KEY_GENERIC_CLASS, VALUE_GENERIC_CLASS>> entrySet() { return
(ObjectSet)ENTRYSET(); }

  @Override
  public SET KEY_GENERIC keySet() {
   synchronized(sync) { if (keys == null) keys = SETS.synchronize(map.keySet(), sync); return keys; }
  }

  @Override
  public VALUE_COLLECTION VALUE_GENERIC values() {
   synchronized(sync) { if (values == null) return VALUE_COLLECTIONS.synchronize(map.values(), sync); return
values; }
  }

  @Override
  public boolean isEmpty() { synchronized(sync) { return map.isEmpty(); } }

  @Override
  public int hashCode() { synchronized(sync) { return map.hashCode(); } }

  @Override
  public boolean equals(final Object o) {
   if (o == this) return true;
   synchronized(sync) { return map.equals(o); }
  }

  private void writeObject(java.io.ObjectOutputStream s) throws java.io.IOException {
```

```
  synchronized(sync) { s.defaultWriteObject(); }
  }

  // Defaultable methods

  @Override
  public VALUE_GENERIC_TYPE getOrDefault(final KEY_TYPE key, final VALUE_GENERIC_TYPE
defaultValue) { synchronized(sync) { return map.getOrDefault(key, defaultValue); } }

  @Override
  public void forEach(final java.util.function.BiConsumer<? super KEY_GENERIC_CLASS, ? super
VALUE_GENERIC_CLASS> action) { synchronized (sync) { map.forEach(action); } }

  @Override
  public void replaceAll(final java.util.function.BiFunction<? super KEY_GENERIC_CLASS, ? super
VALUE_GENERIC_CLASS, ? extends VALUE_GENERIC_CLASS> function) { synchronized (sync) {
map.replaceAll(function); } }

  @Override
  public VALUE_GENERIC_TYPE putIfAbsent(final KEY_GENERIC_TYPE key, final VALUE_GENERIC_TYPE
value) { synchronized(sync) { return map.putIfAbsent(key, value); } }

  @Override
  public boolean remove(final KEY_TYPE key, final VALUE_TYPE value) { synchronized(sync) { return
map.remove(key, value); } }

  @Override
  public VALUE_GENERIC_TYPE replace(final KEY_GENERIC_TYPE key, final VALUE_GENERIC_TYPE
value) { synchronized(sync) { return map.replace(key, value); } }

  @Override
  public boolean replace(final KEY_GENERIC_TYPE key, final VALUE_GENERIC_TYPE oldValue, final
VALUE_GENERIC_TYPE newValue) { synchronized(sync) { return map.replace(key, oldValue, newValue); } }

#ifdef JDK_PRIMITIVE_FUNCTION
  @Override
  public VALUE_GENERIC_TYPE COMPUTE_IF_ABSENT_JDK(final KEY_GENERIC_TYPE key, final
JDK_PRIMITIVE_FUNCTION KEY_SUPER_GENERIC VALUE_EXTENDS_GENERIC mappingFunction) {
synchronized(sync) { return map.COMPUTE_IF_ABSENT_JDK(key, mappingFunction); } }
#endif

#if KEYS_PRIMITIVE && VALUES_PRIMITIVE
  @Override
  public VALUE_GENERIC_TYPE COMPUTE_IF_ABSENT_NULLABLE(final KEY_GENERIC_TYPE key,
final JDK_KEY_TO_GENERIC_FUNCTION<? extends VALUE_GENERIC_CLASS> mappingFunction) {
synchronized(sync) { return map.COMPUTE_IF_ABSENT_NULLABLE(key, mappingFunction); } }
#endif
```

```
#if KEYS_PRIMITIVE || VALUES_PRIMITIVE
  @Override
  public VALUE_GENERIC_TYPE COMPUTE_IF_ABSENT_PARTIAL(final KEY_GENERIC_TYPE key, final
FUNCTION KEY_SUPER_GENERIC VALUE_EXTENDS_GENERIC mappingFunction) { synchronized(sync) {
return map.COMPUTE_IF_ABSENT_PARTIAL(key, mappingFunction); } }
#endif

  @Override
  public VALUE_GENERIC_TYPE COMPUTE_IF_PRESENT(final KEY_GENERIC_TYPE key, final
java.util.function.BiFunction<? super KEY_GENERIC_CLASS, ? super VALUE_GENERIC_CLASS, ? extends
VALUE_GENERIC_CLASS> remappingFunction) {
   synchronized (sync) { return map.COMPUTE_IF_PRESENT(key, remappingFunction); }
  }

  @Override
  public VALUE_GENERIC_TYPE COMPUTE(final KEY_GENERIC_TYPE key, final
java.util.function.BiFunction<? super KEY_GENERIC_CLASS, ? super VALUE_GENERIC_CLASS, ? extends
VALUE_GENERIC_CLASS> remappingFunction) {
   synchronized (sync) { return map.COMPUTE(key, remappingFunction); }
  }

  @Override
  public VALUE_GENERIC_TYPE MERGE(final KEY_GENERIC_TYPE key, final VALUE_GENERIC_TYPE
value, final java.util.function.BiFunction<? super VALUE_GENERIC_CLASS, ? super
VALUE_GENERIC_CLASS, ? extends VALUE_GENERIC_CLASS> remappingFunction) {
   synchronized (sync) { return map.MERGE(key, value, remappingFunction); }
  }

#if KEYS_PRIMITIVE || VALUES_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public VALUE_GENERIC_CLASS getOrDefault(final Object key, final VALUE_GENERIC_CLASS
defaultValue) { synchronized (sync) { return map.getOrDefault(key, defaultValue); } }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public boolean remove(final Object key, final Object value) { synchronized (sync) { return map.remove(key, value);
} }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public VALUE_GENERIC_CLASS replace(final KEY_GENERIC_CLASS key, final VALUE_GENERIC_CLASS
```

```
value) { synchronized (sync) { return map.replace(key, value); } }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public boolean replace(final KEY_GENERIC_CLASS key, final VALUE_GENERIC_CLASS oldValue, final
VALUE_GENERIC_CLASS newValue) { synchronized (sync) { return map.replace(key, oldValue, newValue); } }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public VALUE_GENERIC_CLASS putIfAbsent(final KEY_GENERIC_CLASS key, final
VALUE_GENERIC_CLASS value) { synchronized (sync) { return map.putIfAbsent(key, value); } }

#if KEYS_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
#endif
  @Override
  public VALUE_GENERIC_CLASS computeIfAbsent(final KEY_GENERIC_CLASS key, final
java.util.function.Function<? super KEY_GENERIC_CLASS, ? extends VALUE_GENERIC_CLASS>
mappingFunction) { synchronized (sync) { return map.computeIfAbsent(key, mappingFunction); } }

#if KEYS_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
#endif
  @Override
  public VALUE_GENERIC_CLASS computeIfPresent(final KEY_GENERIC_CLASS key, final
java.util.function.BiFunction<? super KEY_GENERIC_CLASS, ? super VALUE_GENERIC_CLASS, ? extends
VALUE_GENERIC_CLASS> remappingFunction) { synchronized (sync) { return map.computeIfPresent(key,
remappingFunction); } }

#if KEYS_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
#endif
  @Override
  public VALUE_GENERIC_CLASS compute(final KEY_GENERIC_CLASS key, final
java.util.function.BiFunction<? super KEY_GENERIC_CLASS, ? super VALUE_GENERIC_CLASS, ? extends
VALUE_GENERIC_CLASS> remappingFunction) { synchronized (sync) { return map.compute(key,
remappingFunction); } }
```

```
 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 public VALUE_GENERIC_CLASS merge(final KEY_GENERIC_CLASS key, final VALUE_GENERIC_CLASS
value, final java.util.function.BiFunction<? super VALUE_GENERIC_CLASS, ? super
VALUE_GENERIC_CLASS, ? extends VALUE_GENERIC_CLASS> remappingFunction) { synchronized (sync)
{ return map.merge(key, value, remappingFunction); } }
#endif

 }

 /** Returns a synchronized type-specific map backed by the given type-specific map.
  *
  * @param m the map to be wrapped in a synchronized map.
  * @return a synchronized view of the specified map.
  * @see java.util.Collections#synchronizedMap(Map)
  */
 public static KEY_VALUE_GENERIC MAP KEY_VALUE_GENERIC synchronize(final MAP
KEY_VALUE_GENERIC m) { return new SynchronizedMap KEY_VALUE_GENERIC_DIAMOND(m); }

 /** Returns a synchronized type-specific map backed by the given type-specific map, using an assigned object to
synchronize.
  *
  * @param m the map to be wrapped in a synchronized map.
  * @param sync an object that will be used to synchronize the access to the map.
  * @return a synchronized view of the specified map.
  * @see java.util.Collections#synchronizedMap(Map)
  */

 public static KEY_VALUE_GENERIC MAP KEY_VALUE_GENERIC synchronize(final MAP
KEY_VALUE_GENERIC m, final Object sync) { return new SynchronizedMap
KEY_VALUE_GENERIC_DIAMOND(m, sync); }



 /** An unmodifiable wrapper class for maps. */

 public static class UnmodifiableMap KEY_VALUE_GENERIC extends FUNCTIONS.UnmodifiableFunction
KEY_VALUE_GENERIC implements MAP KEY_VALUE_GENERIC, java.io.Serializable {

  private static final long serialVersionUID = -7046029254386353129L;

  protected final MAP KEY_VALUE_GENERIC map;

  protected transient ObjectSet<MAP.Entry KEY_VALUE_GENERIC> entries;
  protected transient SET KEY_GENERIC keys;
  protected transient VALUE_COLLECTION VALUE_GENERIC values;
```

```
  protected UnmodifiableMap(final MAP KEY_VALUE_GENERIC m) {
   super(m);
   this.map = m;
  }

  @Override
  public boolean containsValue(final VALUE_TYPE v) { return map.containsValue(v); }
#if VALUES_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public boolean containsValue(final Object ov) { return map.containsValue(ov); }
#endif

  @Override
  public void putAll(final Map<? extends KEY_GENERIC_CLASS, ? extends VALUE_GENERIC_CLASS> m) {
throw new UnsupportedOperationException(); }

  @Override
  public ObjectSet<MAP.Entry KEY_VALUE_GENERIC> ENTRYSET() { if (entries == null) entries =
ObjectSets.unmodifiable(map.ENTRYSET()); return entries; }

#if KEYS_PRIMITIVE || VALUES_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
#else
  /** {@inheritDoc} */
#endif
  @Override
  @SuppressWarnings({ "unchecked", "rawtypes" })
  public ObjectSet<Map.Entry<KEY_GENERIC_CLASS, VALUE_GENERIC_CLASS>> entrySet() { return
(ObjectSet)ENTRYSET(); }

  @Override
  public SET KEY_GENERIC keySet() { if (keys == null) keys = SETS.unmodifiable(map.keySet()); return keys; }

  @Override
  public VALUE_COLLECTION VALUE_GENERIC values() { if (values == null) return
VALUE_COLLECTIONS.unmodifiable(map.values()); return values; }

  @Override
  public boolean isEmpty() { return map.isEmpty(); }

  @Override
  public int hashCode() { return map.hashCode(); }
```

```
  @Override
  public boolean equals(final Object o) {
   if (o == this) return true;
   return map.equals(o);
  }

  // Defaultable methods

  @Override
  public VALUE_GENERIC_TYPE getOrDefault(final KEY_TYPE key, final VALUE_GENERIC_TYPE
defaultValue) { return map.getOrDefault(key, defaultValue); }

  @Override
  public void forEach(final java.util.function.BiConsumer<? super KEY_GENERIC_CLASS, ? super
VALUE_GENERIC_CLASS> action) { map.forEach(action); }

  @Override
  public void replaceAll(final java.util.function.BiFunction<? super KEY_GENERIC_CLASS, ? super
VALUE_GENERIC_CLASS, ? extends VALUE_GENERIC_CLASS> function) { throw new
UnsupportedOperationException(); }

  @Override
  public VALUE_GENERIC_TYPE putIfAbsent(final KEY_GENERIC_TYPE key, final VALUE_GENERIC_TYPE
value) { throw new UnsupportedOperationException(); }

  @Override
  public boolean remove(final KEY_TYPE key, final VALUE_TYPE value) { throw new
UnsupportedOperationException(); }

  @Override
  public VALUE_GENERIC_TYPE replace(final KEY_GENERIC_TYPE key, final VALUE_GENERIC_TYPE
value) { throw new UnsupportedOperationException(); }

  @Override
  public boolean replace(final KEY_GENERIC_TYPE key, final VALUE_GENERIC_TYPE oldValue, final
VALUE_GENERIC_TYPE newValue) { throw new UnsupportedOperationException(); }

#ifdef JDK_PRIMITIVE_FUNCTION
  @Override
  public VALUE_GENERIC_TYPE COMPUTE_IF_ABSENT_JDK(final KEY_GENERIC_TYPE key, final
JDK_PRIMITIVE_FUNCTION KEY_SUPER_GENERIC VALUE_EXTENDS_GENERIC mappingFunction) {
throw new UnsupportedOperationException(); }
#endif

#if KEYS_PRIMITIVE && VALUES_PRIMITIVE
  @Override
  public VALUE_GENERIC_TYPE COMPUTE_IF_ABSENT_NULLABLE(final KEY_GENERIC_TYPE key,
```

```
final JDK_KEY_TO_GENERIC_FUNCTION<? extends VALUE_GENERIC_CLASS> mappingFunction) { throw new UnsupportedOperationException(); }
#endif

#if KEYS_PRIMITIVE || VALUES_PRIMITIVE
  @Override
  public VALUE_GENERIC_TYPE COMPUTE_IF_ABSENT_PARTIAL(final KEY_GENERIC_TYPE key, final FUNCTION KEY_SUPER_GENERIC VALUE_EXTENDS_GENERIC mappingFunction) { throw new UnsupportedOperationException(); }
#endif

  @Override
  public VALUE_GENERIC_TYPE COMPUTE_IF_PRESENT(final KEY_GENERIC_TYPE key, final java.util.function.BiFunction<? super KEY_GENERIC_CLASS, ? super VALUE_GENERIC_CLASS, ? extends VALUE_GENERIC_CLASS> remappingFunction) { throw new UnsupportedOperationException(); }

  @Override
  public VALUE_GENERIC_TYPE COMPUTE(final KEY_GENERIC_TYPE key, final java.util.function.BiFunction<? super KEY_GENERIC_CLASS, ? super VALUE_GENERIC_CLASS, ? extends VALUE_GENERIC_CLASS> remappingFunction) { throw new UnsupportedOperationException(); }

  @Override
  public VALUE_GENERIC_TYPE MERGE(final KEY_GENERIC_TYPE key, final VALUE_GENERIC_TYPE value, final java.util.function.BiFunction<? super VALUE_GENERIC_CLASS, ? super VALUE_GENERIC_CLASS, ? extends VALUE_GENERIC_CLASS> remappingFunction) { throw new UnsupportedOperationException(); }

#if KEYS_PRIMITIVE || VALUES_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public VALUE_GENERIC_CLASS getOrDefault(final Object key, final VALUE_GENERIC_CLASS defaultValue) { return map.getOrDefault(key, defaultValue); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public boolean remove(final Object key, final Object value) { throw new UnsupportedOperationException(); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public VALUE_GENERIC_CLASS replace(final KEY_GENERIC_CLASS key, final VALUE_GENERIC_CLASS value) { throw new UnsupportedOperationException(); }
```

```
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public boolean replace(final KEY_GENERIC_CLASS key, final VALUE_GENERIC_CLASS oldValue, final
VALUE_GENERIC_CLASS newValue) { throw new UnsupportedOperationException(); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public VALUE_GENERIC_CLASS putIfAbsent(final KEY_GENERIC_CLASS key, final
VALUE_GENERIC_CLASS value) { throw new UnsupportedOperationException(); }

#if KEYS_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
#endif
  @Override
  public VALUE_GENERIC_CLASS computeIfAbsent(final KEY_GENERIC_CLASS key, final
java.util.function.Function<? super KEY_GENERIC_CLASS, ? extends VALUE_GENERIC_CLASS>
mappingFunction) { throw new UnsupportedOperationException(); }

#if KEYS_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
#endif
  @Override
  public VALUE_GENERIC_CLASS computeIfPresent(final KEY_GENERIC_CLASS key, final
java.util.function.BiFunction<? super KEY_GENERIC_CLASS, ? super VALUE_GENERIC_CLASS, ? extends
VALUE_GENERIC_CLASS> remappingFunction) { throw new UnsupportedOperationException(); }

#if KEYS_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
#endif
  @Override
  public VALUE_GENERIC_CLASS compute(final KEY_GENERIC_CLASS key, final
java.util.function.BiFunction<? super KEY_GENERIC_CLASS, ? super VALUE_GENERIC_CLASS, ? extends
VALUE_GENERIC_CLASS> remappingFunction) { throw new UnsupportedOperationException(); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
```

```
  public VALUE_GENERIC_CLASS merge(final KEY_GENERIC_CLASS key, final VALUE_GENERIC_CLASS value, final java.util.function.BiFunction<? super VALUE_GENERIC_CLASS, ? super VALUE_GENERIC_CLASS, ? extends VALUE_GENERIC_CLASS> remappingFunction) { throw new UnsupportedOperationException(); }
#endif

 }

 /** Returns an unmodifiable type-specific map backed by the given type-specific map.
  *
  * @param m the map to be wrapped in an unmodifiable map.
  * @return an unmodifiable view of the specified map.
  * @see java.util.Collections#unmodifiableMap(Map)
  */
 public static KEY_VALUE_GENERIC MAP KEY_VALUE_GENERIC unmodifiable(final MAP KEY_VALUE_GENERIC m) { return new UnmodifiableMap KEY_VALUE_GENERIC_DIAMOND(m); }

}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/Maps.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import it.unimi.dsi.fastutil.Stack;

/** A type-specific {@link Stack}; provides some additional methods that use polymorphism to avoid (un)boxing.
*/
```

```
public interface STACK KEY_GENERIC extends Stack<KEY_GENERIC_CLASS> {

/** Pushes the given object on the stack.
 * @param k the object to push on the stack.
 * @see Stack#push(Object)
 */
void push(KEY_TYPE k);

/** Pops the top off the stack.
 *
 * @return the top of the stack.
 * @see Stack#pop()
 */
KEY_TYPE POP();

/** Peeks at the top of the stack (optional operation).
 * @return the top of the stack.
 * @see Stack#top()
 */
KEY_TYPE TOP();

/** Peeks at an element on the stack (optional operation).
 * @param i an index from the stop of the stack (0 represents the top).
 * @return the <code>i</code>-th element on the stack.
 * @see Stack#peek(int)
 */
KEY_TYPE PEEK(int i);

/** {@inheritDoc}
 * <p>This default implementation delegates to the corresponding type-specific method.
 * @deprecated Please use the corresponding type-specific method instead. */
@Deprecated
@Override
default void push(KEY_GENERIC_CLASS o) {
 push(o.KEY_VALUE());
}

/** {@inheritDoc}
 * <p>This default implementation delegates to the corresponding type-specific method.
 * @deprecated Please use the corresponding type-specific method instead. */
@Deprecated
@Override
default KEY_GENERIC_CLASS pop() {
 return KEY_CLASS.valueOf(POP());
}

/** {@inheritDoc}
 * <p>This default implementation delegates to the corresponding type-specific method.
```

```
 * @deprecated Please use the corresponding type-specific method instead. */
@Deprecated
@Override
default KEY_GENERIC_CLASS top() {
 return KEY_CLASS.valueOf(TOP());
}

/** {@inheritDoc}
 * <p>This default implementation delegates to the corresponding type-specific method.
 * @deprecated Please use the corresponding type-specific method instead. */
@Deprecated
@Override
default KEY_GENERIC_CLASS peek(final int i) {
 return KEY_CLASS.valueOf(PEEK(i));
}
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/Stack.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import it.unimi.dsi.fastutil.BigList;
import java.util.Collection;
import java.util.List;
import java.util.Random;

/** A class providing static methods and objects that do useful things with type-specific big lists.
*
```

```
* @see java.util.Collections
* @see it.unimi.dsi.fastutil.BigList
*/

public final class BIG_LISTS {

 private BIG_LISTS() {}

 /** Shuffles the specified big list using the specified pseudorandom number generator.
  *
  * @param l the big list to be shuffled.
  * @param random a pseudorandom number generator.
  * @return {@code l}.
  */
 public static KEY_GENERIC BIG_LIST KEY_GENERIC shuffle(final BIG_LIST KEY_GENERIC l, final
Random random) {
  for(long i = l.size64(); i-- != 0;) {
   final long p = (random.nextLong() & 0x7FFFFFFFFFFFFFFFL) % (i + 1);
   final KEY_GENERIC_TYPE t = l.GET_KEY(i);
   l.set(i, l.GET_KEY(p));
   l.set(p, t);
  }
  return l;
 }


 /** An immutable class representing an empty type-specific big list.
  *
  * <p>This class may be useful to implement your own in case you subclass
  * a type-specific list.
  */

 public static class EmptyBigList KEY_GENERIC extends COLLECTIONS.EmptyCollection KEY_GENERIC
implements BIG_LIST KEY_GENERIC, java.io.Serializable, Cloneable {

  private static final long serialVersionUID = -7046029254386353129L;

  protected EmptyBigList() {}

  @Override
  public KEY_GENERIC_TYPE GET_KEY(long i) { throw new IndexOutOfBoundsException(); }

  @Override
  public boolean REMOVE(KEY_TYPE k) { throw new UnsupportedOperationException(); }

  @Override
  public KEY_GENERIC_TYPE REMOVE_KEY(long i) { throw new UnsupportedOperationException(); }
```

```
  @Override
  public void add(final long index, final KEY_GENERIC_TYPE k) { throw new UnsupportedOperationException();
}

  @Override
  public KEY_GENERIC_TYPE set(final long index, final KEY_GENERIC_TYPE k) { throw new
UnsupportedOperationException(); }

  @Override
  public long indexOf(KEY_TYPE k) { return -1; }

  @Override
  public long lastIndexOf(KEY_TYPE k) { return -1; }

  @Override
  public boolean addAll(long i, Collection<? extends KEY_GENERIC_CLASS> c) { throw new
UnsupportedOperationException(); }

#if KEYS_PRIMITIVE
  @Override
  public boolean addAll(COLLECTION c) { throw new UnsupportedOperationException(); }

  @Override
  public boolean addAll(BIG_LIST c) { throw new UnsupportedOperationException(); }

  @Override
  public boolean addAll(long i, COLLECTION c) { throw new UnsupportedOperationException(); }

  @Override
  public boolean addAll(long i, BIG_LIST c) { throw new UnsupportedOperationException(); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public void add(final long index, final KEY_GENERIC_CLASS k) { throw new UnsupportedOperationException();
}

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public boolean add(final KEY_GENERIC_CLASS k) { throw new UnsupportedOperationException(); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
```

```
  public KEY_GENERIC_CLASS get(long i) { throw new IndexOutOfBoundsException(); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public KEY_GENERIC_CLASS set(final long index, final KEY_GENERIC_CLASS k) { throw new
UnsupportedOperationException(); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public KEY_GENERIC_CLASS remove(long k) { throw new UnsupportedOperationException(); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public long indexOf(Object k) { return -1; }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public long lastIndexOf(Object k) { return -1; }
#endif

  @Override
  SUPPRESS_WARNINGS_KEY_UNCHECKED
  public KEY_BIG_LIST_ITERATOR KEY_GENERIC listIterator() { return
BIG_LIST_ITERATORS.EMPTY_BIG_LIST_ITERATOR; }

  @Override
  SUPPRESS_WARNINGS_KEY_UNCHECKED
  public KEY_BIG_LIST_ITERATOR KEY_GENERIC iterator() { return
BIG_LIST_ITERATORS.EMPTY_BIG_LIST_ITERATOR; }

  @Override
  SUPPRESS_WARNINGS_KEY_UNCHECKED
  public KEY_BIG_LIST_ITERATOR KEY_GENERIC listIterator(long i) { if (i == 0) return
BIG_LIST_ITERATORS.EMPTY_BIG_LIST_ITERATOR; throw new
IndexOutOfBoundsException(String.valueOf(i)); }

  @Override
  public BIG_LIST KEY_GENERIC subList(long from, long to) { if (from == 0 && to == 0) return this; throw new
IndexOutOfBoundsException(); }
```

```
  @Override
  public void getElements(long from, KEY_TYPE[][] a, long offset, long length) {
BIG_ARRAYS.ensureOffsetLength(a, offset, length); if (from != 0) throw new IndexOutOfBoundsException(); }

  @Override
  public void removeElements(long from, long to) { throw new UnsupportedOperationException(); }

  @Override
  public void addElements(long index, final KEY_GENERIC_TYPE a[][], long offset, long length) { throw new
UnsupportedOperationException(); }

  @Override
  public void addElements(long index, final KEY_GENERIC_TYPE a[][]) { throw new
UnsupportedOperationException(); }

  @Override
  public void size(long s)  { throw new UnsupportedOperationException(); }

  @Override
  public long size64() { return 0; }

#if ! KEY_CLASS_Reference
  @Override
  public int compareTo(final BigList<? extends KEY_GENERIC_CLASS> o) {
   if (o == this) return 0;
   return ((BigList<?>)o).isEmpty() ? 0 : -1;
  }
#endif
  @Override
  public Object clone() { return EMPTY_BIG_LIST; }

  @Override
  public int hashCode() { return 1; }

  @Override
  @SuppressWarnings("rawtypes")
  public boolean equals(Object o) { return o instanceof BigList && ((BigList)o).isEmpty(); }

  @Override
  public String toString() { return "[]"; }

  private Object readResolve() { return EMPTY_BIG_LIST; }
 }

 /** An empty big list (immutable). It is serializable and cloneable.
  */
 SUPPRESS_WARNINGS_KEY_RAWTYPES
 public static final EmptyBigList EMPTY_BIG_LIST = new EmptyBigList();
```

```
#if KEYS_REFERENCE
 /** Returns an empty big list (immutable). It is serializable and cloneable.
  *
  * <p>This method provides a typesafe access to {@link #EMPTY_BIG_LIST}.
  * @return an empty big list (immutable).
  */
 @SuppressWarnings("unchecked")
 public static KEY_GENERIC BIG_LIST KEY_GENERIC emptyList() {
  return EMPTY_BIG_LIST;
 }
#endif

 /** An immutable class representing a type-specific singleton big list.
  *
  * <p>This class may be useful to implement your own in case you subclass
  * a type-specific big list.
  */

 public static class Singleton KEY_GENERIC extends ABSTRACT_BIG_LIST KEY_GENERIC implements
java.io.Serializable, Cloneable {

  private static final long serialVersionUID = -7046029254386353129L;

  private final KEY_GENERIC_TYPE element;

  protected Singleton(final KEY_GENERIC_TYPE element) { this.element = element; }

  @Override
  public KEY_GENERIC_TYPE GET_KEY(final long i) { if (i == 0) return element; throw new
IndexOutOfBoundsException(); }

  @Override
  public boolean REMOVE(KEY_TYPE k) { throw new UnsupportedOperationException(); }

  @Override
  public KEY_GENERIC_TYPE REMOVE_KEY(final long i) { throw new UnsupportedOperationException(); }

  @Override
  public boolean contains(final KEY_TYPE k) { return KEY_EQUALS(k, element); }

  /* Slightly optimized w.r.t. the one in ABSTRACT_SET. */

  @Override
  public KEY_TYPE[] TO_KEY_ARRAY() {
   KEY_TYPE a[] = new KEY_TYPE[1];
   a[0] = element;
   return a;
```

```
  }

  @Override
  public KEY_BIG_LIST_ITERATOR KEY_GENERIC listIterator() { return
BIG_LIST_ITERATORS.singleton(element); }

  @Override
  public KEY_BIG_LIST_ITERATOR KEY_GENERIC listIterator(long i) {
   if (i > 1 || i < 0) throw new  IndexOutOfBoundsException();
   KEY_BIG_LIST_ITERATOR KEY_GENERIC l = listIterator();
   if (i == 1) l.NEXT_KEY();
   return l;
  }

  @Override
  SUPPRESS_WARNINGS_KEY_UNCHECKED
  public BIG_LIST KEY_GENERIC subList(final long from, final long to) {
   ensureIndex(from);
   ensureIndex(to);
   if (from > to) throw new IndexOutOfBoundsException("Start index (" + from + ") is greater than end index (" + to +
")");

   if (from != 0 || to != 1) return EMPTY_BIG_LIST;
   return this;
  }

  @Override
  public boolean addAll(long i, Collection<? extends KEY_GENERIC_CLASS> c) { throw new
UnsupportedOperationException(); }

  @Override
  public boolean addAll(final Collection<? extends KEY_GENERIC_CLASS> c) { throw new
UnsupportedOperationException(); }

  @Override
  public boolean removeAll(final Collection<?> c) { throw new UnsupportedOperationException(); }

  @Override
  public boolean retainAll(final Collection<?> c) { throw new UnsupportedOperationException(); }

#if KEYS_PRIMITIVE

  @Override
  public boolean addAll(BIG_LIST c) { throw new UnsupportedOperationException(); }

  @Override
  public boolean addAll(long i, BIG_LIST c) { throw new UnsupportedOperationException(); }
```

```
 @Override
 public boolean addAll(long i, COLLECTION c) { throw new UnsupportedOperationException(); }

 @Override
 public boolean addAll(final COLLECTION c) { throw new UnsupportedOperationException(); }

 @Override
 public boolean removeAll(final COLLECTION c) { throw new UnsupportedOperationException(); }

 @Override
 public boolean retainAll(final COLLECTION c) { throw new UnsupportedOperationException(); }

#endif

 @Override
 public void clear() { throw new UnsupportedOperationException(); }

 @Override
 public long size64() { return 1; }

 @Override
 public Object clone() { return this; }
 }

 /** Returns a type-specific immutable big list containing only the specified element. The returned big list is serializable and cloneable.
  *
  * @param element the only element of the returned big list.
  * @return a type-specific immutable big list containing just {@code element}.
  */

 public static KEY_GENERIC BIG_LIST KEY_GENERIC singleton(final KEY_GENERIC_TYPE element) { return new Singleton KEY_GENERIC_DIAMOND(element); }

#if KEYS_PRIMITIVE

 /** Returns a type-specific immutable big list containing only the specified element. The returned big list is serializable and cloneable.
  *
  * @param element the only element of the returned big list.
  * @return a type-specific immutable big list containing just {@code element}.
  */

 public static KEY_GENERIC BIG_LIST KEY_GENERIC singleton(final Object element) { return new Singleton KEY_GENERIC_DIAMOND(KEY_OBJ2TYPE(element)); }

#endif
```

```
 /** A synchronized wrapper class for big lists. */

 public static class SynchronizedBigList KEY_GENERIC extends COLLECTIONS.SynchronizedCollection
KEY_GENERIC implements BIG_LIST KEY_GENERIC, java.io.Serializable {

  private static final long serialVersionUID = -7046029254386353129L;

  protected final BIG_LIST KEY_GENERIC list; // Due to the large number of methods that are not in
COLLECTION, this is worth caching.

  protected SynchronizedBigList(final BIG_LIST KEY_GENERIC l, final Object sync) {
   super(l, sync);
   this.list = l;
  }

  protected SynchronizedBigList(final BIG_LIST KEY_GENERIC l) {
   super(l);
   this.list = l;
  }

  @Override
  public KEY_GENERIC_TYPE GET_KEY(final long i) { synchronized(sync) { return list.GET_KEY(i); } }

  @Override
  public KEY_GENERIC_TYPE set(final long i, final KEY_GENERIC_TYPE k) { synchronized(sync) { return
list.set(i, k); } }

  @Override
  public void add(final long i, final KEY_GENERIC_TYPE k) { synchronized(sync) { list.add(i, k); } }

  @Override
  public KEY_GENERIC_TYPE REMOVE_KEY(final long i) { synchronized(sync) { return list.REMOVE_KEY(i);
} }

  @Override
  public long indexOf(final KEY_TYPE k) { synchronized(sync) { return list.indexOf(k); } }

  @Override
  public long lastIndexOf(final KEY_TYPE k) { synchronized(sync) { return list.lastIndexOf(k); } }

  @Override
  public boolean addAll(final long index, final Collection<? extends KEY_GENERIC_CLASS> c) {
synchronized(sync) { return list.addAll(index, c); } }

  @Override
  public void getElements(final long from, final KEY_TYPE a[][], final long offset, final long length) {
synchronized(sync) { list.getElements(from, a, offset, length); } }
```

```
  @Override
  public void removeElements(final long from, final long to) { synchronized(sync) { list.removeElements(from, to); }
}

  @Override
  public void addElements(long index, final KEY_GENERIC_TYPE a[][], long offset, long length) {
synchronized(sync) { list.addElements(index, a, offset, length); } }

  @Override
  public void addElements(long index, final KEY_GENERIC_TYPE a[][]) { synchronized(sync) {
list.addElements(index, a); } }

  /* {@inheritDoc}
   * @deprecated Use {@link #size64()} instead.
   */
  @Deprecated
  @Override
  public void size(final long size) { synchronized(sync) { list.size(size); } }

  @Override
  public long size64() { synchronized(sync) { return list.size64(); } }

  @Override
  public KEY_BIG_LIST_ITERATOR KEY_GENERIC iterator() { return list.listIterator(); }

  @Override
  public KEY_BIG_LIST_ITERATOR KEY_GENERIC listIterator() { return list.listIterator(); }

  @Override
  public KEY_BIG_LIST_ITERATOR KEY_GENERIC listIterator(final long i) { return list.listIterator(i); }

  @Override
  public BIG_LIST KEY_GENERIC subList(final long from, final long to) { synchronized(sync) { return
synchronize(list.subList(from, to), sync); } }

  @Override
  public boolean equals(final Object o) { if (o == this) return true; synchronized(sync) { return list.equals(o); } }

  @Override
  public int hashCode() { synchronized(sync) { return list.hashCode(); } }

#if ! KEY_CLASS_Reference
  @Override
  public int compareTo(final BigList<? extends KEY_GENERIC_CLASS> o) { synchronized(sync) { return
list.compareTo(o); } }
#endif
```

```
#if KEYS_PRIMITIVE
  @Override
  public boolean addAll(final long index, final COLLECTION c) { synchronized(sync) { return list.addAll(index, c);
} }

  @Override
  public boolean addAll(final long index, BIG_LIST l) { synchronized(sync) { return list.addAll(index, l); } }

  @Override
  public boolean addAll(BIG_LIST l) { synchronized(sync) { return list.addAll(l); } }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public void add(final long i, KEY_GENERIC_CLASS k) { synchronized(sync) { list.add(i, k); } }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public KEY_GENERIC_CLASS get(final long i) { synchronized(sync) { return list.get(i); } }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public KEY_GENERIC_CLASS set(final long index, KEY_GENERIC_CLASS k) { synchronized(sync) { return
list.set(index, k); } }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public KEY_GENERIC_CLASS remove(final long i) { synchronized(sync) { return list.remove(i); } }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public long indexOf(final Object o) { synchronized(sync) { return list.indexOf(o); } }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public long lastIndexOf(final Object o) { synchronized(sync) { return list.lastIndexOf(o); } }
#endif
```

```
}

/** Returns a synchronized type-specific big list backed by the given type-specific big list.
 *
 * @param l the big list to be wrapped in a synchronized big list.
 * @return a synchronized view of the specified big list.
 * @see java.util.Collections#synchronizedList(List)
 */
public static KEY_GENERIC BIG_LIST KEY_GENERIC synchronize(final BIG_LIST KEY_GENERIC l) {
return new SynchronizedBigList KEY_GENERIC_DIAMOND(l); }

/** Returns a synchronized type-specific big list backed by the given type-specific big list, using an assigned object
to synchronize.
 *
 * @param l the big list to be wrapped in a synchronized big list.
 * @param sync an object that will be used to synchronize the access to the big list.
 * @return a synchronized view of the specified big list.
 * @see java.util.Collections#synchronizedList(List)
 */

public static KEY_GENERIC BIG_LIST KEY_GENERIC synchronize(final BIG_LIST KEY_GENERIC l, final
Object sync) { return new SynchronizedBigList KEY_GENERIC_DIAMOND(l, sync); }



/** An unmodifiable wrapper class for big lists. */

public static class UnmodifiableBigList KEY_GENERIC extends COLLECTIONS.UnmodifiableCollection
KEY_GENERIC implements BIG_LIST KEY_GENERIC, java.io.Serializable {

 private static final long serialVersionUID = -7046029254386353129L;

 protected final BIG_LIST KEY_GENERIC list; // Due to the large number of methods that are not in
COLLECTION, this is worth caching.

 protected UnmodifiableBigList(final BIG_LIST KEY_GENERIC l) {
  super(l);
  this.list = l;
 }

 @Override
 public KEY_GENERIC_TYPE GET_KEY(final long i) { return list.GET_KEY(i); }

 @Override
 public KEY_GENERIC_TYPE set(final long i, final KEY_GENERIC_TYPE k) { throw new
UnsupportedOperationException(); }
```

```
  @Override
  public void add(final long i, final KEY_GENERIC_TYPE k) { throw new UnsupportedOperationException(); }

  @Override
  public KEY_GENERIC_TYPE REMOVE_KEY(final long i) { throw new UnsupportedOperationException(); }

  @Override
  public long indexOf(final KEY_TYPE k) { return list.indexOf(k); }

  @Override
  public long lastIndexOf(final KEY_TYPE k) { return list.lastIndexOf(k); }

  @Override
  public boolean addAll(final long index, final Collection<? extends KEY_GENERIC_CLASS> c) { throw new
UnsupportedOperationException(); }

  @Override
  public void getElements(final long from, final KEY_TYPE a[][], final long offset, final long length) {
list.getElements(from, a, offset, length); }

  @Override
  public void removeElements(final long from, final long to) { throw new UnsupportedOperationException(); }

  @Override
  public void addElements(long index, final KEY_GENERIC_TYPE a[][], long offset, long length) { throw new
UnsupportedOperationException(); }

  @Override
  public void addElements(long index, final KEY_GENERIC_TYPE a[][]) { throw new
UnsupportedOperationException(); }

  /* {@inheritDoc}
   * @deprecated Use {@link #size64()} instead.
   */
  @Deprecated
  @Override
  public void size(final long size) { list.size(size); }

  @Override
  public long size64() { return list.size64(); }

  @Override
  public KEY_BIG_LIST_ITERATOR KEY_GENERIC iterator() { return listIterator(); }

  @Override
  public KEY_BIG_LIST_ITERATOR KEY_GENERIC listIterator() { return
BIG_LIST_ITERATORS.unmodifiable(list.listIterator()); }
```

```
  @Override
  public KEY_BIG_LIST_ITERATOR KEY_GENERIC listIterator(final long i) { return
BIG_LIST_ITERATORS.unmodifiable(list.listIterator(i)); }

  @Override
  public BIG_LIST KEY_GENERIC subList(final long from, final long to) { return unmodifiable(list.subList(from,
to)); }

  @Override
  public boolean equals(final Object o) { if (o == this) return true; return list.equals(o); }

  @Override
  public int hashCode() { return list.hashCode(); }

#if ! KEY_CLASS_Reference
  @Override
  public int compareTo(final BigList<? extends KEY_GENERIC_CLASS> o) { return list.compareTo(o); }
#endif

#if KEYS_PRIMITIVE
  @Override
  public boolean addAll(final long index, final COLLECTION c) { throw new UnsupportedOperationException(); }

  @Override
  public boolean addAll(final BIG_LIST l) { throw new UnsupportedOperationException(); }

  @Override
  public boolean addAll(final long index, final BIG_LIST l) { throw new UnsupportedOperationException(); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public KEY_GENERIC_CLASS get(final long i) { return list.get(i); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public void add(final long i, KEY_GENERIC_CLASS k) { throw new UnsupportedOperationException(); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public KEY_GENERIC_CLASS set(final long index, KEY_GENERIC_CLASS k) { throw new
UnsupportedOperationException(); }
```

```
 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 public KEY_GENERIC_CLASS remove(final long i) { throw new UnsupportedOperationException(); }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 public long indexOf(final Object o) { return list.indexOf(o); }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 public long lastIndexOf(final Object o) { return list.lastIndexOf(o); }
#endif
 }


 /** Returns an unmodifiable type-specific big list backed by the given type-specific big list.
  *
  * @param l the big list to be wrapped in an unmodifiable big list.
  * @return an unmodifiable view of the specified big list.
  * @see java.util.Collections#unmodifiableList(List)
  */
 public static KEY_GENERIC BIG_LIST KEY_GENERIC unmodifiable(final BIG_LIST KEY_GENERIC l) {
return new UnmodifiableBigList KEY_GENERIC_DIAMOND(l); }

 /** A class exposing a list as a big list. */

 public static class ListBigList KEY_GENERIC extends ABSTRACT_BIG_LIST KEY_GENERIC implements
java.io.Serializable {

  private static final long serialVersionUID = -7046029254386353129L;

  private final LIST KEY_GENERIC list;

  protected ListBigList(final LIST KEY_GENERIC list) {
   this.list = list;
  }

  private int intIndex(long index) {
   if (index >= Integer.MAX_VALUE) throw new IndexOutOfBoundsException("This big list is restricted to 32-bit
indices");
   return (int)index;
  }
```

```
  @Override
  public long size64() { return list.size(); }

  @Override
  public void size(final long size) { list.size(intIndex(size)); }

  @Override
  public KEY_BIG_LIST_ITERATOR KEY_GENERIC iterator() { return
BIG_LIST_ITERATORS.asBigListIterator(list.iterator()); }

  @Override
  public KEY_BIG_LIST_ITERATOR KEY_GENERIC listIterator() { return
BIG_LIST_ITERATORS.asBigListIterator(list.listIterator()); }

  @Override
  public KEY_BIG_LIST_ITERATOR KEY_GENERIC listIterator(final long index) { return
BIG_LIST_ITERATORS.asBigListIterator(list.listIterator(intIndex(index))); }

  @Override
  public boolean addAll(final long index, final Collection<? extends KEY_GENERIC_CLASS> c) { return
list.addAll(intIndex(index), c); }

  @Override
  public BIG_LIST KEY_GENERIC subList(long from, long to) { return new ListBigList
KEY_GENERIC_DIAMOND(list.subList(intIndex(from), intIndex(to))); }

  @Override
  public boolean contains(final KEY_TYPE key) { return list.contains(key); }

  @Override
  public KEY_TYPE[] TO_KEY_ARRAY() { return list.TO_KEY_ARRAY(); }

  @Override
  public void removeElements(final long from, final long to) { list.removeElements(intIndex(from), intIndex(to)); }

#if KEYS_PRIMITIVE
  /* {@inheritDoc}
   * @deprecated Please use {@code toArray()} instead&amp;mdash;this method is redundant and will be removed in the
future.
   */
  @Deprecated
  @Override
  public KEY_TYPE[] TO_KEY_ARRAY(KEY_TYPE[] a) { return list.toArray(a); }

  @Override
  public boolean addAll(long index, COLLECTION KEY_GENERIC c) { return list.addAll(intIndex(index), c); }
```

```
  @Override
  public boolean addAll(COLLECTION KEY_GENERIC c) { return list.addAll(c); }

  @Override
  public boolean addAll(long index, BIG_LIST KEY_GENERIC c) { return list.addAll(intIndex(index), c); }

  @Override
  public boolean addAll(BIG_LIST KEY_GENERIC c) { return list.addAll(c); }

  @Override
  public boolean containsAll(COLLECTION KEY_GENERIC c) { return list.containsAll(c); }

  @Override
  public boolean removeAll(COLLECTION KEY_GENERIC c) { return list.removeAll(c); }

  @Override
  public boolean retainAll(COLLECTION KEY_GENERIC c) { return list.retainAll(c); }
#endif
  @Override
  public void add(long index, KEY_GENERIC_TYPE key) { list.add(intIndex(index), key); }

  @Override
  public boolean add(KEY_GENERIC_TYPE key) { return list.add(key); }

  @Override
  public KEY_GENERIC_TYPE GET_KEY(long index) { return list.GET_KEY(intIndex(index)); }

  @Override
  public long indexOf(KEY_TYPE k) { return list.indexOf(k); }

  @Override
  public long lastIndexOf(KEY_TYPE k) { return list.lastIndexOf(k); }

  @Override
  public KEY_GENERIC_TYPE REMOVE_KEY(long index) { return list.REMOVE_KEY(intIndex(index)); }

  @Override
  public KEY_GENERIC_TYPE set(long index, KEY_GENERIC_TYPE k) { return list.set(intIndex(index), k); }

  @Override
  public boolean isEmpty() { return list.isEmpty(); }

  @Override
  public <T> T[] toArray(T[] a) { return list.toArray(a); }

  @Override
  public boolean containsAll(Collection<?> c) { return list.containsAll(c); }
```

```
 @Override
 public boolean addAll(Collection<? extends KEY_GENERIC_CLASS> c) { return list.addAll(c); }

 @Override
 public boolean removeAll(Collection<?> c) { return list.removeAll(c); }

 @Override
 public boolean retainAll(Collection<?> c) { return list.retainAll(c); }

 @Override
 public void clear() { list.clear(); }

 @Override
 public int hashCode() { return list.hashCode(); }
}

/** Returns a big list backed by the specified list.
*
* @param list a list.
* @return a big list backed by the specified list.
*/
public static KEY_GENERIC BIG_LIST KEY_GENERIC asBigList(final LIST KEY_GENERIC list) { return new
ListBigList KEY_GENERIC_DIAMOND(list); }




#ifdef TEST

 private static KEY_TYPE genKey() {
#if KEY_CLASS_Byte || KEY_CLASS_Short || KEY_CLASS_Character
  return (KEY_TYPE)(r.nextInt());
#elif KEYS_PRIMITIVE
  return r.NEXT_KEY();
#elif KEY_CLASS_Object
  return Integer.toBinaryString(r.nextInt());
#else
  return new java.io.Serializable() {};
#endif
 }


 private static void testLists(KEY_TYPE k, BIG_LIST m, BIG_LIST t, int level) {
  int n = 100;
  int c;

  long ms;
  boolean mThrowsIllegal, tThrowsIllegal, mThrowsNoElement, tThrowsNoElement, mThrowsIndex, tThrowsIndex,
```

```
mThrowsUnsupp, tThrowsUnsupp;
  boolean rt = false, rm = false;
  Object Rt = null, Rm = null;

  if (level == 0) return;

  /* Now we check that m and t are equal. */
  if (!m.equals(t) || ! t.equals(m)) System.err.println("m: " + m + " t: " + t);

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) at start");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) at start");

  /* Now we check that m actually holds that data. */
  for(java.util.Iterator i=t.iterator(); i.hasNext();) {
   ensure(m.contains(i.next()), "Error (" + level + ", " + seed + "): m and t differ on an entry after insertion (iterating on
t)");
  }

  /* Now we check that m actually holds that data, but iterating on m. */
  for(java.util.Iterator i=m.listIterator(); i.hasNext();) {
   ensure(t.contains(i.next()), "Error (" + level + ", " + seed + "): m and t differ on an entry after insertion (iterating on
m)");
  }

  /* Now we check that inquiries about random data give the same answer in m and t. For
     m we use the polymorphic method. */

  for(int i=0; i<n;  i++) {
   KEY_TYPE T = genKey();

   mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal =
mThrowsUnsupp = tThrowsUnsupp = false;

   try {
    m.contains(T);
   }
   catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
   catch (IllegalArgumentException e) { mThrowsIllegal = true; }
   catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }

   try {
    t.contains(KEY2OBJ(T));
   }
   catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
   catch (IllegalArgumentException e) { tThrowsIllegal = true; }
   catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }

   ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): contains() divergence in
```

```
java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ") " + m);
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): contains() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
  ensure(mThrowsIndex == tThrowsIndex, "Error (" + level + ", " + seed + "): contains() divergence in
IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
  if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex) ensure(m.contains(KEY2OBJ(T)) ==
t.contains(KEY2OBJ(T)), "Error (" + level + ", " + seed + "): divergence in keys between t and m (polymorphic
method) " + m);
 }

 /* Again, we check that inquiries about random data give the same answer in m and t, but
    for m we use the standard method. */

 for(int i=0; i<n;  i++) {
  KEY_TYPE T = genKey();

  mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal =
mThrowsUnsupp = tThrowsUnsupp = false;

  try {
   m.contains(KEY2OBJ(T));
  }
  catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }
  catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }

  try {
   t.contains(KEY2OBJ(T));
  }
  catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
  catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): contains() divergence in
java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ") " + m);
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): contains() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
  ensure(mThrowsIndex == tThrowsIndex, "Error (" + level + ", " + seed + "): contains() divergence in
IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
  ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + level + ", " + seed + "): contains() divergence in
UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);
  if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex && !mThrowsUnsupp)
ensure(m.contains(KEY2OBJ(T)) ==  t.contains(KEY2OBJ(T)), "Error (" + level + ", " + seed + "): divergence
between t and m (standard method) " + m);
 }
```

```
  /* Now we add and remove random data in m and t, checking that the result is the same. */

  for(int i=0; i<20*n;  i++) {
   KEY_TYPE T = genKey();

   mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal =
mThrowsUnsupp = tThrowsUnsupp = false;

   try {
    rm = m.add(KEY2OBJ(T));
   }
   catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
   catch (IllegalArgumentException e) { mThrowsIllegal = true; }
   catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }
   catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }

   try {
    rt = t.add(KEY2OBJ(T));
   }
   catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
   catch (IllegalArgumentException e) { tThrowsIllegal = true; }
   catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
   catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }

   ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): add() divergence in
java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ") " + m);
   ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): add() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
   ensure(mThrowsIndex == tThrowsIndex, "Error (" + level + ", " + seed + "): add() divergence in
IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
   ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + level + ", " + seed + "): add() divergence in
UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);
   if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex && !mThrowsUnsupp) ensure(rm == rt, "Error
(" + level + ", " + seed + "): divergence in add() between t and m " + m);

   T = genKey();

   mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal =
mThrowsUnsupp = tThrowsUnsupp = false;

   try {
    rm = m.remove(KEY2OBJ(T));
   }
   catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
   catch (IllegalArgumentException e) { mThrowsIllegal = true; }
   catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }
   catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }
```

```
  try {
   rt = t.remove(KEY2OBJ(T));
  }
  catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
  catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }

  if (! KEY_EQUALS(T, k) && mThrowsUnsupp && ! tThrowsUnsupp) mThrowsUnsupp = true; // Stupid bug in
Collections.singleton()

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): remove() divergence in
java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ") " + m);
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): remove() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
  ensure(mThrowsIndex == tThrowsIndex, "Error (" + level + ", " + seed + "): remove() divergence in
IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
  ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + level + ", " + seed + "): remove() divergence in
UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);
  if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex && !mThrowsUnsupp) ensure(rm == rt, "Error
(" + level + ", " + seed + "): divergence in remove() between t and m " + m);
 }

 ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after removal " + m);
 ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after removal " + m);

 /* Now we add and remove random data in m and t at specific positions, checking that the result is the same. */

 for(int i=0; i<20*n;  i++) {
  KEY_TYPE T = genKey();

  mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal =
mThrowsUnsupp = tThrowsUnsupp = false;

  int pos = r.nextInt(2);

  try {
   m.add(pos, KEY2OBJ(T));
  }
  catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }
  catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }

  try {
   t.add(pos, KEY2OBJ(T));
  }
  catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
```

```
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
  catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): add() at " + pos + " divergence in java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ") " + m);
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): add() at " + pos + " divergence in IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
  ensure(mThrowsIndex == tThrowsIndex, "Error (" + level + ", " + seed + "): add() at " + pos + " divergence in IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
  ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + level + ", " + seed + "): add() at " + pos + " divergence in UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);

  T = genKey();

  mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal = mThrowsUnsupp = tThrowsUnsupp = false;

  pos = r.nextInt(2);

  try {
   Rm = m.remove(pos);
  }
  catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }
  catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }

  try {
   Rt = t.remove(pos);
  }
  catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
  catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }


  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): remove() at " + pos + " divergence in java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ") " + m);
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): remove() at " + pos + " divergence in IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
  ensure(mThrowsIndex == tThrowsIndex, "Error (" + level + ", " + seed + "): remove() at " + pos + " divergence in IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
  ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + level + ", " + seed + "): remove() at " + pos + " divergence in UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);
  if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex && !mThrowsUnsupp) ensure(Rm == Rt || Rm
```

```
!= null && Rm.equals(Rt), "Error (" + level + ", " + seed + "): divergence in remove() at " + pos + " between t and m " + m);
  }

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after removal " + m);
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after removal " + m);

  /* Now we add and remove random collections in m and t, checking that the result is the same. */

  for(int i=0; i<20*n;  i++) {
   KEY_TYPE T = genKey();

   mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal = mThrowsUnsupp = tThrowsUnsupp = false;

   try {
    rm = m.addAll(java.util.Collections.singleton(KEY2OBJ(T)));
   }
   catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
   catch (IllegalArgumentException e) { mThrowsIllegal = true; }
   catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }
   catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }

   try {
    rt = t.addAll(java.util.Collections.singleton(KEY2OBJ(T)));
   }
   catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
   catch (IllegalArgumentException e) { tThrowsIllegal = true; }
   catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
   catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }

   ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): addAll() divergence in java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ") " + m);
   ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): addAll() divergence in IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
   ensure(mThrowsIndex == tThrowsIndex, "Error (" + level + ", " + seed + "): addAll() divergence in IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
   ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + level + ", " + seed + "): addAll() divergence in UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);
   if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex && !mThrowsUnsupp) ensure(rm == rt, "Error (" + level + ", " + seed + "): divergence in addAll() between t and m " + m);

   T = genKey();

   mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal = mThrowsUnsupp = tThrowsUnsupp = false;

   try {
```

```
   rm = m.removeAll(java.util.Collections.singleton(KEY2OBJ(T)));
   }
   catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
   catch (IllegalArgumentException e) { mThrowsIllegal = true; }
   catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }
   catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }

   try {
    rt = t.removeAll(java.util.Collections.singleton(KEY2OBJ(T)));
   }
   catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
   catch (IllegalArgumentException e) { tThrowsIllegal = true; }
   catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
   catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }


   ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): removeAll() divergence in
java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ") " + m);
   ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): removeAll() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
   ensure(mThrowsIndex == tThrowsIndex, "Error (" + level + ", " + seed + "): removeAll() divergence in
IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
   ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + level + ", " + seed + "): removeAll() divergence in
UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);
   if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex && !mThrowsUnsupp) ensure(rm == rt, "Error
(" + level + ", " + seed + "): divergence in removeAll() between t and m " + m);
   }

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after set removal " + m);
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after set removal " + m);

  /* Now we add random collections at specific positions in m and t, checking that the result is the same. */

  for(int i=0; i<20*n;  i++) {
   KEY_TYPE T = genKey();

   mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal =
mThrowsUnsupp = tThrowsUnsupp = false;

   int pos = r.nextInt(2);

   try {
    rm = m.addAll(pos, java.util.Collections.singleton(KEY2OBJ(T)));
   }
   catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
   catch (IllegalArgumentException e) { mThrowsIllegal = true; }
   catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }
   catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }
```

```
  try {
   rt = t.addAll(pos, java.util.Collections.singleton(KEY2OBJ(T)));
  }
  catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
  catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): addAll() at " + pos + "
divergence in java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement
+ ") " + m);
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): addAll() at " + pos + " divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
  ensure(mThrowsIndex == tThrowsIndex, "Error (" + level + ", " + seed + "): addAll() at " + pos + " divergence in
IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
  ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + level + ", " + seed + "): addAll() at " + pos + " divergence
in UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);
  if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex && !mThrowsUnsupp) ensure(rm == rt, "Error
(" + level + ", " + seed + "): divergence in addAll() at " + pos + " between t and m " + m);

 }

 ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after set removal " + m);
 ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after set removal " + m);

 /* Now we check that m actually holds the same data. */

 for(java.util.Iterator i=t.iterator(); i.hasNext();) {
  ensure(m.contains(i.next()), "Error (" + level + ", " + seed + "): m and t differ on an entry after removal (iterating on
t)");
 }

 /* Now we check that m actually holds that data, but iterating on m. */

 for(java.util.Iterator i=m.listIterator(); i.hasNext();) {
  ensure(t.contains(i.next()), "Error (" + level + ", " + seed + "): m and t differ on an entry after removal (iterating on
m)");
 }

 if (m instanceof Singleton) {
  ensure(m.equals(((Singleton)m).clone()), "Error (" + level + ", " + seed + "): m does not equal m.clone()");
  ensure(((Singleton)m).clone().equals(m), "Error (" + level + ", " + seed + "): m.clone() does not equal m");
 }

 int h = m.hashCode();

 /* Now we save and read m. */
```

```
  BIG_LIST m2 = null;

  try {
   java.io.File ff = new java.io.File("it.unimi.dsi.fastutil.test");
   java.io.OutputStream os = new java.io.FileOutputStream(ff);
   java.io.ObjectOutputStream oos = new java.io.ObjectOutputStream(os);

   oos.writeObject(m);
   oos.close();

   java.io.InputStream is = new java.io.FileInputStream(ff);
   java.io.ObjectInputStream ois = new java.io.ObjectInputStream(is);

   m2 = (BIG_LIST)ois.readObject();
   ois.close();
   ff.delete();
  }
  catch(Exception e) {
   e.printStackTrace();
   System.exit(1);
  }

#if ! KEY_CLASS_Reference

  ensure(m2.hashCode() == h, "Error (" + level + ", " + seed + "): hashCode() changed after save/read");

  /* Now we check that m2 actually holds that data. */

  ensure(m2.equals(t), "Error (" + level + ", " + seed + "): ! m2.equals(t) after save/read");
  ensure(t.equals(m2), "Error (" + level + ", " + seed + "): ! t.equals(m2) after save/read");
#endif

  if (! m.isEmpty()) {
   long start = (r.nextLong() & 0x7FFFFFFFFFFFFFFFL) % m.size64();
   long end = start + (r.nextLong() & 0x7FFFFFFFFFFFFFFFL) % (m.size64() - start);
   //System.err.println("Checking subList from " + start + " to " + end + " (level=" + (level+1) + ")...");
   testLists(k, m.subList(start, end), t.subList(start, end), level - 1);

   ensure(m.equals(t), "Error (" + level + ", " + seed + m + t + "): ! m.equals(t) after subList");
   ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after subList");

  }

  return;
  }

  private static void test() {
```

```
 KEY_TYPE k = genKey();
 BIG_LIST m = new Singleton(k);
 BIG_LIST u = BIG_LISTS.unmodifiable(BIG_LISTS.asBigList(LISTS.singleton(KEY2OBJ(k))));
 testLists(k, m, u, 3);
 System.out.println("Test OK");
}

private static long seed = System.currentTimeMillis();
private static java.util.Random r = new java.util.Random(seed);

private static java.text.NumberFormat format = new java.text.DecimalFormat("#,###.00");
private static java.text.FieldPosition fp = new java.text.FieldPosition(0);

private static String format(double d) {
 StringBuffer s = new StringBuffer();
 return format.format(d, s, fp).toString();
}

private static void fatal(String msg) {
 System.out.println(msg);
 System.exit(1);
}

private static void ensure(boolean cond, String msg) {
 if (cond) return;
 fatal(msg);
}

/** This method expects as first argument a lower-cased type (e.g., "int"),
 * and as second optional argument a seed. */

public static void main(String arg[]) throws Exception {
 if (arg.length > 1) r = new java.util.Random(seed = Long.parseLong(arg[1]));

 try {
  test();
 } catch(Throwable e) {
  e.printStackTrace(System.err);
  System.err.println("seed: " + seed);
 }
}

#endif

}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-

a775574/drv/BigLists.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import java.util.List;

#if ! KEY_CLASS_Reference

/** A type-specific {@link List}; provides some additional methods that use polymorphism to avoid (un)boxing.
*
* <p>Note that this type-specific interface extends {@link Comparable}: it is expected that implementing
* classes perform a lexicographical comparison using the standard operator "less then" for primitive types,
* and the usual {@link Comparable#compareTo(Object) compareTo()} method for objects.
*
* <p>Additionally, this interface strengthens {@link #listIterator()},
* {@link #listIterator(int)} and {@link #subList(int,int)}.
*
* <p>Besides polymorphic methods, this interfaces specifies methods to copy into an array or remove contiguous
* sublists. Although the abstract implementation of this interface provides simple, one-by-one implementations
* of these methods, it is expected that concrete implementation override them with optimized versions.
*
* @see List
*/

public interface LIST KEY_GENERIC extends List<KEY_GENERIC_CLASS>, Comparable<List<? extends
KEY_GENERIC_CLASS>>, COLLECTION KEY_GENERIC {
#else

/** A type-specific {@link List}; provides some additional methods that use polymorphism to avoid (un)boxing.
*
```

```
* <p>Additionally, this interface strengthens {@link #iterator()}, {@link #listIterator()},
* {@link #listIterator(int)} and {@link #subList(int,int)}. The former had been already
* strengthened upstream, but unfortunately {@link List} re-specifies it.
*
* <p>Besides polymorphic methods, this interfaces specifies methods to copy into an array or remove contiguous
* sublists. Although the abstract implementation of this interface provides simple, one-by-one implementations
* of these methods, it is expected that concrete implementation override them with optimized versions.
*
* @see List
*/

public interface LIST KEY_GENERIC extends List<KEY_GENERIC_CLASS>, COLLECTION KEY_GENERIC
{
#endif

 /** Returns a type-specific iterator on the elements of this list.
  *
  * <p>Note that this specification strengthens the one given in {@link List#iterator()}.
  * It would not be normally necessary, but {@link java.lang.Iterable#iterator()} is bizarrily re-specified
  * in {@link List}.
  *
  * @return an iterator on the elements of this list.
  */
 @Override
 KEY_LIST_ITERATOR KEY_GENERIC iterator();

 /** Returns a type-specific list iterator on the list.
  *
  * @see List#listIterator()
  */
 @Override
 KEY_LIST_ITERATOR KEY_GENERIC listIterator();

 /** Returns a type-specific list iterator on the list starting at a given index.
  *
  * @see List#listIterator(int)
  */
 @Override
 KEY_LIST_ITERATOR KEY_GENERIC listIterator(int index);

 /** Returns a type-specific view of the portion of this list from the index {@code from}, inclusive, to the index
 {@code to}, exclusive.
  *
  * <p>Note that this specification strengthens the one given in {@link List#subList(int,int)}.
  *
  * @see List#subList(int,int)
  */
 @Override
```

```
LIST KEY_GENERIC subList(int from, int to);

/** Sets the size of this list.
 *
 * <p>If the specified size is smaller than the current size, the last elements are
 * discarded. Otherwise, they are filled with 0/{@code null}/{@code false}.
 *
 * @param size the new size.
 */

void size(int size);

/** Copies (hopefully quickly) elements of this type-specific list into the given array.
 *
 * @param from the start index (inclusive).
 * @param a the destination array.
 * @param offset the offset into the destination array where to store the first element copied.
 * @param length the number of elements to be copied.
 */
void getElements(int from, KEY_TYPE a[], int offset, int length);

/** Removes (hopefully quickly) elements of this type-specific list.
 *
 * @param from the start index (inclusive).
 * @param to the end index (exclusive).
 */
void removeElements(int from, int to);

/** Add (hopefully quickly) elements to this type-specific list.
 *
 * @param index the index at which to add elements.
 * @param a the array containing the elements.
 */
void addElements(int index, KEY_GENERIC_TYPE a[]);

/** Add (hopefully quickly) elements to this type-specific list.
 *
 * @param index the index at which to add elements.
 * @param a the array containing the elements.
 * @param offset the offset of the first element to add.
 * @param length the number of elements to add.
 */
void addElements(int index, KEY_GENERIC_TYPE a[], int offset, int length);

#if KEYS_PRIMITIVE

/** Appends the specified element to the end of this list (optional operation).
 * @see List#add(Object)
```

```
 */
@Override
boolean add(KEY_TYPE key);

/** Inserts the specified element at the specified position in this list (optional operation).
 * @see List#add(int,Object)
 */
void add(int index, KEY_TYPE key);

/** {@inheritDoc}
 * @deprecated Please use the corresponding type-specific method instead. */
@Deprecated
@Override
default void add(int index, KEY_CLASS key) {
 add(index, KEY_CLASS2TYPE(key));
}

/** Inserts all of the elements in the specified type-specific collection into this type-specific list at the specified position (optional operation).
 * @see List#addAll(int,java.util.Collection)
 */
boolean addAll(int index, COLLECTION c);

/** Inserts all of the elements in the specified type-specific list into this type-specific list at the specified position (optional operation).
 * @see List#add(int,Object)
 */
boolean addAll(int index, LIST c);

/** Appends all of the elements in the specified type-specific list to the end of this type-specific list (optional operation).
 * @see List#add(int,Object)
 */
boolean addAll(LIST c);

/** Replaces the element at the specified position in this list with the specified element (optional operation).
 * @see List#set(int,Object)
 */
KEY_TYPE set(int index, KEY_TYPE k);

/** Returns the element at the specified position in this list.
 * @see List#get(int)
 */
KEY_TYPE GET_KEY(int index);

/** Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
 * @see List#indexOf(Object)
```

```
 */
int indexOf(KEY_TYPE k);

/** Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.
 * @see List#lastIndexOf(Object)
 */
int lastIndexOf(KEY_TYPE k);

/** {@inheritDoc}
 * @deprecated Please use the corresponding type-specific method instead.
 */
@Deprecated
@Override
default boolean contains(final Object key) {
 return COLLECTION.super.contains(key);
}

/** {@inheritDoc}
 * @deprecated Please use the corresponding type-specific method instead. */
@Deprecated
@Override
default KEY_GENERIC_CLASS get(int index) {
 return KEY2OBJ(GET_KEY(index));
}

/** {@inheritDoc}
 * @deprecated Please use the corresponding type-specific method instead. */
@Deprecated
@Override
default int indexOf(Object o) {
 return indexOf(KEY_OBJ2TYPE(o));
}

/** {@inheritDoc}
 * @deprecated Please use the corresponding type-specific method instead. */
@Deprecated
@Override
default int lastIndexOf(Object o) {
 return lastIndexOf(KEY_OBJ2TYPE(o));
}

/** {@inheritDoc}
 * <p>This method specification is a workaround for
 * <a href="http://bugs.java.com/bugdatabase/view_bug.do?bug_id=JDK-8177440">bug 8177440</a>.
 * @deprecated Please use the corresponding type-specific method instead. */
@Deprecated
@Override
```

```
 default boolean add(KEY_CLASS k) {
  return add(KEY_CLASS2TYPE(k));
 }

 /** Removes the element at the specified position in this list (optional operation).
  * @see List#remove(int)
  */
 KEY_TYPE REMOVE_KEY(int index);

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead.
  */
 @Deprecated
 @Override
 default boolean remove(final Object key) {
  return COLLECTION.super.remove(key);
 }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 default KEY_GENERIC_CLASS remove(int index) {
  return KEY2OBJ(REMOVE_KEY(index));
 }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 default KEY_GENERIC_CLASS set(int index, KEY_CLASS k) {
  return KEY2OBJ(set(index, KEY_CLASS2TYPE(k)));
 }

#endif

}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/List.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
```

```
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import it.unimi.dsi.fastutil.BidirectionalIterator;
#if KEYS_PRIMITIVE
import it.unimi.dsi.fastutil.objects.ObjectBidirectionalIterator;
#endif

/** A type-specific bidirectional iterator; provides an additional method to avoid (un)boxing,
* and the possibility to skip elements backwards.
*
* @see BidirectionalIterator
*/

#if KEYS_PRIMITIVE
public interface KEY_BIDI_ITERATOR KEY_GENERIC extends KEY_ITERATOR KEY_GENERIC,
ObjectBidirectionalIterator<KEY_GENERIC_CLASS> {
#else
public interface KEY_BIDI_ITERATOR KEY_GENERIC extends KEY_ITERATOR KEY_GENERIC,
BidirectionalIterator<KEY_GENERIC_CLASS> {
#endif

#if KEYS_PRIMITIVE

 /**
  * Returns the previous element as a primitive type.
  *
  * @return the previous element in the iteration.
  * @see java.util.ListIterator#previous()
  */

 KEY_TYPE PREV_KEY();

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
```

```
 default KEY_GENERIC_CLASS previous() { return KEY_CLASS.valueOf(PREV_KEY()); }

#endif

 /** Moves back for the given number of elements.
  *
  * <p>The effect of this call is exactly the same as that of
  * calling {@link #previous()} for {@code n} times (possibly stopping
  * if {@link #hasPrevious()} becomes false).
  *
  * @param n the number of elements to skip back.
  * @return the number of elements actually skipped.
  * @see #previous()
  */
#if KEYS_PRIMITIVE
 @Override
#endif
 default int back(final int n) {
  int i = n;
  while(i-- != 0 && hasPrevious()) PREV_KEY();
  return n - i - 1;
 }

 /** {@inheritDoc} */
 @Override
 default int skip(final int n) {
  return KEY_ITERATOR.super.skip(n);
 }
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/BidirectionalIterator.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
```

```
* limitations under the License.
*/


package PACKAGE;

/** A type-specific {@link Iterable} that further strengthens the specification of {@link Iterable#iterator()}.
*/
public interface KEY_BIDI_ITERABLE KEY_GENERIC extends KEY_ITERABLE KEY_GENERIC {

 /** Returns a type-specific {@link it.unimi.dsi.fastutil.BidirectionalIterator}.
  *
  * @return a type-specific bidirectional iterator.
  */
 @Override
 KEY_BIDI_ITERATOR KEY_GENERIC iterator();
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/BidirectionalIterable.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import java.util.SortedSet;
import java.util.NoSuchElementException;
#if KEYS_REFERENCE
import java.util.Comparator;
#endif
```

```
/** A class providing static methods and objects that do useful things with type-specific sorted sets.
*
* @see java.util.Collections
*/

public final class SORTED_SETS {

 private SORTED_SETS() {}

 /** An immutable class representing the empty sorted set and implementing a type-specific set interface.
  *
  * <p>This class may be useful to implement your own in case you subclass
  * a type-specific sorted set.
  */

 public static class EmptySet KEY_GENERIC extends SETS.EmptySet KEY_GENERIC implements
SORTED_SET KEY_GENERIC, java.io.Serializable, Cloneable {
  private static final long serialVersionUID = -7046029254386353129L;

  protected EmptySet() {}

  @Override
  SUPPRESS_WARNINGS_KEY_UNCHECKED
  public KEY_BIDI_ITERATOR KEY_GENERIC iterator(KEY_GENERIC_TYPE from) { return
ITERATORS.EMPTY_ITERATOR; }

  @Override
  SUPPRESS_WARNINGS_KEY_UNCHECKED
  public SORTED_SET KEY_GENERIC subSet(KEY_GENERIC_TYPE from, KEY_GENERIC_TYPE to) { return
EMPTY_SET; }

  @Override
  SUPPRESS_WARNINGS_KEY_UNCHECKED
  public SORTED_SET KEY_GENERIC headSet(KEY_GENERIC_TYPE from) { return EMPTY_SET; }

  @Override
  SUPPRESS_WARNINGS_KEY_UNCHECKED
  public SORTED_SET KEY_GENERIC tailSet(KEY_GENERIC_TYPE to) { return EMPTY_SET; }

  @Override
  public KEY_GENERIC_TYPE FIRST() { throw new NoSuchElementException(); }

  @Override
  public KEY_GENERIC_TYPE LAST() { throw new NoSuchElementException(); }

  @Override
  public KEY_COMPARATOR KEY_SUPER_GENERIC comparator() { return null; }
```

```
#if KEYS_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public SORTED_SET KEY_GENERIC subSet(KEY_GENERIC_CLASS from, KEY_GENERIC_CLASS to) {
return EMPTY_SET; }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public SORTED_SET KEY_GENERIC headSet(KEY_GENERIC_CLASS from) { return EMPTY_SET; }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public SORTED_SET KEY_GENERIC tailSet(KEY_GENERIC_CLASS to) { return EMPTY_SET; }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public KEY_GENERIC_CLASS first() { throw new NoSuchElementException(); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public KEY_GENERIC_CLASS last() { throw new NoSuchElementException(); }
#endif

  @Override
  public Object clone() { return EMPTY_SET; }

  private Object readResolve() { return EMPTY_SET; }
  }


 /** An empty sorted set (immutable). It is serializable and cloneable.
  *
  */
 SUPPRESS_WARNINGS_KEY_RAWTYPES
 public static final EmptySet EMPTY_SET = new EmptySet();

#if KEYS_REFERENCE
```

```
 /** Returns an empty sorted set (immutable). It is serializable and cloneable.
  *
  * <p>This method provides a typesafe access to {@link #EMPTY_SET}.
  * @return an empty sorted set (immutable).
  */
 @SuppressWarnings("unchecked")
 public static KEY_GENERIC SET KEY_GENERIC emptySet() {
  return EMPTY_SET;
 }
#endif

 /** A class representing a singleton sorted set.
  *
  * <p>This class may be useful to implement your own in case you subclass
  * a type-specific sorted set.
  */

 public static class Singleton KEY_GENERIC extends SETS.Singleton KEY_GENERIC implements SORTED_SET
KEY_GENERIC, java.io.Serializable, Cloneable {

  private static final long serialVersionUID = -7046029254386353129L;

  final KEY_COMPARATOR KEY_SUPER_GENERIC comparator;

  protected Singleton(final KEY_GENERIC_TYPE element, final KEY_COMPARATOR KEY_SUPER_GENERIC
comparator) {
   super(element);
   this.comparator = comparator;
  }

  private Singleton(final KEY_GENERIC_TYPE element) {
   this(element, null);
  }

  SUPPRESS_WARNINGS_KEY_UNCHECKED
  final int compare(final KEY_GENERIC_TYPE k1, final KEY_GENERIC_TYPE k2) {
   return comparator == null ? KEY_CMP(k1, k2) : comparator.compare(k1, k2);
  }

  @Override
  public KEY_BIDI_ITERATOR KEY_GENERIC iterator(KEY_GENERIC_TYPE from) {
   KEY_BIDI_ITERATOR KEY_GENERIC i = iterator();
   if (compare(element, from) <= 0) i.NEXT_KEY();
   return i;
  }

  @Override
  public KEY_COMPARATOR KEY_SUPER_GENERIC comparator() { return comparator; }
```

```
  @Override
  SUPPRESS_WARNINGS_KEY_UNCHECKED
  public SORTED_SET KEY_GENERIC subSet(final KEY_GENERIC_TYPE from, final KEY_GENERIC_TYPE
to) { if (compare(from, element) <= 0 && compare(element, to) < 0) return this; return EMPTY_SET; }

  @Override
  SUPPRESS_WARNINGS_KEY_UNCHECKED
  public SORTED_SET KEY_GENERIC headSet(final KEY_GENERIC_TYPE to) { if (compare(element, to) < 0)
return this; return EMPTY_SET; }

  @Override
  SUPPRESS_WARNINGS_KEY_UNCHECKED
  public SORTED_SET KEY_GENERIC tailSet(final KEY_GENERIC_TYPE from) { if (compare(from, element)
<= 0) return this; return EMPTY_SET; }

  @Override
  public KEY_GENERIC_TYPE FIRST() { return element; }

  @Override
  public KEY_GENERIC_TYPE LAST() { return element; }

#if KEYS_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public SORTED_SET KEY_GENERIC subSet(final KEY_CLASS from, final KEY_CLASS to) { return
subSet(KEY_CLASS2TYPE(from), KEY_CLASS2TYPE(to)); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public SORTED_SET KEY_GENERIC headSet(final KEY_CLASS to) { return headSet(KEY_CLASS2TYPE(to));
}

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public SORTED_SET KEY_GENERIC tailSet(final KEY_CLASS from) { return
tailSet(KEY_CLASS2TYPE(from)); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
```

```
 public KEY_CLASS first() { return KEY2OBJ(element); }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 public KEY_CLASS last() { return KEY2OBJ(element); }
#endif
 }


 /** Returns a type-specific immutable sorted set containing only the specified element. The returned sorted set is serializable and cloneable.
 *
 * @param element the only element of the returned sorted set.
 * @return a type-specific immutable sorted set containing just {@code element}.
 */

 public static KEY_GENERIC SORTED_SET KEY_GENERIC singleton(final KEY_GENERIC_TYPE element) {
  return new Singleton KEY_GENERIC_DIAMOND(element);
 }

 /** Returns a type-specific immutable sorted set containing only the specified element, and using a specified comparator. The returned sorted set is serializable and cloneable.
 *
 * @param element the only element of the returned sorted set.
 * @param comparator the comparator to use in the returned sorted set.
 * @return a type-specific immutable sorted set containing just {@code element}.
 */

 public static KEY_GENERIC SORTED_SET KEY_GENERIC singleton(final KEY_GENERIC_TYPE element, final KEY_COMPARATOR KEY_SUPER_GENERIC comparator) {
  return new Singleton KEY_GENERIC_DIAMOND(element, comparator);
 }

#if KEYS_PRIMITIVE

 /** Returns a type-specific immutable sorted set containing only the specified element. The returned sorted set is serializable and cloneable.
 *
 * @param element the only element of the returned sorted set.
 * @return a type-specific immutable sorted set containing just {@code element}.
 */

 public static KEY_GENERIC SORTED_SET KEY_GENERIC singleton(final Object element) {
  return new Singleton(KEY_OBJ2TYPE(element));
 }
```

```
 /** Returns a type-specific immutable sorted set containing only the specified element, and using a specified comparator. The returned sorted set is serializable and cloneable.
  *
  * @param element the only element of the returned sorted set.
  * @param comparator the comparator to use in the returned sorted set.
  * @return a type-specific immutable sorted set containing just {@code element}.
  */

 public static KEY_GENERIC SORTED_SET KEY_GENERIC singleton(final Object element, final KEY_COMPARATOR KEY_SUPER_GENERIC comparator) {
  return new Singleton(KEY_OBJ2TYPE(element), comparator);
 }
#endif


 /** A synchronized wrapper class for sorted sets. */

 public static class SynchronizedSortedSet KEY_GENERIC extends SETS.SynchronizedSet KEY_GENERIC implements SORTED_SET KEY_GENERIC, java.io.Serializable {

  private static final long serialVersionUID = -7046029254386353129L;

  protected final SORTED_SET KEY_GENERIC sortedSet;

  protected SynchronizedSortedSet(final SORTED_SET KEY_GENERIC s, final Object sync) {
   super(s, sync);
   sortedSet = s;
  }

  protected SynchronizedSortedSet(final SORTED_SET KEY_GENERIC s) {
   super(s);
   sortedSet = s;
  }

  @Override
  public KEY_COMPARATOR KEY_SUPER_GENERIC comparator() { synchronized(sync) { return sortedSet.comparator(); } }

  @Override
  public SORTED_SET KEY_GENERIC subSet(final KEY_GENERIC_TYPE from, final KEY_GENERIC_TYPE to) { return new SynchronizedSortedSet KEY_GENERIC_DIAMOND(sortedSet.subSet(from, to), sync); }

  @Override
  public SORTED_SET KEY_GENERIC headSet(final KEY_GENERIC_TYPE to) { return new SynchronizedSortedSet KEY_GENERIC_DIAMOND(sortedSet.headSet(to), sync); }

  @Override
  public SORTED_SET KEY_GENERIC tailSet(final KEY_GENERIC_TYPE from) { return new
```

```
SynchronizedSortedSet KEY_GENERIC_DIAMOND(sortedSet.tailSet(from), sync); }

  @Override
  public KEY_BIDI_ITERATOR KEY_GENERIC iterator() { return sortedSet.iterator(); }

  @Override
  public KEY_BIDI_ITERATOR KEY_GENERIC iterator(final KEY_GENERIC_TYPE from) { return
sortedSet.iterator(from); }

  @Override
  public KEY_GENERIC_TYPE FIRST() {  synchronized(sync) { return sortedSet.FIRST(); } }

  @Override
  public KEY_GENERIC_TYPE LAST() {  synchronized(sync) { return sortedSet.LAST(); } }

#if KEYS_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public KEY_CLASS first() {  synchronized(sync) { return sortedSet.first(); } }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public KEY_CLASS last() {  synchronized(sync) { return sortedSet.last(); } }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public SORTED_SET KEY_GENERIC subSet(final KEY_CLASS from, final KEY_CLASS to) { return new
SynchronizedSortedSet(sortedSet.subSet(from, to), sync); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public SORTED_SET KEY_GENERIC headSet(final KEY_CLASS to) { return new
SynchronizedSortedSet(sortedSet.headSet(to), sync); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public SORTED_SET KEY_GENERIC tailSet(final KEY_CLASS from) { return new
SynchronizedSortedSet(sortedSet.tailSet(from), sync); }
```

```
#endif
 }


 /** Returns a synchronized type-specific sorted set backed by the given type-specific sorted set.
  *
  * @param s the sorted set to be wrapped in a synchronized sorted set.
  * @return a synchronized view of the specified sorted set.
  * @see java.util.Collections#synchronizedSortedSet(SortedSet)
  */
 public static KEY_GENERIC SORTED_SET KEY_GENERIC synchronize(final SORTED_SET KEY_GENERIC s) { return new SynchronizedSortedSet KEY_GENERIC_DIAMOND(s); }

 /** Returns a synchronized type-specific sorted set backed by the given type-specific sorted set, using an assigned object to synchronize.
  *
  * @param s the sorted set to be wrapped in a synchronized sorted set.
  * @param sync an object that will be used to synchronize the access to the sorted set.
  * @return a synchronized view of the specified sorted set.
  * @see java.util.Collections#synchronizedSortedSet(SortedSet)
  */

 public static KEY_GENERIC SORTED_SET KEY_GENERIC synchronize(final SORTED_SET KEY_GENERIC s, final Object sync) { return new SynchronizedSortedSet KEY_GENERIC_DIAMOND(s, sync); }




 /** An unmodifiable wrapper class for sorted sets. */

 public static class UnmodifiableSortedSet KEY_GENERIC extends SETS.UnmodifiableSet KEY_GENERIC implements SORTED_SET KEY_GENERIC, java.io.Serializable {

  private static final long serialVersionUID = -7046029254386353129L;

  protected final SORTED_SET KEY_GENERIC sortedSet;

  protected UnmodifiableSortedSet(final SORTED_SET KEY_GENERIC s) {
   super(s);
   sortedSet = s;
  }

  @Override
  public KEY_COMPARATOR KEY_SUPER_GENERIC comparator() { return sortedSet.comparator(); }

  @Override
  public SORTED_SET KEY_GENERIC subSet(final KEY_GENERIC_TYPE from, final KEY_GENERIC_TYPE
```

```
to) { return new UnmodifiableSortedSet KEY_GENERIC_DIAMOND(sortedSet.subSet(from, to)); }

  @Override
  public SORTED_SET KEY_GENERIC headSet(final KEY_GENERIC_TYPE to) { return new
UnmodifiableSortedSet KEY_GENERIC_DIAMOND(sortedSet.headSet(to)); }

  @Override
  public SORTED_SET KEY_GENERIC tailSet(final KEY_GENERIC_TYPE from) { return new
UnmodifiableSortedSet KEY_GENERIC_DIAMOND(sortedSet.tailSet(from)); }

  @Override
  public KEY_BIDI_ITERATOR KEY_GENERIC iterator() { return
ITERATORS.unmodifiable(sortedSet.iterator()); }

  @Override
  public KEY_BIDI_ITERATOR KEY_GENERIC iterator(final KEY_GENERIC_TYPE from) { return
ITERATORS.unmodifiable(sortedSet.iterator(from)); }

  @Override
  public KEY_GENERIC_TYPE FIRST() {  return sortedSet.FIRST(); }

  @Override
  public KEY_GENERIC_TYPE LAST() {  return sortedSet.LAST(); }

#if KEYS_PRIMITIVE
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public KEY_CLASS first() {  return sortedSet.first(); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public KEY_CLASS last() {  return sortedSet.last(); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public SORTED_SET KEY_GENERIC subSet(final KEY_GENERIC_CLASS from, final
KEY_GENERIC_CLASS to) { return new UnmodifiableSortedSet(sortedSet.subSet(from, to)); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
```

```
  public SORTED_SET KEY_GENERIC headSet(final KEY_GENERIC_CLASS to) { return new
UnmodifiableSortedSet(sortedSet.headSet(to)); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public SORTED_SET KEY_GENERIC tailSet(final KEY_GENERIC_CLASS from) { return new
UnmodifiableSortedSet(sortedSet.tailSet(from)); }
#endif
 }


 /** Returns an unmodifiable type-specific sorted set backed by the given type-specific sorted set.
  *
  * @param s the sorted set to be wrapped in an unmodifiable sorted set.
  * @return an unmodifiable view of the specified sorted set.
  * @see java.util.Collections#unmodifiableSortedSet(SortedSet)
  */
 public static KEY_GENERIC SORTED_SET KEY_GENERIC unmodifiable(final SORTED_SET
KEY_GENERIC s) { return new UnmodifiableSortedSet KEY_GENERIC_DIAMOND(s); }



#if defined(TEST) && ! KEY_CLASS_Reference

 private static KEY_TYPE genKey() {
#if KEY_CLASS_Byte || KEY_CLASS_Short || KEY_CLASS_Character
  return (KEY_TYPE)(r.nextInt());
#elif KEYS_PRIMITIVE
  return r.NEXT_KEY();
#elif KEY_CLASS_Object
  return Integer.toBinaryString(r.nextInt());
#endif
 }


 protected static void testSets(KEY_TYPE k, SORTED_SET m, SortedSet t, int level) {
  int n = 100;
  int c;

  long ms;
  boolean mThrowsIllegal, tThrowsIllegal, mThrowsNoElement, tThrowsNoElement, mThrowsIndex, tThrowsIndex,
mThrowsUnsupp, tThrowsUnsupp;
  boolean rt = false, rm = false;

  if (level == 0) return;
```

```
  /* Now we check that m and t are equal. */
  if (!m.equals(t) || ! t.equals(m)) System.err.println("m: " + m + " t: " + t);

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) at start");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) at start");

  /* Now we check that m actually holds that data. */
  for(java.util.Iterator i=t.iterator(); i.hasNext();) {
   ensure(m.contains(i.next()), "Error (" + level + ", " + seed + "): m and t differ on an entry after insertion (iterating on
t)");
  }

  /* Now we check that m actually holds that data, but iterating on m. */
  for(java.util.Iterator i=m.iterator(); i.hasNext();) {
   ensure(t.contains(i.next()), "Error (" + level + ", " + seed + "): m and t differ on an entry after insertion (iterating on
m)");
  }

  /* Now we check that inquiries about random data give the same answer in m and t. For
     m we use the polymorphic method. */

  for(int i=0; i<n;  i++) {
   KEY_TYPE T = genKey();

   mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal =
mThrowsUnsupp = tThrowsUnsupp = false;

   try {
    m.contains(T);
   }
   catch (NoSuchElementException e) { mThrowsNoElement = true; }
   catch (IllegalArgumentException e) { mThrowsIllegal = true; }
   catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }

   try {
    t.contains(KEY2OBJ(T));
   }
   catch (NoSuchElementException e) { tThrowsNoElement = true; }
   catch (IllegalArgumentException e) { tThrowsIllegal = true; }
   catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }

   ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): contains() divergence in
NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ") " + m);
   ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): contains() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
   ensure(mThrowsIndex == tThrowsIndex, "Error (" + level + ", " + seed + "): contains() divergence in
IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
   if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex) ensure(m.contains(KEY2OBJ(T)) ==
```

```
t.contains(KEY2OBJ(T)), "Error (" + level + ", " + seed + "): divergence in keys between t and m (polymorphic method) " + m);
  }

  /* Again, we check that inquiries about random data give the same answer in m and t, but
     for m we use the standard method. */

  for(int i=0; i<n;  i++) {
   KEY_TYPE T = genKey();

   mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal = mThrowsUnsupp = tThrowsUnsupp = false;

   try {
    m.contains(KEY2OBJ(T));
   }
   catch (NoSuchElementException e) { mThrowsNoElement = true; }
   catch (IllegalArgumentException e) { mThrowsIllegal = true; }
   catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }
   catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }

   try {
    t.contains(KEY2OBJ(T));
   }
   catch (NoSuchElementException e) { tThrowsNoElement = true; }
   catch (IllegalArgumentException e) { tThrowsIllegal = true; }
   catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
   catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }

   ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): contains() divergence in NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ") " + m);
   ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): contains() divergence in IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
   ensure(mThrowsIndex == tThrowsIndex, "Error (" + level + ", " + seed + "): contains() divergence in IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
   ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + level + ", " + seed + "): contains() divergence in UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);
   if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex && !mThrowsUnsupp) ensure(m.contains(KEY2OBJ(T)) ==  t.contains(KEY2OBJ(T)), "Error (" + level + ", " + seed + "): divergence between t and m (standard method) " + m);
  }

  /* Now we add and remove random data in m and t, checking that the result is the same. */

  for(int i=0; i<20*n;  i++) {
   KEY_TYPE T = genKey();

   mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal =
```

```
mThrowsUnsupp = tThrowsUnsupp = false;

  try {
   rm = m.add(KEY2OBJ(T));
  }
  catch (NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }
  catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }

  try {
   rt = t.add(KEY2OBJ(T));
  }
  catch (NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
  catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): add() divergence in
NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ") " + m);
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): add() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
  ensure(mThrowsIndex == tThrowsIndex, "Error (" + level + ", " + seed + "): add() divergence in
IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
  ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + level + ", " + seed + "): add() divergence in
UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);
  if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex && !mThrowsUnsupp) ensure(rm == rt, "Error
(" + level + ", " + seed + "): divergence in add() between t and m " + m);

  T = genKey();

  mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal =
mThrowsUnsupp = tThrowsUnsupp = false;

  try {
   rm = m.remove(KEY2OBJ(T));
  }
  catch (NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }
  catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }

  try {
   rt = t.remove(KEY2OBJ(T));
  }
  catch (NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
```

```
   catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }

   if (! KEY_EQUALS(T, k) && ! mThrowsUnsupp && tThrowsUnsupp) mThrowsUnsupp = false; // Stupid bug in
Collections.singleton()

   ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): remove() divergence in
NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ") " + m);
   ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): remove() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
   ensure(mThrowsIndex == tThrowsIndex, "Error (" + level + ", " + seed + "): remove() divergence in
IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
   // This cannot be possibly made work without a lot of fuss
   //ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + level + ", " + seed + "): remove() divergence in
UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);
   if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex && !mThrowsUnsupp) ensure(rm == rt, "Error
(" + level + ", " + seed + "): divergence in remove() between t and m " + m);
  }

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after removal " + m);
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after removal " + m);

  /* Now we add and remove random collections in m and t, checking that the result is the same. */

  for(int i=0; i<20*n;  i++) {
   KEY_TYPE T = genKey();

   mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal =
mThrowsUnsupp = tThrowsUnsupp = false;

   try {
    rm = m.addAll(java.util.Collections.singleton(KEY2OBJ(T)));
   }
   catch (NoSuchElementException e) { mThrowsNoElement = true; }
   catch (IllegalArgumentException e) { mThrowsIllegal = true; }
   catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }
   catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }

   try {
    rt = t.addAll(java.util.Collections.singleton(KEY2OBJ(T)));
   }
   catch (NoSuchElementException e) { tThrowsNoElement = true; }
   catch (IllegalArgumentException e) { tThrowsIllegal = true; }
   catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
   catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }

   ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): addAll() divergence in
NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ") " + m);
   ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): addAll() divergence in
```

```
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
  ensure(mThrowsIndex == tThrowsIndex, "Error (" + level + ", " + seed + "): addAll() divergence in
IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
  ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + level + ", " + seed + "): addAll() divergence in
UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);
  if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex && !mThrowsUnsupp) ensure(rm == rt, "Error
(" + level + ", " + seed + "): divergence in addAll() between t and m " + m);

  T = genKey();

  mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal =
mThrowsUnsupp = tThrowsUnsupp = false;

  try {
   rm = m.removeAll(java.util.Collections.singleton(KEY2OBJ(T)));
  }
  catch (NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }
  catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }

  try {
   rt = t.removeAll(java.util.Collections.singleton(KEY2OBJ(T)));
  }
  catch (NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
  catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }


  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): removeAll() divergence in
NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ") " + m);
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): removeAll() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
  ensure(mThrowsIndex == tThrowsIndex, "Error (" + level + ", " + seed + "): removeAll() divergence in
IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
  ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + level + ", " + seed + "): removeAll() divergence in
UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);
  if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex && !mThrowsUnsupp) ensure(rm == rt, "Error
(" + level + ", " + seed + "): divergence in removeAll() between t and m " + m);
  }

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after set removal " + m);
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after set removal " + m);

  /* Now we check that m actually holds the same data. */

  for(java.util.Iterator i=t.iterator(); i.hasNext();) {
```

```
  ensure(m.contains(i.next()), "Error (" + level + ", " + seed + "): m and t differ on an entry after removal (iterating on t)");
  }

  /* Now we check that m actually holds that data, but iterating on m. */

  for(java.util.Iterator i=m.iterator(); i.hasNext();) {
  ensure(t.contains(i.next()), "Error (" + level + ", " + seed + "): m and t differ on an entry after removal (iterating on m)");
  }

  if (m instanceof Singleton) {
  ensure(m.equals(((Singleton)m).clone()), "Error (" + level + ", " + seed + "): m does not equal m.clone()");
  ensure(((Singleton)m).clone().equals(m), "Error (" + level + ", " + seed + "): m.clone() does not equal m");
  }

  int h = m.hashCode();

  /* Now we save and read m. */

  SORTED_SET m2 = null;

  try {
  java.io.File ff = new java.io.File("it.unimi.dsi.fastutil.test");
  java.io.OutputStream os = new java.io.FileOutputStream(ff);
  java.io.ObjectOutputStream oos = new java.io.ObjectOutputStream(os);

  oos.writeObject(m);
  oos.close();

  java.io.InputStream is = new java.io.FileInputStream(ff);
  java.io.ObjectInputStream ois = new java.io.ObjectInputStream(is);

  m2 = (SORTED_SET)ois.readObject();
  ois.close();
  ff.delete();
  }
  catch(Exception e) {
  e.printStackTrace();
  System.exit(1);
  }

#if ! KEY_CLASS_Reference

  ensure(m2.hashCode() == h, "Error (" + level + ", " + seed + "): hashCode() changed after save/read");

  /* Now we check that m2 actually holds that data. */
```

```
  ensure(m2.equals(t), "Error (" + level + ", " + seed + "): ! m2.equals(t) after save/read");
  ensure(t.equals(m2), "Error (" + level + ", " + seed + "): ! t.equals(m2) after save/read");
 #endif

  /* Now we select a pair of keys and create a subset. */

  if (! m.isEmpty()) {
   java.util.ListIterator i;
   Object start = m.first(), end = m.first();
   for(i = (java.util.ListIterator)m.iterator(); i.hasNext() && r.nextBoolean(); start = end = i.next());
   for(; i.hasNext() && r.nextBoolean(); end = i.next());

   //System.err.println("Checking subSet from " + start + " to " + end + " (level=" + (level+1) + ")...");
   testSets(k, (SORTED_SET)m.subSet((KEY_CLASS)start, (KEY_CLASS)end), t.subSet(start, end), level - 1);

   ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after subSet");
   ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after subSet");

   //System.err.println("Checking headSet to " + end + " (level=" + (level+1) + ")...");
   testSets(k, (SORTED_SET)m.headSet((KEY_CLASS)end), t.headSet(end), level - 1);

   ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after headSet");
   ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after headSet");

   //System.err.println("Checking tailSet from " + start + " (level=" + (level+1) + ")...");
   testSets(k, (SORTED_SET)m.tailSet((KEY_CLASS)start), t.tailSet(start), level - 1);

   ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after tailSet");
   ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after tailSet");
  }

  return;
 }

 private static void test() {
  KEY_TYPE k = genKey();
  Singleton m = new Singleton(k);
  SortedSet u = new java.util.TreeSet();
  u.add(KEY2OBJ(k));
  testSets(k, m, java.util.Collections.unmodifiableSortedSet(u), 2);
  System.out.println("Test OK");
 }


 private static long seed = System.currentTimeMillis();
 private static java.util.Random r = new java.util.Random(seed);

 private static java.text.NumberFormat format = new java.text.DecimalFormat("#,###.00");
```

```
 private static java.text.FieldPosition fp = new java.text.FieldPosition(0);

 private static String format(double d) {
  StringBuffer s = new StringBuffer();
  return format.format(d, s, fp).toString();
 }

 private static void fatal(String msg) {
  System.out.println(msg);
  System.exit(1);
 }

 private static void ensure(boolean cond, String msg) {
  if (cond) return;
  fatal(msg);
 }

 /** This method expects as first argument a lower-cased type (e.g., "int"),
  * and as second optional argument a seed. */

 public static void main(String arg[]) throws Exception {
  if (arg.length > 1) r = new java.util.Random(seed = Long.parseLong(arg[1]));

  try {
   test();
  } catch(Throwable e) {
   e.printStackTrace(System.err);
   System.err.println("seed: " + seed);
  }
 }

#endif

}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/SortedSets.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
```

```
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import java.util.ListIterator;

/** A type-specific bidirectional iterator that is also a {@link ListIterator}.
*
* <p>This interface merges the methods provided by a {@link ListIterator} and
* a type-specific {@link it.unimi.dsi.fastutil.BidirectionalIterator}. Moreover, it provides
* type-specific versions of {@link ListIterator#add(Object) add()}
* and {@link ListIterator#set(Object) set()}.
*
* @see java.util.ListIterator
* @see it.unimi.dsi.fastutil.BidirectionalIterator
*/

public interface KEY_LIST_ITERATOR KEY_GENERIC extends KEY_BIDI_ITERATOR KEY_GENERIC,
ListIterator<KEY_GENERIC_CLASS> {

 /**
  * Replaces the last element returned by {@link #next} or
  * {@link #previous} with the specified element (optional operation).
  * @param k the element used to replace the last element returned.
  *
  * <p>This default implementation just throws an {@link UnsupportedOperationException}.
  * @see ListIterator#set(Object)
  */

#if KEYS_REFERENCE
 @Override
#endif
 default void set(final KEY_GENERIC_TYPE k) { throw new UnsupportedOperationException(); }

 /**
  * Inserts the specified element into the list (optional operation).
  *
  * <p>This default implementation just throws an {@link UnsupportedOperationException}.
  * @param k the element to insert.
  * @see ListIterator#add(Object)
  */
```

```
#if KEYS_REFERENCE
 @Override
#endif
 default void add(final KEY_GENERIC_TYPE k) { throw new UnsupportedOperationException(); }

 /**
  * Removes from the underlying collection the last element returned
  * by this iterator (optional operation).
  *
  * <p>This default implementation just throws an {@link UnsupportedOperationException}.
  * @see ListIterator#remove()
  */

 @Override
 default void remove() { throw new UnsupportedOperationException(); }



#if KEYS_PRIMITIVE
 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 default void set(final KEY_CLASS k) { set(k.KEY_VALUE()); }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 default void add(final KEY_CLASS k) { add(k.KEY_VALUE()); }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 default KEY_GENERIC_CLASS next() { return KEY_BIDI_ITERATOR.super.next(); }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 default KEY_GENERIC_CLASS previous() { return KEY_BIDI_ITERATOR.super.previous(); }
#endif

}
```

Found in path(s):

* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/ListIterator.drv
No license file was found, but licenses were detected in source scan.


```
package PACKAGE;

import it.unimi.dsi.fastutil.objects.AbstractObjectList;
import it.unimi.dsi.fastutil.objects.ObjectListIterator;
import it.unimi.dsi.fastutil.longs.LongArrays;

import java.io.Serializable;
import java.util.Iterator;
import java.util.Collection;
import java.util.NoSuchElementException;
import java.util.RandomAccess;

/** Compact storage of lists of arrays using front coding.
*
* <p>This class stores immutably a list of arrays in a single large array
* using front coding (of course, the compression will be reasonable only if
* the list is sorted lexicographically&mdash;see below). It implements an
* immutable type-specific list that returns the <var>i</var>-th array when
* calling {@link #get(int) get(<var>i</var>)}. The returned array may be
* freely modified.
*
* <p>Front coding is based on the idea that if the <var>i</var>-th and the
* (<var>i</var>+1)-th array have a common prefix, we might store the length
* of the common prefix, and then the rest of the second array.
*
* <p>This approach, of course, requires that once in a while an array is
* stored entirely.  The <em>ratio</em> of a front-coded list defines how
```

```
* often this happens (once every {@link #ratio()} arrays). A higher ratio
* means more compression, but means also a longer access time, as more arrays
* have to be probed to build the result. Note that we must build an array
* every time {@link #get(int)} is called, but this class provides also methods
* that extract one of the stored arrays in a given array, reducing garbage
* collection. See the documentation of the family of {@code get()}
* methods.
*
* <p>By setting the ratio to 1 we actually disable front coding: however, we
* still have a data structure storing large list of arrays with a reduced
* overhead (just one integer per array, plus the space required for lengths).
*
* <p>Note that the typical usage of front-coded lists is under the form of
* serialized objects; usually, the data that has to be compacted is processed
* offline, and the resulting structure is stored permanently. Since the
* pointer array is not stored, the serialized format is very small.
*
* <H2>Implementation Details</H2>
*
* <p>All arrays are stored in a {@linkplain it.unimi.dsi.fastutil.BigArrays big array}. A separate array of pointers
* indexes arrays whose position is a multiple of the ratio: thus, a higher ratio
* means also less pointers.
*
* <p>More in detail, an array whose position is a multiple of the ratio is
* stored as the array length, followed by the elements of the array. The array
* length is coded by a simple variable-length list of <var>k</var>-1 bit
* blocks, where <var>k</var> is the number of bits of the underlying primitive
* type.  All other arrays are stored as follows: let {@code common} the
* length of the maximum common prefix between the array and its predecessor.
* Then we store the array length decremented by {@code common}, followed
* by {@code common}, followed by the array elements whose index is
* greater than or equal to {@code common}. For instance, if we store
* {@code foo}, {@code foobar}, {@code football} and
* {@code fool} in a front-coded character-array list with ratio 3, the
* character array will contain
*
* <pre>
* <b>3</b> f o o <b>3</b> <b>3</b> b a r <b>5</b> <b>3</b> t b a l l <b>4</b> f o o l
* </pre>
*/

public class ARRAY_FRONT_CODED_LIST extends AbstractObjectList<KEY_TYPE[]> implements
Serializable, Cloneable, RandomAccess {

 private static final long serialVersionUID = 1L;

 /** The number of arrays in the list. */
 protected final int n;
```

```
/** The ratio of this front-coded list. */
protected final int ratio;
/** The big array containing the compressed arrays. */
protected final KEY_TYPE[][] array;
/** The pointers to entire arrays in the list. */
protected transient long[] p;

/** Creates a new front-coded list containing the arrays returned by the given iterator.
 *
 * @param arrays an iterator returning arrays.
 * @param ratio the desired ratio.
 */

public ARRAY_FRONT_CODED_LIST(final Iterator<KEY_TYPE[]> arrays, final int ratio) {

 if (ratio < 1) throw new IllegalArgumentException("Illegal ratio (" + ratio + ")");

 KEY_TYPE[][] array = BIG_ARRAYS.EMPTY_BIG_ARRAY;
 long[] p = LongArrays.EMPTY_ARRAY;

 KEY_TYPE[][] a = new KEY_TYPE[2][];
 long curSize = 0;
 int n = 0, b = 0, common, length, minLength;

 while(arrays.hasNext()) {
  a[b] = arrays.next();
  length = a[b].length;

  if (n % ratio == 0) {
   p = LongArrays.grow(p, n / ratio + 1);
   p[n / ratio] = curSize;

   array = BIG_ARRAYS.grow(array, curSize + count(length) + length, curSize);
   curSize += writeInt(array, length, curSize);
   BIG_ARRAYS.copyToBig(a[b], 0, array, curSize, length);
   curSize += length;
  }
  else {
   minLength = a[1 - b].length;
   if (length < minLength) minLength = length;
   for(common = 0; common < minLength; common++) if (a[0][common] != a[1][common]) break;
   length -= common;

   array = BIG_ARRAYS.grow(array, curSize + count(length) + count(common) + length, curSize);
   curSize += writeInt(array, length, curSize);
   curSize += writeInt(array, common, curSize);
   BIG_ARRAYS.copyToBig(a[b], common, array, curSize, length);
   curSize += length;
```

```
 }

 b = 1 - b;
 n++;
 }

 this.n = n;
 this.ratio = ratio;
 this.array = BIG_ARRAYS.trim(array, curSize);
 this.p = LongArrays.trim(p, (n + ratio - 1) / ratio);

}

/** Creates a new front-coded list containing the arrays in the given collection.
 *
 * @param c a collection containing arrays.
 * @param ratio the desired ratio.
 */

public ARRAY_FRONT_CODED_LIST(final Collection<KEY_TYPE[]> c, final int ratio) {
 this(c.iterator(), ratio);
}



/* The following (rather messy) methods implements the encoding of arbitrary integers inside a big array.
 * Unfortunately, we have to specify different codes for almost every type. */

/** Reads a coded length.
 * @param a the data big array.
 * @param pos the starting position.
 * @return the length coded at {@code pos}.
 */
private static int readInt(final KEY_TYPE a[][], long pos) {
#if KEY_CLASS_Integer
  return IntBigArrays.get(a, pos);
#elif KEY_CLASS_Long
  return (int)LongBigArrays.get(a, pos);
#elif KEY_CLASS_Character
  final char c0 = CharBigArrays.get(a, pos);
  return c0  < 0x8000 ? c0 : (c0 & 0x7FFF) << 16 | CharBigArrays.get(a, pos + 1);
#elif KEY_CLASS_Short
  final short s0 = ShortBigArrays.get(a, pos);
  return s0 >= 0 ? s0 : s0 << 16 | (ShortBigArrays.get(a, pos + 1) & 0xFFFF);
#else
  final byte b0 = ByteBigArrays.get(a, pos);
  if (b0  >= 0) return b0;
  final byte b1 = ByteBigArrays.get(a, pos + 1);
```

```
  if (b1 >= 0) return (- b0 - 1) << 7 | b1;
  final byte b2 = ByteBigArrays.get(a, pos + 2);
  if (b2 >= 0) return (- b0 - 1) << 14 | (- b1 - 1) << 7 | b2;
  final byte b3 = ByteBigArrays.get(a, pos + 3);
  if (b3 >= 0) return (- b0 - 1) << 21 | (- b1 - 1) << 14 | (- b2 - 1) << 7 | b3;
  return (- b0 - 1) << 28 | (- b1 - 1) << 21 | (- b2 - 1) << 14 | (- b3 - 1) << 7 | ByteBigArrays.get(a, pos + 4);
#endif
 }

 /** Computes the number of elements coding a given length.
  * @param length the length to be coded.
  * @return the number of elements coding {@code length}.
  */
 private static int count(final int length) {
#if KEY_CLASS_Integer || KEY_CLASS_Long
  return 1;
#elif KEY_CLASS_Character || KEY_CLASS_Short
  return length < (1 << 15) ? 1 : 2;
#else
  if (length < (1 << 7)) return 1;
  if (length < (1 << 14)) return 2;
  if (length < (1 << 21)) return 3;
  if (length < (1 << 28)) return 4;
  return 5;
#endif
 }

 /** Writes a length.
  * @param a the data array.
  * @param length the length to be written.
  * @param pos the starting position.
  * @return the number of elements coding {@code length}.
  */
 private static int writeInt(final KEY_TYPE a[][], int length, long pos) {
#if KEY_CLASS_Long
  LongBigArrays.set(a, pos, length);
  return 1;
#elif KEY_CLASS_Integer
  IntBigArrays.set(a, pos, length);
  return 1;
#elif KEY_CLASS_Character
  if (length < (1 << 15)) {
   CharBigArrays.set(a, pos, (char)length);
   return 1;
  }
  CharBigArrays.set(a, pos++, (char)(length >>> 16 | 0x8000));
  CharBigArrays.set(a, pos, (char)(length & 0xFFFF));
  return 2;
```

```
#elif KEY_CLASS_Short
  if (length < (1 << 15)) {
   ShortBigArrays.set(a, pos, (short)length);
   return 1;
  }
  ShortBigArrays.set(a, pos++, (short)(- (length >>> 16) - 1));
  ShortBigArrays.set(a, pos, (short)(length & 0xFFFF));
  return 2;
#else
  final int count = count(length);
  ByteBigArrays.set(a, pos + count - 1, (byte)(length & 0x7F));

  if (count != 1) {
   int i = count - 1;
   while(i-- != 0) {
    length >>>= 7;
    ByteBigArrays.set(a, pos + i, (byte)(- (length  & 0x7F) - 1));
   }
  }

  return count;
#endif
 }



 /** Returns the ratio of this list.
  *
  * @return the ratio of this list.
  */

 public int ratio() {
  return ratio;
 }



 /** Computes the length of the array at the given index.
  *
  * <p>This private version of {@link #arrayLength(int)} does not check its argument.
  *
  * @param index an index.
  * @return the length of the {@code index}-th array.
  */
 private int length(final int index) {
  final KEY_TYPE[][] array = this.array;
  final int delta = index % ratio; // The index into the p array, and the delta inside the block.

  long pos = p[index / ratio]; // The position into the array of the first entire word before the index-th.
```

```
 int length = readInt(array, pos);

 if (delta == 0) return length;

 // First of all, we recover the array length and the maximum amount of copied elements.
 int common;
 pos += count(length) + length;
 length = readInt(array, pos);
 common = readInt(array, pos + count(length));

 for(int i = 0; i < delta - 1; i++) {
  pos += count(length) + count(common) + length;
  length = readInt(array, pos);
  common = readInt(array, pos + count(length));
 }

 return length + common;
}


/** Computes the length of the array at the given index.
 *
 * @param index an index.
 * @return the length of the {@code index}-th array.
 */
public int arrayLength(final int index) {
 ensureRestrictedIndex(index);
 return length(index);
}

/** Extracts the array at the given index.
 *
 * @param index an index.
 * @param a the array that will store the result (we assume that it can hold the result).
 * @param offset an offset into {@code a} where elements will be store.
 * @param length a maximum number of elements to store in {@code a}.
 * @return the length of the extracted array.
 */
private int extract(final int index, final KEY_TYPE a[], final int offset, final int length) {
 final int delta = index % ratio; // The delta inside the block.
 final long startPos = p[index / ratio]; // The position into the array of the first entire word before the index-th.
 long pos, prevArrayPos;
 int arrayLength = readInt(array, pos = startPos), currLen = 0, actualCommon;

 if (delta == 0) {
  pos = p[index / ratio] + count(arrayLength);
  BIG_ARRAYS.copyFromBig(array, pos, a, offset, Math.min(length, arrayLength));
  return arrayLength;
```

```
 }

 int common = 0;

 for(int i = 0; i < delta; i++) {
  prevArrayPos = pos + count(arrayLength) + (i != 0 ? count(common) : 0);
  pos = prevArrayPos + arrayLength;

  arrayLength = readInt(array, pos);
  common = readInt(array, pos + count(arrayLength));

  actualCommon = Math.min(common, length);
  if (actualCommon <= currLen) currLen = actualCommon;
  else {
   BIG_ARRAYS.copyFromBig(array, prevArrayPos, a, currLen + offset, actualCommon - currLen);
   currLen = actualCommon;
  }
 }

  if (currLen < length) BIG_ARRAYS.copyFromBig(array, pos + count(arrayLength) + count(common), a, currLen +
offset, Math.min(arrayLength, length - currLen));

 return arrayLength + common;
 }

 /** {@inheritDoc}
  * <p>This implementation delegates to {@link #getArray(int)}. */
 @Override
 public KEY_TYPE[] get(final int index) {
  return getArray(index);
 }

 /** Returns an array stored in this front-coded list.
  *
  * @param index an index.
  * @return the corresponding array stored in this front-coded list.
  */
 public KEY_TYPE[] getArray(final int index) {
  ensureRestrictedIndex(index);
  final int length = length(index);
  final KEY_TYPE a[] = new KEY_TYPE[length];
  extract(index, a, 0, length);
  return a;
 }

 /** Stores in the given array elements from an array stored in this front-coded list.
  *
  * @param index an index.
```

```
 * @param a the array that will store the result.
 * @param offset an offset into {@code a} where elements will be store.
 * @param length a maximum number of elements to store in {@code a}.
 * @return if {@code a} can hold the extracted elements, the number of extracted elements;
 * otherwise, the number of remaining elements with the sign changed.
 */
public int get(final int index, final KEY_TYPE[] a, final int offset, final int length) {
 ensureRestrictedIndex(index);
 ARRAYS.ensureOffsetLength(a, offset, length);

 final int arrayLength = extract(index, a, offset, length);
 if (length >= arrayLength) return arrayLength;
 return length - arrayLength;
}

/** Stores in the given array an array stored in this front-coded list.
 *
 * @param index an index.
 * @param a the array that will store the content of the result (we assume that it can hold the result).
 * @return if {@code a} can hold the extracted elements, the number of extracted elements;
 * otherwise, the number of remaining elements with the sign changed.
 */
public int get(final int index, final KEY_TYPE[] a) {
 return get(index, a, 0, a.length);
}

@Override
public int size() {
 return n;
}

@Override
public ObjectListIterator<KEY_TYPE[]> listIterator(final int start) {
 ensureIndex(start);

 return new ObjectListIterator<KEY_TYPE[]>() {
  KEY_TYPE s[] = ARRAYS.EMPTY_ARRAY;
  int i = 0;
  long pos = 0;
  boolean inSync; // Whether the current value in a is the string just before the next to be produced.

  {
   if (start != 0) {
    if (start == n) i = start; // If we start at the end, we do nothing.
    else {
     pos = p[start / ratio];
     int j = start % ratio;
     i = start - j;
```

```
   while(j-- != 0) next();
  }
 }
}

@Override
public boolean hasNext() {
 return i < n;
}

@Override
public boolean hasPrevious() {
 return i > 0;
}

@Override
public int previousIndex() {
 return i - 1;
}

@Override
public int nextIndex() {
 return i;
}

@Override
public KEY_TYPE[] next() {
 int length, common;

 if (! hasNext()) throw new NoSuchElementException();

 if (i % ratio == 0) {
  pos = p[i / ratio];
  length = readInt(array, pos);
  s = ARRAYS.ensureCapacity(s, length, 0);
  BIG_ARRAYS.copyFromBig(array, pos + count(length), s, 0, length);
  pos += length + count(length);
  inSync = true;
 }
 else {
  if (inSync) {
   length = readInt(array, pos);
   common = readInt(array, pos + count(length));
   s = ARRAYS.ensureCapacity(s, length + common, common);
   BIG_ARRAYS.copyFromBig(array, pos + count(length) + count (common), s, common, length);
   pos += count(length) + count(common) + length;
   length += common;
  }
```

```
     else {
      s = ARRAYS.ensureCapacity(s, length = length(i), 0);
      extract(i, s, 0, length);
     }
    }
    i++;
    return ARRAYS.copy(s, 0, length);
   }

   @Override
   public KEY_TYPE[] previous() {
    if (! hasPrevious()) throw new NoSuchElementException();
    inSync = false;
    return getArray(--i);
   }
  };
}


/** Returns a copy of this list.
 *
 *  @return a copy of this list.
 */
@Override
public ARRAY_FRONT_CODED_LIST clone() {
 return this;
}

@Override
public String toString() {
 final StringBuffer s = new StringBuffer();
 s.append("[");
 for(int i = 0; i < n; i++) {
  if (i != 0) s.append(", ");
  s.append(ARRAY_LIST.wrap(getArray(i)).toString());
 }
 s.append("]");
 return s.toString();
}

/** Computes the pointer array using the currently set ratio, number of elements and underlying array.
 *
 * @return the computed pointer array.
 */

protected long[] rebuildPointerArray() {
 final long[] p = new long[(n + ratio - 1) / ratio];
 final KEY_TYPE a[][] = array;
```

```
 int length, count;
 long pos = 0;

 for(int i = 0, j = 0, skip = ratio - 1; i < n; i++) {
  length = readInt(a, pos);
  count = count(length);
  if (++skip == ratio) {
   skip = 0;
   p[j++] = pos;
   pos += count + length;
  }
  else pos += count + count(readInt(a, pos + count)) + length;
 }

 return p;
}


 private void readObject(java.io.ObjectInputStream s) throws java.io.IOException, ClassNotFoundException {
  s.defaultReadObject();

  // Rebuild pointer array
  p = rebuildPointerArray();
 }


#ifdef TEST

 private static long seed = System.currentTimeMillis();
 private static java.util.Random r = new java.util.Random(seed);

 private static KEY_TYPE genKey() {
#if KEY_CLASS_Byte || KEY_CLASS_Short || KEY_CLASS_Character
  return (KEY_TYPE)(r.nextInt());
#elif KEYS_PRIMITIVE
  return r.NEXT_KEY();
#elif KEY_CLASS_Object
  return Integer.toBinaryString(r.nextInt());
#else
  return new java.io.Serializable() {};
#endif
 }


 private static java.text.NumberFormat format = new java.text.DecimalFormat("#,###.00");
 private static java.text.FieldPosition fp = new java.text.FieldPosition(0);

 private static String format(double d) {
```

```
 StringBuffer s = new StringBuffer();
 return format.format(d, s, fp).toString();
 }

 private static void speedTest(int n, boolean comp) {
 System.out.println("There are presently no speed tests for this class.");
 }


 private static void fatal(String msg) {
 System.out.println(msg);
 System.exit(1);
 }

 private static void ensure(boolean cond, String msg) {
 if (cond) return;
 fatal(msg);
 }

 private static boolean contentEquals(java.util.List x, java.util.List y) {
 if (x.size() != y.size()) return false;
 for(int i = 0; i < x.size(); i++) if (! java.util.Arrays.equals((KEY_TYPE[])x.get(i), (KEY_TYPE[])y.get(i))) return
false;
 return true;
 }

 private static int l[];
 private static KEY_TYPE[][] a;


 private static void runTest(int n) {
 int c;

 l = new int[n];
 a = new KEY_TYPE[n][];

 for(int i = 0; i < n; i++) l[i] = (int)(Math.abs(r.nextGaussian())*32);
 for(int i = 0; i < n; i++) a[i] = new KEY_TYPE[l[i]];
 for(int i = 0; i < n; i++) for(int j = 0; j < l[i]; j++) a[i][j] = genKey();

 ARRAY_FRONT_CODED_LIST m = new
ARRAY_FRONT_CODED_LIST(it.unimi.dsi.fastutil.objects.ObjectIterators.wrap(a), r.nextInt(4) + 1);
 it.unimi.dsi.fastutil.objects.ObjectArrayList t = new it.unimi.dsi.fastutil.objects.ObjectArrayList(a);

 //System.out.println(m);
 //for(i = 0; i < t.size(); i++) System.out.println(ARRAY_LIST.wrap((KEY_TYPE[])t.get(i)));

 /* Now we check that m actually holds that data. */
```

```
  ensure(contentEquals(m, t), "Error (" + seed + "): m does not equal t at creation");

  /* Now we check cloning. */

  ensure(contentEquals(m, (java.util.List)m.clone()), "Error (" + seed + "): m does not equal m.clone()");

  /* Now we play with iterators. */

  {
   ObjectListIterator i;
   java.util.ListIterator j;
   Object J;
   i = m.listIterator();
   j = t.listIterator();

   for(int k = 0; k < 2*n; k++) {
    ensure(i.hasNext() == j.hasNext(), "Error (" + seed + "): divergence in hasNext()");
    ensure(i.hasPrevious() == j.hasPrevious(), "Error (" + seed + "): divergence in hasPrevious()");

    if (r.nextFloat() < .8 && i.hasNext()) {
     ensure(java.util.Arrays.equals((KEY_TYPE[])i.next(), (KEY_TYPE[])j.next()), "Error (" + seed + "): divergence in
next()");

    }
    else if (r.nextFloat() < .2 && i.hasPrevious()) {
     ensure(java.util.Arrays.equals((KEY_TYPE[])i.previous(), (KEY_TYPE[])j.previous()), "Error (" + seed + "):
divergence in previous()");
    }

    ensure(i.nextIndex() == j.nextIndex(), "Error (" + seed + "): divergence in nextIndex()");
    ensure(i.previousIndex() == j.previousIndex(), "Error (" + seed + "): divergence in previousIndex()");

   }

  }

  {
   Object previous = null;
   Object I, J;
   int from = r.nextInt(m.size() +1);
   ObjectListIterator i;
   java.util.ListIterator j;
   i = m.listIterator(from);
   j = t.listIterator(from);

   for(int k = 0; k < 2*n; k++) {
    ensure(i.hasNext() == j.hasNext(), "Error (" + seed + "): divergence in hasNext() (iterator with starting point " +
```

```
from + ")");
    ensure(i.hasPrevious() == j.hasPrevious() , "Error (" + seed + "): divergence in hasPrevious() (iterator with starting point " + from + ")");

    if (r.nextFloat() < .8 && i.hasNext()) {
     ensure(java.util.Arrays.equals((KEY_TYPE[])i.next(), (KEY_TYPE[])j.next()), "Error (" + seed + "): divergence in next() (iterator with starting point " + from + ")");
     //System.err.println("Done next " + I + " " + J + "  " + badPrevious);

    }
    else if (r.nextFloat() < .2 && i.hasPrevious()) {
     ensure(java.util.Arrays.equals((KEY_TYPE[])i.previous(), (KEY_TYPE[])j.previous()), "Error (" + seed + "): divergence in previous() (iterator with starting point " + from + ")");

    }
   }

  }



  try {
   java.io.File ff = new java.io.File("it.unimi.dsi.fastutil.test");
   java.io.OutputStream os = new java.io.FileOutputStream(ff);
   java.io.ObjectOutputStream oos = new java.io.ObjectOutputStream(os);

   oos.writeObject(m);
   oos.close();

   java.io.InputStream is = new java.io.FileInputStream(ff);
   java.io.ObjectInputStream ois = new java.io.ObjectInputStream(is);

   m = (ARRAY_FRONT_CODED_LIST)ois.readObject();
   ois.close();
   ff.delete();
  }
  catch(Exception e) {
   e.printStackTrace();
   System.exit(1);
  }

  ensure(contentEquals(m, t), "Error (" + seed + "): m does not equal t after save/read");

  System.out.println("Test OK");
  return;
 }
```

```
 public static void main(String args[]) {
  int n  = Integer.parseInt(args[1]);
  if (args.length > 2) r = new java.util.Random(seed = Long.parseLong(args[2]));

  try {
   if ("speedTest".equals(args[0]) || "speedComp".equals(args[0])) speedTest(n, "speedComp".equals(args[0]));
   else if ("test".equals(args[0])) runTest(n);
  } catch(Throwable e) {
   e.printStackTrace(System.err);
   System.err.println("seed: " + seed);
  }
 }

#endif


}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/ArrayFrontCodedList.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2003-2017 Paolo Boldi and Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

#if KEY_CLASS_Object
import java.util.Comparator;
#endif

import it.unimi.dsi.fastutil.ints.IntArrays;
```

```
/** A class providing static methods and objects that do useful things with semi-indirect heaps.
*
* <p>A semi-indirect heap is based on a <em>reference array</em>. Elements of
* a semi-indirect heap are integers that index the reference array (note that
* in an <em>indirect</em> heap you can also map elements of the reference
* array to heap positions).
*/

public final class SEMI_INDIRECT_HEAPS {

 private SEMI_INDIRECT_HEAPS() {}

 /** Moves the given element down into the semi-indirect heap until it reaches the lowest possible position.
  *
  * @param refArray the reference array.
  * @param heap the semi-indirect heap (starting at 0).
  * @param size the number of elements in the heap.
  * @param i the index in the heap of the element to be moved down.
  * @param c a type-specific comparator, or {@code null} for the natural order.
  * @return the new position in the heap of the element of heap index {@code i}.
  */

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public static KEY_GENERIC int downHeap(final KEY_GENERIC_TYPE[] refArray, final int[] heap, final int size,
int i, final KEY_COMPARATOR KEY_GENERIC c) {
  assert i < size;

  final int e = heap[i];
  final KEY_GENERIC_TYPE E = refArray[e];
  int child;

  if (c == null)
   while ((child = (i << 1) + 1) < size) {
    int t = heap[child];
    final int right = child + 1;
    if (right < size && KEY_LESS(refArray[heap[right]], refArray[t])) t = heap[child = right];
    if (KEY_LESSEQ(E, refArray[t])) break;
    heap[i] = t;
    i = child;
   }
  else
   while ((child = (i << 1) + 1) < size) {
    int t = heap[child];
    final int right = child + 1;
    if (right < size && c.compare(refArray[heap[right]], refArray[t]) < 0) t = heap[child = right];
    if (c.compare(E, refArray[t]) <= 0) break;
    heap[i] = t;
    i = child;
```

```
 }

 heap[i] = e;

 return i;
}

/** Moves the given element up in the semi-indirect heap until it reaches the highest possible position.
 *
 * @param refArray the reference array.
 * @param heap the semi-indirect heap (starting at 0).
 * @param size the number of elements in the heap.
 * @param i the index in the heap of the element to be moved up.
 * @param c a type-specific comparator, or {@code null} for the natural order.
 * @return the new position in the heap of the element of heap index {@code i}.
 */

SUPPRESS_WARNINGS_KEY_UNCHECKED
public static KEY_GENERIC int upHeap(final KEY_GENERIC_TYPE[] refArray, final int[] heap, final int size, int
i, final KEY_COMPARATOR KEY_GENERIC c) {
 assert i < size;

 final int e = heap[i];
 final KEY_GENERIC_TYPE E = refArray[e];

 if (c == null)
  while (i != 0) {
   final int parent = (i - 1) >>> 1;
   final int t = heap[parent];
   if (KEY_LESSEQ(refArray[t], E)) break;
   heap[i] = t;
   i = parent;
  }
 else
  while (i != 0) {
   final int parent = (i - 1) >>> 1;
   final int t = heap[parent];
   if (c.compare(refArray[t], E) <= 0) break;
   heap[i] = t;
   i = parent;
  }

 heap[i] = e;

 return i;
}

/** Creates a semi-indirect heap in the given array.
```

```
 *
 * @param refArray the reference array.
 * @param offset the first element of the reference array to be put in the heap.
 * @param length the number of elements to be put in the heap.
 * @param heap the array where the heap is to be created.
 * @param c a type-specific comparator, or {@code null} for the natural order.
 */

 public static KEY_GENERIC void makeHeap(final KEY_GENERIC_TYPE[] refArray, final int offset, final int
length, final int[] heap, final KEY_COMPARATOR KEY_GENERIC c) {
  ARRAYS.ensureOffsetLength(refArray, offset, length);
  if (heap.length < length) throw new IllegalArgumentException("The heap length (" + heap.length + ") is smaller
than the number of elements (" + length + ")");

  int i = length;
  while(i-- != 0) heap[i] = offset + i;

  i = length >>> 1;
  while(i-- != 0) downHeap(refArray, heap, length, i, c);
 }

 /** Creates a semi-indirect heap, allocating its heap array.
 *
 * @param refArray the reference array.
 * @param offset the first element of the reference array to be put in the heap.
 * @param length the number of elements to be put in the heap.
 * @param c a type-specific comparator, or {@code null} for the natural order.
 * @return the heap array.
 */

 public static KEY_GENERIC int[] makeHeap(final KEY_GENERIC_TYPE[] refArray, final int offset, final int
length, final KEY_COMPARATOR KEY_GENERIC c) {
  final int[] heap = length <= 0 ? IntArrays.EMPTY_ARRAY : new int[length];
  makeHeap(refArray, offset, length, heap, c);
  return heap;
 }



 /** Creates a semi-indirect heap from a given index array.
 *
 * @param refArray the reference array.
 * @param heap an array containing indices into {@code refArray}.
 * @param size the number of elements in the heap.
 * @param c a type-specific comparator, or {@code null} for the natural order.
 */

 public static KEY_GENERIC void makeHeap(final KEY_GENERIC_TYPE[] refArray, final int[] heap, final int
```

```
size, final KEY_COMPARATOR KEY_GENERIC c) {
 int i = size >>> 1;
 while(i-- != 0) downHeap(refArray, heap, size, i, c);
 }

 /** Retrieves the front of a heap in a given array.
  *
  * <p>The <em>front</em> of a semi-indirect heap is the set of indices whose associated elements in the reference
array
  * are equal to the element associated to the first index.
  *
  * <p>In several circumstances you need to know the front, and scanning linearly the entire heap is not
  * the best strategy. This method simulates (using a partial linear scan) a breadth-first visit that
  * terminates when all visited nodes are larger than the element associated
  * to the top index, which implies that no elements of the front can be found later.
  * In most cases this trick yields a significant improvement.
  *
  * @param refArray the reference array.
  * @param heap an array containing indices into {@code refArray}.
  * @param size the number of elements in the heap.
  * @param a an array large enough to hold the front (e.g., at least long as {@code refArray}).
  * @return the number of elements actually written (starting from the first position of {@code a}).
  */
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public static KEY_GENERIC int front(final KEY_GENERIC_TYPE[] refArray, final int[] heap, final int size, final
int[] a) {
 final KEY_GENERIC_TYPE top = refArray[heap[0]];
 int j = 0, // The current position in a
  l = 0, // The first position to visit in the next level (inclusive)
  r = 1, // The last position to visit in the next level (exclusive)
  f = 0; // The first position (in the heap array) of the next level
 for(int i = 0; i < r; i++) {
  if (i == f) { // New level
   if (l >= r) break; // If we are crossing the two bounds, we're over
   f = (f << 1) + 1; // Update the first position of the next level...
   i = l; // ...and jump directly to position l
   l = -1; // Invalidate l
   }
  if (KEY_CMP_EQ(top, refArray[heap[i]])) {
   a[j++] = heap[i];
   if (l == -1) l = i * 2 + 1; // If this is the first time in this level, set l
   r = Math.min(size, i * 2 + 3); // Update r, but do not go beyond size
   }
  }

 return j;
 }
```

```
 /** Retrieves the front of a heap in a given array using a given comparator.
  *
  * <p>The <em>front</em> of a semi-indirect heap is the set of indices whose associated elements in the reference array
  * are equal to the element associated to the first index.
  *
  * <p>In several circumstances you need to know the front, and scanning linearly the entire heap is not
  * the best strategy. This method simulates (using a partial linear scan) a breadth-first visit that
  * terminates when all visited nodes are larger than the element associated
  * to the top index, which implies that no elements of the front can be found later.
  * In most cases this trick yields a significant improvement.
  *
  * @param refArray the reference array.
  * @param heap an array containing indices into {@code refArray}.
  * @param size the number of elements in the heap.
  * @param a an array large enough to hold the front (e.g., at least long as {@code refArray}).
  * @param c a type-specific comparator.
  * @return the number of elements actually written (starting from the first position of {@code a}).
  */
 public static KEY_GENERIC int front(final KEY_GENERIC_TYPE[] refArray, final int[] heap, final int size, final int[] a, final KEY_COMPARATOR KEY_GENERIC c) {
  final KEY_GENERIC_TYPE top = refArray[heap[0]];
  int j = 0, // The current position in a
   l = 0, // The first position to visit in the next level (inclusive)
   r = 1, // The last position to visit in the next level (exclusive)
   f = 0; // The first position (in the heap array) of the next level
  for(int i = 0; i < r; i++) {
   if (i == f) { // New level
    if (l >= r) break; // If we are crossing the two bounds, we're over
    f = (f << 1) + 1; // Update the first position of the next level...
    i = l; // ...and jump directly to position l
    l = -1; // Invalidate l
    }
   if (c.compare(top, refArray[heap[i]]) == 0) {
    a[j++] = heap[i];
    if (l == -1) l = i * 2 + 1; // If this is the first time in this level, set l
    r = Math.min(size, i * 2 + 3); // Update r, but do not go beyond size
    }
   }

  return j;
 }
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/SemiIndirectHeaps.drv

No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2010-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*   http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/Swapper.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/BigList.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/Size64.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/BigListIterator.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/BigSwapper.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*   http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

```
package PACKAGE;

import java.util.List;
import java.util.Collection;
import java.util.Random;
import java.util.RandomAccess;

/** A class providing static methods and objects that do useful things with type-specific lists.
*
* @see java.util.Collections
*/

public final class LISTS {

 private LISTS() {}

 /** Shuffles the specified list using the specified pseudorandom number generator.
  *
  * @param l the list to be shuffled.
  * @param random a pseudorandom number generator.
  * @return {@code l}.
  */
 public static KEY_GENERIC LIST KEY_GENERIC shuffle(final LIST KEY_GENERIC l, final Random random)
{
  for(int i = l.size(); i-- != 0;) {
   final int p = random.nextInt(i + 1);
   final KEY_GENERIC_TYPE t = l.GET_KEY(i);
   l.set(i, l.GET_KEY(p));
   l.set(p, t);
  }
  return l;
 }

 /** An immutable class representing an empty type-specific list.
  *
  * <p>This class may be useful to implement your own in case you subclass
  * a type-specific list.
  */

 public static class EmptyList KEY_GENERIC extends COLLECTIONS.EmptyCollection KEY_GENERIC
implements LIST KEY_GENERIC, RandomAccess, java.io.Serializable, Cloneable {

  private static final long serialVersionUID = -7046029254386353129L;

  protected EmptyList() {}

  @Override
  public KEY_GENERIC_TYPE GET_KEY(int i) { throw new IndexOutOfBoundsException(); }
```

```
  @Override
  public boolean REMOVE(KEY_TYPE k) { throw new UnsupportedOperationException(); }

  @Override
  public KEY_GENERIC_TYPE REMOVE_KEY(int i) { throw new UnsupportedOperationException(); }

  @Override
  public void add(final int index, final KEY_GENERIC_TYPE k) { throw new UnsupportedOperationException(); }

  @Override
  public KEY_GENERIC_TYPE set(final int index, final KEY_GENERIC_TYPE k) { throw new
UnsupportedOperationException(); }

  @Override
  public int indexOf(KEY_TYPE k) { return -1; }

  @Override
  public int lastIndexOf(KEY_TYPE k) { return -1; }

  @Override
  public boolean addAll(int i, Collection<? extends KEY_GENERIC_CLASS> c) { throw new
UnsupportedOperationException(); }

#if KEYS_PRIMITIVE
  @Override
  public boolean addAll(LIST c) { throw new UnsupportedOperationException(); }

  @Override
  public boolean addAll(int i, COLLECTION c) { throw new UnsupportedOperationException(); }

  @Override
  public boolean addAll(int i, LIST c) { throw new UnsupportedOperationException(); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @SuppressWarnings("deprecation")
  @Deprecated
  @Override
  public void add(final int index, final KEY_GENERIC_CLASS k) { throw new UnsupportedOperationException(); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @SuppressWarnings("deprecation")
  @Deprecated
  @Override
  public KEY_CLASS get(final int index) { throw new UnsupportedOperationException(); }
```

```
  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @SuppressWarnings("deprecation")
  @Deprecated
  @Override
  public boolean add(final KEY_GENERIC_CLASS k) { throw new UnsupportedOperationException(); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @SuppressWarnings("deprecation")
  @Deprecated
  @Override
  public KEY_GENERIC_CLASS set(final int index, final KEY_GENERIC_CLASS k) { throw new
UnsupportedOperationException(); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @SuppressWarnings("deprecation")
  @Deprecated
  @Override
  public KEY_GENERIC_CLASS remove(int k) { throw new UnsupportedOperationException(); }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @SuppressWarnings("deprecation")
  @Deprecated
  @Override
  public int indexOf(Object k) { return -1; }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @SuppressWarnings("deprecation")
  @Deprecated
  @Override
  public int lastIndexOf(Object k) { return -1; }
#endif
  SUPPRESS_WARNINGS_KEY_UNCHECKED
  @Override
  public KEY_LIST_ITERATOR KEY_GENERIC listIterator() { return ITERATORS.EMPTY_ITERATOR; }

  SUPPRESS_WARNINGS_KEY_UNCHECKED
  @Override
  public KEY_LIST_ITERATOR KEY_GENERIC iterator() { return ITERATORS.EMPTY_ITERATOR; }

  SUPPRESS_WARNINGS_KEY_UNCHECKED
  @Override
  public KEY_LIST_ITERATOR KEY_GENERIC listIterator(int i) { if (i == 0) return
ITERATORS.EMPTY_ITERATOR; throw new IndexOutOfBoundsException(String.valueOf(i)); }
```

```
  @Override
  public LIST KEY_GENERIC subList(int from, int to) { if (from == 0 && to == 0) return this; throw new
IndexOutOfBoundsException(); }

  @Override
  public void getElements(int from, KEY_TYPE[] a, int offset, int length) { if (from == 0 && length == 0 && offset
>= 0 && offset <= a.length) return; throw new IndexOutOfBoundsException(); }

  @Override
  public void removeElements(int from, int to) { throw new UnsupportedOperationException(); }

  @Override
  public void addElements(int index, final KEY_GENERIC_TYPE a[], int offset, int length) { throw new
UnsupportedOperationException(); }

  @Override
  public void addElements(int index, final KEY_GENERIC_TYPE a[]) { throw new
UnsupportedOperationException(); }

  @Override
  public void size(int s)  { throw new UnsupportedOperationException(); }

#if ! KEY_CLASS_Reference
  @Override
  public int compareTo(final List<? extends KEY_GENERIC_CLASS> o) {
   if (o == this) return 0;
   return ((List<?>)o).isEmpty() ? 0 : -1;
  }
#endif

  @Override
  public Object clone() { return EMPTY_LIST; }

  @Override
  public int hashCode() { return 1; }

  @Override
  @SuppressWarnings("rawtypes")
  public boolean equals(Object o) { return o instanceof List && ((List)o).isEmpty(); }

  @Override
  public String toString() { return "[]"; }

  private Object readResolve() { return EMPTY_LIST; }
 }

 /** An empty list (immutable). It is serializable and cloneable.
```

```
 */
 SUPPRESS_WARNINGS_KEY_RAWTYPES
 public static final EmptyList EMPTY_LIST = new EmptyList();

#if KEYS_REFERENCE
 /** Returns an empty list (immutable). It is serializable and cloneable.
  *
  * <p>This method provides a typesafe access to {@link #EMPTY_LIST}.
  * @return an empty list (immutable).
  */
 @SuppressWarnings("unchecked")
 public static KEY_GENERIC LIST KEY_GENERIC emptyList() {
  return EMPTY_LIST;
 }
#endif


 /** An immutable class representing a type-specific singleton list.
  *
  * <p>This class may be useful to implement your own in case you subclass
  * a type-specific list.
  */

 public static class Singleton KEY_GENERIC extends ABSTRACT_LIST KEY_GENERIC implements
RandomAccess, java.io.Serializable, Cloneable {

  private static final long serialVersionUID = -7046029254386353129L;

  private final KEY_GENERIC_TYPE element;

  protected Singleton(final KEY_GENERIC_TYPE element) { this.element = element; }

  @Override
  public KEY_GENERIC_TYPE GET_KEY(final int i) { if (i == 0) return element; throw new
IndexOutOfBoundsException(); }

  @Override
  public boolean REMOVE(KEY_TYPE k) { throw new UnsupportedOperationException(); }

  @Override
  public KEY_GENERIC_TYPE REMOVE_KEY(final int i) { throw new UnsupportedOperationException(); }

  @Override
  public boolean contains(final KEY_TYPE k) { return KEY_EQUALS(k, element); }

  /* Slightly optimized w.r.t. the one in ABSTRACT_SET. */

  @Override
```

```
  public KEY_TYPE[] TO_KEY_ARRAY() {
   KEY_TYPE a[] = new KEY_TYPE[1];
   a[0] = element;
   return a;
  }

  @Override
  public KEY_LIST_ITERATOR KEY_GENERIC listIterator() { return ITERATORS.singleton(element); }

  @Override
  public KEY_LIST_ITERATOR KEY_GENERIC iterator() { return listIterator(); }

  @Override
  public KEY_LIST_ITERATOR KEY_GENERIC listIterator(final int i) {
   if (i > 1 || i < 0) throw new  IndexOutOfBoundsException();
   final KEY_LIST_ITERATOR KEY_GENERIC l = listIterator();
   if (i == 1) l.NEXT_KEY();
   return l;
  }

  @Override
  SUPPRESS_WARNINGS_KEY_UNCHECKED
  public LIST KEY_GENERIC subList(final int from, final int to) {
   ensureIndex(from);
   ensureIndex(to);
   if (from > to) throw new IndexOutOfBoundsException("Start index (" + from + ") is greater than end index (" + to +
")");

   if (from != 0 || to != 1) return EMPTY_LIST;
   return this;
  }

  @Override
  public boolean addAll(int i, Collection<? extends KEY_GENERIC_CLASS> c) { throw new
UnsupportedOperationException(); }

  @Override
  public boolean addAll(final Collection<? extends KEY_GENERIC_CLASS> c) { throw new
UnsupportedOperationException(); }

  @Override
  public boolean removeAll(final Collection<?> c) { throw new UnsupportedOperationException(); }

  @Override
  public boolean retainAll(final Collection<?> c) { throw new UnsupportedOperationException(); }

#if KEYS_PRIMITIVE
```

```
 @Override
 public boolean addAll(LIST c) { throw new UnsupportedOperationException(); }

 @Override
 public boolean addAll(int i, LIST c) { throw new UnsupportedOperationException(); }

 @Override
 public boolean addAll(int i, COLLECTION c) { throw new UnsupportedOperationException(); }

 @Override
 public boolean addAll(final COLLECTION c) { throw new UnsupportedOperationException(); }

 @Override
 public boolean removeAll(final COLLECTION c) { throw new UnsupportedOperationException(); }

 @Override
 public boolean retainAll(final COLLECTION c) { throw new UnsupportedOperationException(); }

#endif
 @Override
 public int size() { return 1; }

 @Override
 public void size(final int size) { throw new UnsupportedOperationException(); }

 @Override
 public void clear() { throw new UnsupportedOperationException(); }

 @Override
 public Object clone() { return this; }
 }

 /** Returns a type-specific immutable list containing only the specified element. The returned list is serializable and cloneable.
  *
  * @param element the only element of the returned list.
  * @return a type-specific immutable list containing just {@code element}.
  */

 public static KEY_GENERIC LIST KEY_GENERIC singleton(final KEY_GENERIC_TYPE element) { return new Singleton KEY_GENERIC_DIAMOND(element); }

#if KEYS_PRIMITIVE

 /** Returns a type-specific immutable list containing only the specified element. The returned list is serializable and cloneable.
  *
  * @param element the only element of the returned list.
```

```
 * @return a type-specific immutable list containing just {@code element}.
 */

 public static KEY_GENERIC LIST KEY_GENERIC singleton(final Object element) { return new Singleton
KEY_GENERIC_DIAMOND(KEY_OBJ2TYPE(element)); }

#endif


 /** A synchronized wrapper class for lists. */

 public static class SynchronizedList KEY_GENERIC extends COLLECTIONS.SynchronizedCollection
KEY_GENERIC implements LIST KEY_GENERIC, java.io.Serializable {

  private static final long serialVersionUID = -7046029254386353129L;

  protected final LIST KEY_GENERIC list; // Due to the large number of methods that are not in COLLECTION,
this is worth caching.

  protected SynchronizedList(final LIST KEY_GENERIC l, final Object sync) {
   super(l, sync);
   this.list = l;
  }

  protected SynchronizedList(final LIST KEY_GENERIC l) {
   super(l);
   this.list = l;
  }

  @Override
  public KEY_GENERIC_TYPE GET_KEY(final int i) { synchronized(sync) { return list.GET_KEY(i); } }

  @Override
  public KEY_GENERIC_TYPE set(final int i, final KEY_GENERIC_TYPE k) { synchronized(sync) { return
list.set(i, k); } }

  @Override
  public void add(final int i, final KEY_GENERIC_TYPE k) { synchronized(sync) { list.add(i, k); } }

  @Override
  public KEY_GENERIC_TYPE REMOVE_KEY(final int i) { synchronized(sync) { return list.REMOVE_KEY(i); }
}

  @Override
  public int indexOf(final KEY_TYPE k) { synchronized(sync) { return list.indexOf(k); } }

  @Override
  public int lastIndexOf(final KEY_TYPE k) { synchronized(sync) { return list.lastIndexOf(k); } }
```

```
  @Override
  public boolean addAll(final int index, final Collection<? extends KEY_GENERIC_CLASS> c) {
synchronized(sync) { return list.addAll(index, c); } }

  @Override
  public void getElements(final int from, final KEY_TYPE a[], final int offset, final int length) { synchronized(sync)
{ list.getElements(from, a, offset, length); } }

  @Override
  public void removeElements(final int from, final int to) { synchronized(sync) { list.removeElements(from, to); } }

  @Override
  public void addElements(int index, final KEY_GENERIC_TYPE a[], int offset, int length) { synchronized(sync) {
list.addElements(index, a, offset, length); } }

  @Override
  public void addElements(int index, final KEY_GENERIC_TYPE a[]) { synchronized(sync) {
list.addElements(index, a); } }

  @Override
  public void size(final int size) { synchronized(sync) { list.size(size); } }

  @Override
  public KEY_LIST_ITERATOR KEY_GENERIC listIterator() { return list.listIterator(); }

  @Override
  public KEY_LIST_ITERATOR KEY_GENERIC iterator() { return listIterator(); }

  @Override
  public KEY_LIST_ITERATOR KEY_GENERIC listIterator(final int i) { return list.listIterator(i); }

  @Override
  public LIST KEY_GENERIC subList(final int from, final int to) { synchronized(sync) { return new
SynchronizedList KEY_GENERIC_DIAMOND(list.subList(from, to), sync); } }

  @Override
  public boolean equals(final Object o) { if (o == this) return true; synchronized(sync) { return collection.equals(o); }
}

  @Override
  public int hashCode() { synchronized(sync) { return collection.hashCode(); } }

#if ! KEY_CLASS_Reference
  @Override
  public int compareTo(final List<? extends KEY_GENERIC_CLASS> o) { synchronized(sync) { return
list.compareTo(o); } }
#endif
```

```
#if KEYS_PRIMITIVE
  @Override
  public boolean addAll(final int index, final COLLECTION c) { synchronized(sync) { return list.addAll(index, c); }
}

  @Override
  public boolean addAll(final int index, LIST l) { synchronized(sync) { return list.addAll(index, l); } }

  @Override
  public boolean addAll(LIST l) { synchronized(sync) { return list.addAll(l); } }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public KEY_GENERIC_CLASS get(final int i) { synchronized(sync) { return list.get(i); } }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public void add(final int i, KEY_GENERIC_CLASS k) { synchronized(sync) { list.add(i, k); } }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public KEY_GENERIC_CLASS set(final int index, KEY_GENERIC_CLASS k) { synchronized(sync) { return
list.set(index, k); } }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public KEY_GENERIC_CLASS remove(final int i) { synchronized(sync) { return list.remove(i); } }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public int indexOf(final Object o) { synchronized(sync) { return list.indexOf(o); } }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  public int lastIndexOf(final Object o) { synchronized(sync) { return list.lastIndexOf(o); } }
```

```
#endif

  private void writeObject(java.io.ObjectOutputStream s) throws java.io.IOException {
   synchronized(sync) { s.defaultWriteObject(); }
  }
 }


 /** A synchronized wrapper class for random-access lists. */

 public static class SynchronizedRandomAccessList KEY_GENERIC extends SynchronizedList KEY_GENERIC
implements RandomAccess, java.io.Serializable {
  private static final long serialVersionUID = 0L;

  protected SynchronizedRandomAccessList(final LIST KEY_GENERIC l, final Object sync) {
   super(l, sync);
  }

  protected SynchronizedRandomAccessList(final LIST KEY_GENERIC l) {
   super(l);
  }

  @Override
  public LIST KEY_GENERIC subList(final int from, final int to) { synchronized(sync) { return new
SynchronizedRandomAccessList KEY_GENERIC_DIAMOND(list.subList(from, to), sync); } }
 }


 /** Returns a synchronized type-specific list backed by the given type-specific list.
  *
  * @param l the list to be wrapped in a synchronized list.
  * @return a synchronized view of the specified list.
  * @see java.util.Collections#synchronizedList(List)
  */
 public static KEY_GENERIC LIST KEY_GENERIC synchronize(final LIST KEY_GENERIC l) {
  return l instanceof RandomAccess ? new SynchronizedRandomAccessList KEY_GENERIC_DIAMOND(l) : new
SynchronizedList KEY_GENERIC_DIAMOND(l);
 }

 /** Returns a synchronized type-specific list backed by the given type-specific list, using an assigned object to
synchronize.
  *
  * @param l the list to be wrapped in a synchronized list.
  * @param sync an object that will be used to synchronize the access to the list.
  * @return a synchronized view of the specified list.
  * @see java.util.Collections#synchronizedList(List)
  */
```

```
 public static KEY_GENERIC LIST KEY_GENERIC synchronize(final LIST KEY_GENERIC l, final Object sync)
{
  return l instanceof RandomAccess ? new SynchronizedRandomAccessList KEY_GENERIC_DIAMOND(l, sync) :
new SynchronizedList KEY_GENERIC_DIAMOND(l, sync);
 }



 /** An unmodifiable wrapper class for lists. */

 public static class UnmodifiableList KEY_GENERIC extends COLLECTIONS.UnmodifiableCollection
KEY_GENERIC implements LIST KEY_GENERIC, java.io.Serializable {

  private static final long serialVersionUID = -7046029254386353129L;

  protected final LIST KEY_GENERIC list; // Due to the large number of methods that are not in COLLECTION,
this is worth caching.

  protected UnmodifiableList(final LIST KEY_GENERIC l) {
   super(l);
   this.list = l;
  }

  @Override
  public KEY_GENERIC_TYPE GET_KEY(final int i) { return list.GET_KEY(i); }

  @Override
  public KEY_GENERIC_TYPE set(final int i, final KEY_GENERIC_TYPE k) { throw new
UnsupportedOperationException(); }

  @Override
  public void add(final int i, final KEY_GENERIC_TYPE k) { throw new UnsupportedOperationException(); }

  @Override
  public KEY_GENERIC_TYPE REMOVE_KEY(final int i) { throw new UnsupportedOperationException(); }

  @Override
  public int indexOf(final KEY_TYPE k) { return list.indexOf(k); }

  @Override
  public int lastIndexOf(final KEY_TYPE k) { return list.lastIndexOf(k); }

  @Override
  public boolean addAll(final int index, final Collection<? extends KEY_GENERIC_CLASS> c) { throw new
UnsupportedOperationException(); }

  @Override
  public void getElements(final int from, final KEY_TYPE a[], final int offset, final int length) {
```

```
list.getElements(from, a, offset, length); }

  @Override
  public void removeElements(final int from, final int to) { throw new UnsupportedOperationException(); }

  @Override
  public void addElements(int index, final KEY_GENERIC_TYPE a[], int offset, int length) { throw new
UnsupportedOperationException(); }

  @Override
  public void addElements(int index, final KEY_GENERIC_TYPE a[]) { throw new
UnsupportedOperationException(); }

  @Override
  public void size(final int size) { list.size(size); }

  @Override
  public KEY_LIST_ITERATOR KEY_GENERIC listIterator() { return
ITERATORS.unmodifiable(list.listIterator()); }

  @Override
  public KEY_LIST_ITERATOR KEY_GENERIC iterator() { return listIterator(); }

  @Override
  public KEY_LIST_ITERATOR KEY_GENERIC listIterator(final int i) { return
ITERATORS.unmodifiable(list.listIterator(i)); }

  @Override
  public LIST KEY_GENERIC subList(final int from, final int to) { return new UnmodifiableList
KEY_GENERIC_DIAMOND(list.subList(from, to)); }

  @Override
  public boolean equals(final Object o) { if (o == this) return true; return collection.equals(o); }

  @Override
  public int hashCode() { return collection.hashCode(); }

#if ! KEY_CLASS_Reference
  @Override
  public int compareTo(final List<? extends KEY_GENERIC_CLASS> o) { return list.compareTo(o); }
#endif

#if KEYS_PRIMITIVE
  @Override
  public boolean addAll(final int index, final COLLECTION c) { throw new UnsupportedOperationException(); }

  @Override
  public boolean addAll(final LIST l) { throw new UnsupportedOperationException(); }
```

```
 @Override
 public boolean addAll(final int index, final LIST l) { throw new UnsupportedOperationException(); }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 public KEY_GENERIC_CLASS get(final int i) { return list.get(i); }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 public void add(final int i, KEY_GENERIC_CLASS k) { throw new UnsupportedOperationException(); }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 public KEY_GENERIC_CLASS set(final int index, KEY_GENERIC_CLASS k) { throw new
UnsupportedOperationException(); }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 public KEY_GENERIC_CLASS remove(final int i) { throw new UnsupportedOperationException(); }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 public int indexOf(final Object o) { return list.indexOf(o); }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 public int lastIndexOf(final Object o) { return list.lastIndexOf(o); }
#endif
 }


 /** An unmodifiable wrapper class for random-access lists. */

 public static class UnmodifiableRandomAccessList KEY_GENERIC extends UnmodifiableList KEY_GENERIC
implements RandomAccess, java.io.Serializable {
```

```
 private static final long serialVersionUID = 0L;

 protected UnmodifiableRandomAccessList(final LIST KEY_GENERIC l) {
  super(l);
 }


 @Override
 public LIST KEY_GENERIC subList(final int from, final int to) { return new UnmodifiableRandomAccessList
KEY_GENERIC_DIAMOND(list.subList(from, to)); }
 }


 /** Returns an unmodifiable type-specific list backed by the given type-specific list.
  *
  * @param l the list to be wrapped in an unmodifiable list.
  * @return an unmodifiable view of the specified list.
  * @see java.util.Collections#unmodifiableList(List)
  */
 public static KEY_GENERIC LIST KEY_GENERIC unmodifiable(final LIST KEY_GENERIC l) {
  return l instanceof RandomAccess ? new UnmodifiableRandomAccessList KEY_GENERIC_DIAMOND(l) : new
UnmodifiableList KEY_GENERIC_DIAMOND(l);
 }




#ifdef TEST

 private static KEY_TYPE genKey() {
#if KEY_CLASS_Byte || KEY_CLASS_Short || KEY_CLASS_Character
  return (KEY_TYPE)(r.nextInt());
#elif KEYS_PRIMITIVE
  return r.NEXT_KEY();
#elif KEY_CLASS_Object
  return Integer.toBinaryString(r.nextInt());
#else
  return new java.io.Serializable() {};
#endif
 }


 private static void testLists(KEY_TYPE k, LIST m, List t, int level) {
  int n = 100;
  int c;

  long ms;
```

```
  boolean mThrowsIllegal, tThrowsIllegal, mThrowsNoElement, tThrowsNoElement, mThrowsIndex, tThrowsIndex,
mThrowsUnsupp, tThrowsUnsupp;
  boolean rt = false, rm = false;
  Object Rt = null, Rm = null;

  if (level == 0) return;

  /* Now we check that m and t are equal. */
  if (!m.equals(t) || ! t.equals(m)) System.err.println("m: " + m + " t: " + t);

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) at start");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) at start");

  /* Now we check that m actually holds that data. */
  for(java.util.Iterator i=t.iterator(); i.hasNext();) {
   ensure(m.contains(i.next()), "Error (" + level + ", " + seed + "): m and t differ on an entry after insertion (iterating on
t)");
  }

  /* Now we check that m actually holds that data, but iterating on m. */
  for(java.util.Iterator i=m.listIterator(); i.hasNext();) {
   ensure(t.contains(i.next()), "Error (" + level + ", " + seed + "): m and t differ on an entry after insertion (iterating on
m)");
  }

  /* Now we check that inquiries about random data give the same answer in m and t. For
     m we use the polymorphic method. */

  for(int i=0; i<n;  i++) {
   KEY_TYPE T = genKey();

   mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal =
mThrowsUnsupp = tThrowsUnsupp = false;

   try {
    m.contains(T);
   }
   catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
   catch (IllegalArgumentException e) { mThrowsIllegal = true; }
   catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }

   try {
    t.contains(KEY2OBJ(T));
   }
   catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
   catch (IllegalArgumentException e) { tThrowsIllegal = true; }
   catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
```

```
  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): contains() divergence in
java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ") " + m);
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): contains() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
  ensure(mThrowsIndex == tThrowsIndex, "Error (" + level + ", " + seed + "): contains() divergence in
IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
  if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex) ensure(m.contains(KEY2OBJ(T)) ==
t.contains(KEY2OBJ(T)), "Error (" + level + ", " + seed + "): divergence in keys between t and m (polymorphic
method) " + m);
 }

 /* Again, we check that inquiries about random data give the same answer in m and t, but
    for m we use the standard method. */

 for(int i=0; i<n;  i++) {
  KEY_TYPE T = genKey();

  mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal =
mThrowsUnsupp = tThrowsUnsupp = false;

  try {
   m.contains(KEY2OBJ(T));
  }
  catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }
  catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }

  try {
   t.contains(KEY2OBJ(T));
  }
  catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
  catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): contains() divergence in
java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ") " + m);
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): contains() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
  ensure(mThrowsIndex == tThrowsIndex, "Error (" + level + ", " + seed + "): contains() divergence in
IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
  ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + level + ", " + seed + "): contains() divergence in
UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);
  if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex && !mThrowsUnsupp)
ensure(m.contains(KEY2OBJ(T)) ==  t.contains(KEY2OBJ(T)), "Error (" + level + ", " + seed + "): divergence
between t and m (standard method) " + m);
 }
```

```
  /* Now we add and remove random data in m and t, checking that the result is the same. */

  for(int i=0; i<20*n;  i++) {
   KEY_TYPE T = genKey();

   mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal =
mThrowsUnsupp = tThrowsUnsupp = false;

   try {
    rm = m.add(KEY2OBJ(T));
   }
   catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
   catch (IllegalArgumentException e) { mThrowsIllegal = true; }
   catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }
   catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }

   try {
    rt = t.add(KEY2OBJ(T));
   }
   catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
   catch (IllegalArgumentException e) { tThrowsIllegal = true; }
   catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
   catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }

   ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): add() divergence in
java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ") " + m);
   ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): add() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
   ensure(mThrowsIndex == tThrowsIndex, "Error (" + level + ", " + seed + "): add() divergence in
IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
   ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + level + ", " + seed + "): add() divergence in
UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);
   if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex && !mThrowsUnsupp) ensure(rm == rt, "Error
(" + level + ", " + seed + "): divergence in add() between t and m " + m);

   T = genKey();

   mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal =
mThrowsUnsupp = tThrowsUnsupp = false;

   try {
    rm = m.remove(KEY2OBJ(T));
   }
   catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
   catch (IllegalArgumentException e) { mThrowsIllegal = true; }
   catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }
   catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }
```

```
  try {
   rt = t.remove(KEY2OBJ(T));
  }
  catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
  catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }

  if (! KEY_EQUALS(T, k) && mThrowsUnsupp && ! tThrowsUnsupp) mThrowsUnsupp = true; // Stupid bug in
Collections.singleton()

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): remove() divergence in
java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ") " + m);
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): remove() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
  ensure(mThrowsIndex == tThrowsIndex, "Error (" + level + ", " + seed + "): remove() divergence in
IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
  ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + level + ", " + seed + "): remove() divergence in
UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);
  if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex && !mThrowsUnsupp) ensure(rm == rt, "Error
(" + level + ", " + seed + "): divergence in remove() between t and m " + m);
 }

 ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after removal " + m);
 ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after removal " + m);

 /* Now we add and remove random data in m and t at specific positions, checking that the result is the same. */

 for(int i=0; i<20*n;  i++) {
  KEY_TYPE T = genKey();

  mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal =
mThrowsUnsupp = tThrowsUnsupp = false;

  int pos = r.nextInt(2);

  try {
   m.add(pos, KEY2OBJ(T));
  }
  catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }
  catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }

  try {
   t.add(pos, KEY2OBJ(T));
  }
```

```
  catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
  catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): add() at " + pos + "
divergence in java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement
+ ") " + m);
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): add() at " + pos + " divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
  ensure(mThrowsIndex == tThrowsIndex, "Error (" + level + ", " + seed + "): add() at " + pos + " divergence in
IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
  ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + level + ", " + seed + "): add() at " + pos + " divergence in
UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);

  T = genKey();

  mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal =
mThrowsUnsupp = tThrowsUnsupp = false;

  pos = r.nextInt(2);

  try {
   Rm = m.remove(pos);
  }
  catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }
  catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }

  try {
   Rt = t.remove(pos);
  }
  catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
  catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }


  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): remove() at " + pos + "
divergence in java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement
+ ") " + m);
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): remove() at " + pos + " divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
  ensure(mThrowsIndex == tThrowsIndex, "Error (" + level + ", " + seed + "): remove() at " + pos + " divergence in
IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
  ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + level + ", " + seed + "): remove() at " + pos + " divergence
in UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);
```

```
  if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex && !mThrowsUnsupp) ensure(Rm == Rt || Rm
!= null && Rm.equals(Rt), "Error (" + level + ", " + seed + "): divergence in remove() at " + pos + " between t and
m " + m);
  }

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after removal " + m);
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after removal " + m);

  /* Now we add and remove random collections in m and t, checking that the result is the same. */

  for(int i=0; i<20*n;  i++) {
   KEY_TYPE T = genKey();

   mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal =
mThrowsUnsupp = tThrowsUnsupp = false;

   try {
    rm = m.addAll(java.util.Collections.singleton(KEY2OBJ(T)));
   }
   catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
   catch (IllegalArgumentException e) { mThrowsIllegal = true; }
   catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }
   catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }

   try {
    rt = t.addAll(java.util.Collections.singleton(KEY2OBJ(T)));
   }
   catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
   catch (IllegalArgumentException e) { tThrowsIllegal = true; }
   catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
   catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }

   ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): addAll() divergence in
java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ") " + m);
   ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): addAll() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
   ensure(mThrowsIndex == tThrowsIndex, "Error (" + level + ", " + seed + "): addAll() divergence in
IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
   ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + level + ", " + seed + "): addAll() divergence in
UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);
   if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex && !mThrowsUnsupp) ensure(rm == rt, "Error
(" + level + ", " + seed + "): divergence in addAll() between t and m " + m);

   T = genKey();

   mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal =
mThrowsUnsupp = tThrowsUnsupp = false;
```

```
  try {
   rm = m.removeAll(java.util.Collections.singleton(KEY2OBJ(T)));
  }
  catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }
  catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }

  try {
   rt = t.removeAll(java.util.Collections.singleton(KEY2OBJ(T)));
  }
  catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
  catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }


  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): removeAll() divergence in
java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ") " + m);
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): removeAll() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
  ensure(mThrowsIndex == tThrowsIndex, "Error (" + level + ", " + seed + "): removeAll() divergence in
IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
  ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + level + ", " + seed + "): removeAll() divergence in
UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);
  if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex && !mThrowsUnsupp) ensure(rm == rt, "Error
(" + level + ", " + seed + "): divergence in removeAll() between t and m " + m);
  }

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after set removal " + m);
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after set removal " + m);

  /* Now we add random collections at specific positions in m and t, checking that the result is the same. */

  for(int i=0; i<20*n;  i++) {
   KEY_TYPE T = genKey();

  mThrowsIndex = tThrowsIndex = mThrowsNoElement = tThrowsNoElement = mThrowsIllegal = tThrowsIllegal =
mThrowsUnsupp = tThrowsUnsupp = false;

  int pos = r.nextInt(2);

  try {
   rm = m.addAll(pos, java.util.Collections.singleton(KEY2OBJ(T)));
  }
  catch (java.util.NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { mThrowsIndex = true; }
```

```
  catch (UnsupportedOperationException e) { mThrowsUnsupp = true; }

  try {
   rt = t.addAll(pos, java.util.Collections.singleton(KEY2OBJ(T)));
  }
  catch (java.util.NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }
  catch (IndexOutOfBoundsException e) { tThrowsIndex = true; }
  catch (UnsupportedOperationException e) { tThrowsUnsupp = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): addAll() at " + pos + "
divergence in java.util.NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement
+ ") " + m);
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): addAll() at " + pos + " divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ") " + m);
  ensure(mThrowsIndex == tThrowsIndex, "Error (" + level + ", " + seed + "): addAll() at " + pos + " divergence in
IndexOutOfBoundsException for " + T + " (" + mThrowsIndex + ", " + tThrowsIndex + ") " + m);
  ensure(mThrowsUnsupp == tThrowsUnsupp, "Error (" + level + ", " + seed + "): addAll() at " + pos + " divergence
in UnsupportedOperationException for " + T + " (" + mThrowsUnsupp + ", " + tThrowsUnsupp + ") " + m);
  if (!mThrowsNoElement && !mThrowsIllegal && !mThrowsIndex && !mThrowsUnsupp) ensure(rm == rt, "Error
(" + level + ", " + seed + "): divergence in addAll() at " + pos + " between t and m " + m);

  }

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after set removal " + m);
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after set removal " + m);

  /* Now we check that m actually holds the same data. */

  for(java.util.Iterator i=t.iterator(); i.hasNext();) {
   ensure(m.contains(i.next()), "Error (" + level + ", " + seed + "): m and t differ on an entry after removal (iterating on
t)");
  }

  /* Now we check that m actually holds that data, but iterating on m. */

  for(java.util.Iterator i=m.listIterator(); i.hasNext();) {
   ensure(t.contains(i.next()), "Error (" + level + ", " + seed + "): m and t differ on an entry after removal (iterating on
m)");
  }

  if (m instanceof Singleton) {
   ensure(m.equals(((Singleton)m).clone()), "Error (" + level + ", " + seed + "): m does not equal m.clone()");
   ensure(((Singleton)m).clone().equals(m), "Error (" + level + ", " + seed + "): m.clone() does not equal m");
  }

  int h = m.hashCode();
```

```
  /* Now we save and read m. */

  LIST m2 = null;

  try {
   java.io.File ff = new java.io.File("it.unimi.dsi.fastutil.test");
   java.io.OutputStream os = new java.io.FileOutputStream(ff);
   java.io.ObjectOutputStream oos = new java.io.ObjectOutputStream(os);

   oos.writeObject(m);
   oos.close();

   java.io.InputStream is = new java.io.FileInputStream(ff);
   java.io.ObjectInputStream ois = new java.io.ObjectInputStream(is);

   m2 = (LIST)ois.readObject();
   ois.close();
   ff.delete();
  }
  catch(Exception e) {
   e.printStackTrace();
   System.exit(1);
  }

#if ! KEY_CLASS_Reference

  ensure(m2.hashCode() == h, "Error (" + level + ", " + seed + "): hashCode() changed after save/read");

  /* Now we check that m2 actually holds that data. */

  ensure(m2.equals(t), "Error (" + level + ", " + seed + "): ! m2.equals(t) after save/read");
  ensure(t.equals(m2), "Error (" + level + ", " + seed + "): ! t.equals(m2) after save/read");
#endif

  if (! m.isEmpty()) {
   int start = r.nextInt(m.size());
   int end = start + r.nextInt(m.size() - start);
   //System.err.println("Checking subList from " + start + " to " + end + " (level=" + (level+1) + ")...");
   testLists(k, m.subList(start, end), t.subList(start, end), level - 1);

   ensure(m.equals(t), "Error (" + level + ", " + seed + m + t + "): ! m.equals(t) after subList");
   ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after subList");

  }

  return;
 }
```

```
 private static void test() {
  KEY_TYPE k = genKey();
  LIST m = new Singleton(k);
  List u = java.util.Collections.singletonList(KEY2OBJ(k));
  testLists(k, m, java.util.Collections.unmodifiableList(u), 3);
  System.out.println("Test OK");
 }

 private static long seed = System.currentTimeMillis();
 private static java.util.Random r = new java.util.Random(seed);

 private static java.text.NumberFormat format = new java.text.DecimalFormat("#,###.00");
 private static java.text.FieldPosition fp = new java.text.FieldPosition(0);

 private static String format(double d) {
  StringBuffer s = new StringBuffer();
  return format.format(d, s, fp).toString();
 }

 private static void fatal(String msg) {
  System.out.println(msg);
  System.exit(1);
 }

 private static void ensure(boolean cond, String msg) {
  if (cond) return;
  fatal(msg);
 }

 /** This method expects as first argument a lower-cased type (e.g., "int"),
  * and as second optional argument a seed. */

 public static void main(String arg[]) throws Exception {
  if (arg.length > 1) r = new java.util.Random(seed = Long.parseLong(arg[1]));

  try {
   test();
  } catch(Throwable e) {
   e.printStackTrace(System.err);
   System.err.println("seed: " + seed);
  }
 }

#endif

}
```

Found in path(s):

* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/Lists.drv
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2005-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*   http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/io/FastBufferedInputStream.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/io/FastMultiByteArrayInputStream.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/io/MeasurableOutputStream.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/io/MeasurableStream.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/io/FastByteArrayInputStream.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/TextIO.drv
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/io/MeasurableInputStream.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/io/RepositionableStream.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/BinIO.drv
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/io/InspectableFileCachedInputStream.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/io/FastByteArrayOutputStream.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/io/FastBufferedOutputStream.java
No license file was found, but licenses were detected in source scan.

<project>

```
 <modelVersion>4.0.0</modelVersion>
 <groupId>it.unimi.dsi</groupId>
 <artifactId>fastutil</artifactId>
 <packaging>jar</packaging>
 <name>fastutil</name>
 <version>VERSION</version>
 <description>fastutil extends the Java Collections Framework by providing type-specific maps, sets, lists and
priority queues with a small memory footprint and fast access and insertion; provides also big (64-bit) arrays, sets
and lists, and fast, practical I/O classes for binary and text files.</description>
 <url>http://fastutil.di.unimi.it/</url>
 <licenses>
  <license>
   <name>Apache License, Version 2.0</name>
   <url>http://www.apache.org/licenses/LICENSE-2.0.html</url>
   <distribution>repo</distribution>
  </license>
 </licenses>
 <scm>
  <connection>scm:git://github.com/vigna/fastutil.git</connection>
  <url>https://github.com/vigna/fastutil</url>
 </scm>
 <developers>
  <developer>
   <id>vigna</id>
   <name>Sebastiano Vigna</name>
   <email>vigna@di.unimi.it</email>
  </developer>
 </developers>
 <properties>
  <maven.compiler.source>1.8</maven.compiler.source>
  <maven.compiler.target>1.8</maven.compiler.target>
 </properties>
 <dependencies>
  <dependency>
   <groupId>junit</groupId>
   <artifactId>junit</artifactId>
   <version>4.12</version>
   <scope>test</scope>
  </dependency>
 </dependencies>
</project>
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/pom-model.xml
No license file was found, but licenses were detected in source scan.

```
/*
```

```
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

package PACKAGE;

import it.unimi.dsi.fastutil.objects.Object2IntOpenHashMap;

import java.io.Serializable;
import java.util.concurrent.locks.ReentrantReadWriteLock;
import java.util.concurrent.locks.ReentrantReadWriteLock.ReadLock;
import java.util.concurrent.locks.ReentrantReadWriteLock.WriteLock;

/** A concurrent counting map. The map is made by a number of <em>stripes</em> (instances of {@link
Object2IntOpenHashMap})
* which are accessed independently
* using a {@link ReentrantReadWriteLock}. Only one thread can write in a stripe at a time, but different stripes
* can be modified independently and read access can happen concurrently on each stripe.
*
* @param <K> the type of keys.
*/

public class STRIPED_OPEN_HASH_MAP KEY_VALUE_GENERIC extends ABSTRACT_MAP
KEY_VALUE_GENERIC implements java.io.Serializable, Cloneable {
 private static final long serialVersionUID = 1L;

 /** The stripes. Keys are distributed among them using the lower bits of their {@link Object#hashCode()}. */
 private final OPEN_HASH_MAP KEY_VALUE_GENERIC[] map;
 /** An array of locks parallel to {@link #map}, protecting each stripe. */
 private final transient ReentrantReadWriteLock[] lock;
 /** {@link #map map.length} &minus; 1, cached. */
 private final int mask;

 /** Creates a new concurrent counting map with concurrency level equal to {@link
Runtime#availableProcessors()}. */
 public STRIPED_OPEN_HASH_MAP() {
  this(Runtime.getRuntime().availableProcessors());
```

```
 }

 /** Creates a new concurrent counting map.
  *
  * @param concurrencyLevel the number of stripes (it will be {@linkplain Integer#highestOneBit(int) forced to be a power of two}); ideally, as large as the number of threads that will ever access
  * this map, but higher values require more space.
  */
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public STRIPED_OPEN_HASH_MAP(final int concurrencyLevel) {
  map = new OPEN_HASH_MAP[Integer.highestOneBit(concurrencyLevel)];
  lock = new ReentrantReadWriteLock[map.length];
  for(int i = map.length; i-- != 0;) {
   map[i] = new OPEN_HASH_MAP KEY_VALUE_GENERIC_DIAMOND();
   lock[i] = new ReentrantReadWriteLock();
  }
  mask = map.length - 1;
 }

#if KEYS_PRIMITIVE

 public VALUE_GENERIC_CLASS get(final KEY_CLASS k) {
  final int stripe = KEY2INTHASH(k) & mask;
  final ReadLock readLock = lock[stripe].readLock();
  try {
   readLock.lock();
   return map[stripe].get(k);
  }
  finally {
   readLock.unlock();
  }
 }

#endif

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public VALUE_GENERIC_TYPE GET_VALUE(final KEY_TYPE k) {
  final int stripe = KEY2INTHASH(k) & mask;
  final ReadLock readLock = lock[stripe].readLock();
  try {
   readLock.lock();
   return map[stripe].GET_VALUE(k);
  }
  finally {
   readLock.unlock();
  }
 }
```

```
 public VALUE_GENERIC_TYPE put(final KEY_GENERIC_TYPE k, final VALUE_GENERIC_TYPE v) {
  final int stripe = KEY2INTHASH(k) & mask;
  final WriteLock writeLock = lock[stripe].writeLock();
  try {
   writeLock.lock();
   return map[stripe].put(k, v);
  }
  finally {
   writeLock.unlock();
  }
 }

 public VALUE_GENERIC_TYPE putIfAbsent(final KEY_GENERIC_TYPE k, final VALUE_GENERIC_TYPE
v) {
  final int stripe = KEY2INTHASH(k) & mask;
  final WriteLock writeLock = lock[stripe].writeLock();
  try {
   writeLock.lock();
   if (map[stripe].containsKey(k)) return map[stripe].get(k);
   return map[stripe].put(k, v);
  }
  finally {
   writeLock.unlock();
  }
 }

#if VALUES_PRIMITIVE || KEYS_PRIMITIVE

 public VALUE_GENERIC_CLASS put(final KEY_GENERIC_CLASS ok, final VALUE_GENERIC_CLASS ov)
{
  final int stripe = KEY2INTHASH(ok) & mask;
  final WriteLock writeLock = lock[stripe].writeLock();
  try {
   writeLock.lock();
   return map[stripe].put(ok, ov);
  }
  finally {
   writeLock.unlock();
  }
 }


 public VALUE_GENERIC_CLASS putIfAbsent(final KEY_GENERIC_CLASS ok, final
VALUE_GENERIC_CLASS ov) {
  final int stripe = KEY2INTHASH(ok) & mask;
  final WriteLock writeLock = lock[stripe].writeLock();
  try {
   writeLock.lock();
```

```
   if (map[stripe].containsKey(ok)) return map[stripe].get(ok);
   return map[stripe].put(ok, ov);
  }
  finally {
   writeLock.unlock();
  }
 }

#endif

 public int size() {
  int size = 0;
  for(int stripe = lock.length; stripe-- != 0;) {
   final ReadLock readLock = lock[stripe].readLock();
   try {
    readLock.lock();
    size += map[stripe].size();
   }
   finally {
    readLock.unlock();
   }
  }

  return size;
 }

 public FastEntrySet KEY_VALUE_GENERIC ENTRYSET() {
  throw new UnsupportedOperationException();
 }
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/StripedOpenHashMap.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
```

```
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import java.util.Collection;

/** A type-specific {@link Collection}; provides some additional methods
* that use polymorphism to avoid (un)boxing.
*
* <p>Additionally, this class defines strengthens (again) {@link #iterator()}.
*
* @see Collection
*/

public interface COLLECTION KEY_GENERIC extends Collection<KEY_GENERIC_CLASS>,
KEY_ITERABLE KEY_GENERIC {

 /** Returns a type-specific iterator on the elements of this collection.
  *
  * <p>Note that this specification strengthens the one given in
  * {@link java.lang.Iterable#iterator()}, which was already
  * strengthened in the corresponding type-specific class,
  * but was weakened by the fact that this interface extends {@link Collection}.
  *
  * @return a type-specific iterator on the elements of this collection.
  */
 @Override
 KEY_ITERATOR KEY_GENERIC iterator();

#if KEYS_PRIMITIVE

 /** Ensures that this collection contains the specified element (optional operation).
  * @see Collection#add(Object)
  */
 boolean add(KEY_TYPE key);

 /** Returns {@code true} if this collection contains the specified element.
  * @see Collection#contains(Object)
  */
 boolean contains(KEY_TYPE key);

 /** Removes a single instance of the specified element from this
  * collection, if it is present (optional operation).
  *
  * <p>Note that this method should be called {@link java.util.Collection#remove(Object) remove()}, but the clash
```

```
 * with the similarly named index-based method in the {@link java.util.List} interface
 * forces us to use a distinguished name. For simplicity, the set interfaces reinstates
 * {@code remove()}.
 *
 * @see Collection#remove(Object)
 */
boolean rem(KEY_TYPE key);

/** {@inheritDoc}
 * @deprecated Please use the corresponding type-specific method instead.
 */
@Deprecated
@Override
default boolean add(final KEY_GENERIC_CLASS key) {
 return add(KEY_CLASS2TYPE(key));
}

/** {@inheritDoc}
 * @deprecated Please use the corresponding type-specific method instead.
 */
@Deprecated
@Override
default boolean contains(final Object key) {
 if (key == null) return false;
 return contains(KEY_OBJ2TYPE(key));
}

/** {@inheritDoc}
 * @deprecated Please use (and implement) the {@code rem()} method instead.
 */
@Deprecated
@Override
default boolean remove(final Object key) {
 if (key == null) return false;
 return rem(KEY_OBJ2TYPE(key));
}

/** Returns a primitive type array containing the items of this collection.
 * @return a primitive type array containing the items of this collection.
 * @see Collection#toArray()
 */
KEY_TYPE[] TO_KEY_ARRAY();

/** Returns a primitive type array containing the items of this collection.
 *
 * <p>Note that, contrarily to {@link Collection#toArray(Object[])}, this
 * methods just writes all elements of this collection: no special
 * value will be added after the last one.
```

```
 *
 * @param a if this array is big enough, it will be used to store this collection.
 * @return a primitive type array containing the items of this collection.
 * @see Collection#toArray(Object[])
 * @deprecated Please use {@code toArray()} instead&amp;mdash;this method is redundant and will be removed in the future.
 */
@Deprecated
KEY_TYPE[] TO_KEY_ARRAY(KEY_TYPE a[]);

/** Returns an array containing all of the elements in this collection; the runtime type of the returned array is that of the specified array.
 *
 * <p>Note that, contrarily to {@link Collection#toArray(Object[])}, this
 * methods just writes all elements of this collection: no special
 * value will be added after the last one.
 *
 * @param a if this array is big enough, it will be used to store this collection.
 * @return a primitive type array containing the items of this collection.
 * @see Collection#toArray(Object[])
 */
KEY_TYPE[] toArray(KEY_TYPE a[]);

/** Adds all elements of the given type-specific collection to this collection.
 *
 * @param c a type-specific collection.
 * @see Collection#addAll(Collection)
 * @return {@code true} if this collection changed as a result of the call.
 */
boolean addAll(COLLECTION c);

/** Checks whether this collection contains all elements from the given type-specific collection.
 *
 * @param c a type-specific collection.
 * @see Collection#containsAll(Collection)
 * @return {@code true} if this collection contains all elements of the argument.
 */
boolean containsAll(COLLECTION c);

/** Remove from this collection all elements in the given type-specific collection.
 *
 * @param c a type-specific collection.
 * @see Collection#removeAll(Collection)
 * @return {@code true} if this collection changed as a result of the call.
 */
boolean removeAll(COLLECTION c);

#ifdef JDK_PRIMITIVE_PREDICATE
```

```
 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead.
  */
 @Deprecated
 @Override
 default boolean removeIf(final java.util.function.Predicate<? super KEY_GENERIC_CLASS> filter) {
  return removeIf((JDK_PRIMITIVE_PREDICATE) key -> filter.test(KEY2OBJ(KEY_NARROWING(key))));
 }

 /** Remove from this collection all elements which satisfy the given predicate.
  *
  * @param filter a predicate which returns {@code true} for elements to be
  *        removed.
  * @see Collection#removeIf(java.util.function.Predicate)
  * @return {@code true} if any elements were removed.
  */
 @SuppressWarnings("overloads")
 default boolean removeIf(final JDK_PRIMITIVE_PREDICATE filter) {
  java.util.Objects.requireNonNull(filter);
  boolean removed = false;
  final KEY_ITERATOR each = iterator();
  while (each.hasNext()) {
   if (filter.test(each.NEXT_KEY())) {
    each.remove();
    removed = true;
   }
  }
  return removed;
 }
#endif

 /** Retains in this collection only elements from the given type-specific collection.
  *
  * @param c a type-specific collection.
  * @see Collection#retainAll(Collection)
  * @return {@code true} if this collection changed as a result of the call.
  */
 boolean retainAll(COLLECTION c);

#endif

}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/Collection.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import it.unimi.dsi.fastutil.objects.AbstractObjectSortedSet;
import it.unimi.dsi.fastutil.objects.ObjectBidirectionalIterator;
import it.unimi.dsi.fastutil.objects.ObjectListIterator;
import it.unimi.dsi.fastutil.objects.ObjectSortedSet;

import VALUE_PACKAGE.VALUE_COLLECTION;
import VALUE_PACKAGE.VALUE_ABSTRACT_COLLECTION;
import VALUE_PACKAGE.VALUE_ITERATOR;

import java.util.Comparator;
import java.util.Iterator;
import java.util.Map;
import java.util.SortedMap;
import java.util.NoSuchElementException;

#if VALUES_PRIMITIVE
import VALUE_PACKAGE.VALUE_LIST_ITERATOR;
#endif

/** A type-specific red-black tree map with a fast, small-footprint implementation.
*
* <p>The iterators provided by the views of this class are type-specific {@linkplain
* it.unimi.dsi.fastutil.BidirectionalIterator bidirectional iterators}.
* Moreover, the iterator returned by {@code iterator()} can be safely cast
* to a type-specific {@linkplain java.util.ListIterator list iterator}.
*
*/

public class RB_TREE_MAP KEY_VALUE_GENERIC extends ABSTRACT_SORTED_MAP
```

```
KEY_VALUE_GENERIC implements java.io.Serializable, Cloneable {

/** A reference to the root entry. */
protected transient Entry KEY_VALUE_GENERIC tree;

/** Number of entries in this map. */
protected int count;

/** The first key in this map. */
protected transient Entry KEY_VALUE_GENERIC firstEntry;

/** The last key in this map. */
protected transient Entry KEY_VALUE_GENERIC lastEntry;

/** Cached set of entries. */
protected transient ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> entries;

/** Cached set of keys. */
protected transient SORTED_SET KEY_GENERIC keys;

/** Cached collection of values. */
protected transient VALUE_COLLECTION VALUE_GENERIC values;

/** The value of this variable remembers, after a {@code put()}
 * or a {@code remove()}, whether the <em>domain</em> of the map
 * has been modified. */
protected transient boolean modified;

/** This map's comparator, as provided in the constructor. */
protected Comparator<? super KEY_GENERIC_CLASS> storedComparator;

/** This map's actual comparator; it may differ from {@link #storedComparator} because it is
 always a type-specific comparator, so it could be derived from the former by wrapping. */
protected transient KEY_COMPARATOR KEY_SUPER_GENERIC actualComparator;

private static final long serialVersionUID = -7046029254386353129L;

{
 allocatePaths();
}

/** Creates a new empty tree map.
 */

public RB_TREE_MAP() {
 tree = null;
 count = 0;
}
```

```
 /** Generates the comparator that will be actually used.
  *
  * <p>When a given {@link Comparator} is specified and stored in {@link
  * #storedComparator}, we must check whether it is type-specific.  If it is
  * so, we can used directly, and we store it in {@link #actualComparator}. Otherwise,
  * we adapt it using a helper static method.
  */
 private void setActualComparator() {
#if KEY_CLASS_Object
  actualComparator = storedComparator;
#else
  actualComparator = COMPARATORS.AS_KEY_COMPARATOR(storedComparator);
#endif
 }


 /** Creates a new empty tree map with the given comparator.
  *
  * @param c a (possibly type-specific) comparator.
  */

 public RB_TREE_MAP(final Comparator<? super KEY_GENERIC_CLASS> c) {
  this();
  storedComparator = c;
  setActualComparator();
 }


 /** Creates a new tree map copying a given map.
  *
  * @param m a {@link Map} to be copied into the new tree map.
  */

 public RB_TREE_MAP(final Map<? extends KEY_GENERIC_CLASS, ? extends VALUE_GENERIC_CLASS>
m) {
  this();
  putAll(m);
 }

 /** Creates a new tree map copying a given sorted map (and its {@link Comparator}).
  *
  * @param m a {@link SortedMap} to be copied into the new tree map.
  */

 public RB_TREE_MAP(final SortedMap<KEY_GENERIC_CLASS,VALUE_GENERIC_CLASS> m) {
  this(m.comparator());
  putAll(m);
```

```
}

/** Creates a new tree map copying a given map.
 *
 * @param m a type-specific map to be copied into the new tree map.
 */

public RB_TREE_MAP(final MAP KEY_VALUE_EXTENDS_GENERIC m) {
 this();
 putAll(m);
}

/** Creates a new tree map copying a given sorted map (and its {@link Comparator}).
 *
 * @param m a type-specific sorted map to be copied into the new tree map.
 */

public RB_TREE_MAP(final SORTED_MAP KEY_VALUE_GENERIC m) {
 this(m.comparator());
 putAll(m);
}

/** Creates a new tree map using the elements of two parallel arrays and the given comparator.
 *
 * @param k the array of keys of the new tree map.
 * @param v the array of corresponding values in the new tree map.
 * @param c a (possibly type-specific) comparator.
 * @throws IllegalArgumentException if {@code k} and {@code v} have different lengths.
 */

public RB_TREE_MAP(final KEY_GENERIC_TYPE[] k, final VALUE_GENERIC_TYPE v[], final
Comparator<? super KEY_GENERIC_CLASS> c) {
 this(c);
 if (k.length != v.length) throw new IllegalArgumentException("The key array and the value array have different
lengths (" + k.length + " and " + v.length + ")");
 for(int i = 0; i < k.length; i++) this.put(k[i], v[i]);
}

/** Creates a new tree map using the elements of two parallel arrays.
 *
 * @param k the array of keys of the new tree map.
 * @param v the array of corresponding values in the new tree map.
 * @throws IllegalArgumentException if {@code k} and {@code v} have different lengths.
 */

public RB_TREE_MAP(final KEY_GENERIC_TYPE[] k, final VALUE_GENERIC_TYPE v[]) {
 this(k, v, null);
}
```

```
/*
 * The following methods implements some basic building blocks used by
 * all accessors.  They are (and should be maintained) identical to those used in RBTreeSet.drv.
 *
 * The put()/remove() code is derived from Ben Pfaff's GNU libavl
 * (http://www.msu.edu/~pfaffben/avl/). If you want to understand what's
 * going on, you should have a look at the literate code contained therein
 * first.
 */


/** Compares two keys in the right way.
 *
 * <p>This method uses the {@link #actualComparator} if it is non-{@code null}.
 * Otherwise, it resorts to primitive type comparisons or to {@link Comparable#compareTo(Object) compareTo()}.
 *
 * @param k1 the first key.
 * @param k2 the second key.
 * @return a number smaller than, equal to or greater than 0, as usual
 * (i.e., when k1 &lt; k2, k1 = k2 or k1 &gt; k2, respectively).
 */

SUPPRESS_WARNINGS_KEY_UNCHECKED
final int compare(final KEY_GENERIC_TYPE k1, final KEY_GENERIC_TYPE k2) {
 return actualComparator == null ? KEY_CMP(k1, k2) : actualComparator.compare(k1, k2);
}

/** Returns the entry corresponding to the given key, if it is in the tree; {@code null}, otherwise.
 *
 * @param k the key to search for.
 * @return the corresponding entry, or {@code null} if no entry with the given key exists.
 */

final Entry KEY_VALUE_GENERIC findKey(final KEY_GENERIC_TYPE k) {
 Entry KEY_VALUE_GENERIC e = tree;
 int cmp;

 while (e != null && (cmp = compare(k, e.key)) != 0) e = cmp < 0 ? e.left() : e.right();

 return e;
}

/** Locates a key.
 *
 * @param k a key.
 * @return the last entry on a search for the given key; this will be
 * the given key, if it present; otherwise, it will be either the smallest greater key or the greatest smaller key.
```

```
 */

 final Entry KEY_VALUE_GENERIC locateKey(final KEY_GENERIC_TYPE k) {
  Entry KEY_VALUE_GENERIC e = tree, last = tree;
  int cmp = 0;

  while (e != null && (cmp = compare(k, e.key)) != 0) {
   last = e;
   e = cmp < 0 ? e.left() : e.right();
  }

  return cmp == 0 ? e : last;
 }

 /** This vector remembers the path and the direction followed during the
  *  current insertion. It suffices for about 2<sup>32</sup> entries. */
 private transient boolean dirPath[];
 private transient Entry KEY_VALUE_GENERIC nodePath[];

 SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED_RAWTYPES
 private void allocatePaths() {
  dirPath = new boolean[64];
  nodePath = new Entry[64];
 }

#if VALUES_PRIMITIVE && !VALUE_CLASS_Boolean
 /** Adds an increment to value currently associated with a key.
  *
  * <p>Note that this method respects the {@linkplain #defaultReturnValue() default return value} semantics: when
  * called with a key that does not currently appears in the map, the key
  * will be associated with the default return value plus
  * the given increment.
  *
  * @param k the key.
  * @param incr the increment.
  * @return the old value, or the {@linkplain #defaultReturnValue() default return value} if no value was present for
the given key.
  */
 public VALUE_GENERIC_TYPE addTo(final KEY_GENERIC_TYPE k, final VALUE_GENERIC_TYPE incr) {
  Entry KEY_VALUE_GENERIC e = add(k);
  final VALUE_GENERIC_TYPE oldValue = e.value;
  e.value += incr;
  return oldValue;
 }
#endif

 @Override
 public VALUE_GENERIC_TYPE put(final KEY_GENERIC_TYPE k, final VALUE_GENERIC_TYPE v) {
```

```
 Entry KEY_VALUE_GENERIC e = add(k);
 final VALUE_GENERIC_TYPE oldValue = e.value;
 e.value = v;
 return oldValue;
}

/** Returns a node with key k in the balanced tree, creating one with defRetValue if necessary.
 *
 * @param k the key
 * @return a node with key k. If a node with key k already exists, then that node is returned,
 * otherwise a new node with defRetValue is created ensuring that the tree is balanced
 * after creation of the node.
 */
private Entry KEY_VALUE_GENERIC add(final KEY_GENERIC_TYPE k) {
 /* After execution of this method, modified is true iff a new entry has been inserted. */
 modified = false;
 int maxDepth = 0;

 Entry KEY_VALUE_GENERIC e;

 if (tree == null) { // The case of the empty tree is treated separately.
  count++;
  e = tree = lastEntry = firstEntry = new Entry KEY_VALUE_GENERIC_DIAMOND(k, defRetValue);
 }
 else {
  Entry KEY_VALUE_GENERIC p = tree;
  int cmp, i = 0;

  while(true) {
   if ((cmp = compare(k, p.key)) == 0) {
    // We clean up the node path, or we could have stale references later.
    while(i-- != 0) nodePath[i] = null;
    return p;
   }

   nodePath[i] = p;

   if (dirPath[i++] = cmp > 0) {
    if (p.succ()) {
     count++;
     e = new Entry KEY_VALUE_GENERIC_DIAMOND(k, defRetValue);

     if (p.right == null) lastEntry = e;

     e.left = p;
     e.right = p.right;

     p.right(e);
```

```
  break;
  }

  p = p.right;
 }
 else {
  if (p.pred()) {
   count++;
   e = new Entry KEY_VALUE_GENERIC_DIAMOND(k, defRetValue);

   if (p.left == null) firstEntry = e;

   e.right = p;
   e.left = p.left;

   p.left(e);

   break;
  }

  p = p.left;
 }
}

modified = true;
maxDepth = i--;

while(i > 0 && ! nodePath[i].black()) {
 if (! dirPath[i - 1]) {
  Entry KEY_VALUE_GENERIC y = nodePath[i - 1].right;

  if (! nodePath[i - 1].succ() && ! y.black()) {
   nodePath[i].black(true);
   y.black(true);
   nodePath[i - 1].black(false);
   i -= 2;
  }
  else {
   Entry KEY_VALUE_GENERIC x;

   if (! dirPath[i]) y = nodePath[i];
   else {
    x = nodePath[i];
    y = x.right;
    x.right = y.left;
    y.left = x;
    nodePath[i - 1].left = y;
```

```
  if (y.pred()) {
   y.pred(false);
   x.succ(y);
  }
 }

 x = nodePath[i - 1];
 x.black(false);
 y.black(true);

 x.left = y.right;
 y.right = x;
 if (i < 2) tree = y;
 else {
  if (dirPath[i - 2]) nodePath[i - 2].right = y;
  else nodePath[i - 2].left = y;
 }

 if (y.succ()) {
  y.succ(false);
  x.pred(y);
 }
 break;
 }
}
else {
 Entry KEY_VALUE_GENERIC y = nodePath[i - 1].left;

 if (! nodePath[i - 1].pred() && ! y.black()) {
  nodePath[i].black(true);
  y.black(true);
  nodePath[i - 1].black(false);
  i -= 2;
 }
 else {
  Entry KEY_VALUE_GENERIC x;

  if (dirPath[i]) y = nodePath[i];
  else {
   x = nodePath[i];
   y = x.left;
   x.left = y.right;
   y.right = x;
   nodePath[i - 1].right = y;

   if (y.succ()) {
    y.succ(false);
```

```
      x.pred(y);
     }

    }

    x = nodePath[i - 1];
    x.black(false);
    y.black(true);

    x.right = y.left;
    y.left = x;
    if (i < 2) tree = y;
    else {
     if (dirPath[i - 2]) nodePath[i - 2].right = y;
     else nodePath[i - 2].left = y;
    }

    if (y.pred()){
     y.pred(false);
     x.succ(y);
    }

    break;
   }
  }
 }
}
tree.black(true);
// We clean up the node path, or we could have stale references later.
while(maxDepth-- != 0) nodePath[maxDepth] = null;
return e;
}


/* After execution of this method, {@link #modified} is true iff an entry
has been deleted. */

SUPPRESS_WARNINGS_KEY_UNCHECKED
@Override
public VALUE_GENERIC_TYPE REMOVE_VALUE(final KEY_TYPE k) {
 modified = false;

 if (tree == null) return defRetValue;

 Entry KEY_VALUE_GENERIC p = tree;
 int cmp;
 int i = 0;
 final KEY_GENERIC_TYPE kk = KEY_GENERIC_CAST k;
```

```
while(true) {
 if ((cmp = compare(kk, p.key)) == 0) break;

 dirPath[i] = cmp > 0;
 nodePath[i] = p;

 if (dirPath[i++]) {
  if ((p = p.right()) == null) {
   // We clean up the node path, or we could have stale references later.
   while(i-- != 0) nodePath[i] = null;
   return defRetValue;
  }
 }
 else {
  if ((p = p.left()) == null) {
   // We clean up the node path, or we could have stale references later.
   while(i-- != 0) nodePath[i] = null;
   return defRetValue;
  }
 }

}

if (p.left == null) firstEntry = p.next();
if (p.right == null) lastEntry = p.prev();

if (p.succ()) {
 if (p.pred()) {
  if (i == 0) tree = p.left;
  else {
   if (dirPath[i - 1]) nodePath[i - 1].succ(p.right);
   else nodePath[i - 1].pred(p.left);
  }
 }
 else {
  p.prev().right = p.right;

  if (i == 0) tree = p.left;
  else {
   if (dirPath[i - 1]) nodePath[i - 1].right = p.left;
   else nodePath[i - 1].left = p.left;
  }
 }
}
else {
 boolean color;
 Entry KEY_VALUE_GENERIC r = p.right;
```

```
if (r.pred()) {
 r.left = p.left;
 r.pred(p.pred());
 if (! r.pred()) r.prev().right = r;
 if (i == 0) tree = r;
 else {
  if (dirPath[i - 1]) nodePath[i - 1].right = r;
  else nodePath[i - 1].left = r;
 }

 color = r.black();
 r.black(p.black());
 p.black(color);
 dirPath[i] = true;
 nodePath[i++] = r;
}
else {
 Entry KEY_VALUE_GENERIC s;
 int j = i++;

 while(true) {
  dirPath[i] = false;
  nodePath[i++] = r;
  s = r.left;
  if (s.pred()) break;
  r = s;
 }

 dirPath[j] = true;
 nodePath[j] = s;

 if (s.succ()) r.pred(s);
 else r.left = s.right;

 s.left = p.left;

 if (! p.pred()) {
  p.prev().right = s;
  s.pred(false);
 }

 s.right(p.right);

 color = s.black();
 s.black(p.black());
 p.black(color);
```

```
  if (j == 0) tree = s;
  else {
   if (dirPath[j - 1]) nodePath[j - 1].right = s;
   else nodePath[j - 1].left = s;
  }
 }
}

int maxDepth = i;

if (p.black()) {
 for(; i > 0; i--) {
  if (dirPath[i - 1] && ! nodePath[i - 1].succ() ||
    ! dirPath[i - 1] && ! nodePath[i - 1].pred()) {
   Entry KEY_VALUE_GENERIC x = dirPath[i - 1] ? nodePath[i - 1].right : nodePath[i - 1].left;

   if (! x.black()) {
    x.black(true);
    break;
   }
  }

  if (! dirPath[i - 1]) {
   Entry KEY_VALUE_GENERIC w = nodePath[i - 1].right;

   if (! w.black()) {
    w.black(true);
    nodePath[i - 1].black(false);

    nodePath[i - 1].right = w.left;
    w.left = nodePath[i - 1];

    if (i < 2) tree = w;
    else {
     if (dirPath[i - 2]) nodePath[i - 2].right = w;
     else nodePath[i - 2].left = w;
    }

    nodePath[i] = nodePath[i - 1];
    dirPath[i] = false;
    nodePath[i - 1] = w;
    if (maxDepth == i++) maxDepth++;

    w = nodePath[i - 1].right;
   }

   if ((w.pred() || w.left.black()) &&
     (w.succ() || w.right.black())) {
```

```
 w.black(false);
 }
 else {
  if (w.succ() || w.right.black()) {
   Entry KEY_VALUE_GENERIC y = w.left;

   y.black (true);
   w.black(false);
   w.left = y.right;
   y.right = w;
   w = nodePath[i - 1].right = y;

   if (w.succ()) {
    w.succ(false);
    w.right.pred(w);
   }
  }

  w.black(nodePath[i - 1].black());
  nodePath[i - 1].black(true);
  w.right.black(true);

  nodePath[i - 1].right = w.left;
  w.left = nodePath[i - 1];

  if (i < 2) tree = w;
  else {
   if (dirPath[i - 2]) nodePath[i - 2].right = w;
   else nodePath[i - 2].left = w;
  }

  if (w.pred()) {
   w.pred(false);
   nodePath[i - 1].succ(w);
  }
  break;
 }
}
else {
 Entry KEY_VALUE_GENERIC w = nodePath[i - 1].left;

 if (! w.black()) {
  w.black (true);
  nodePath[i - 1].black(false);

  nodePath[i - 1].left = w.right;
  w.right = nodePath[i - 1];
```

```
 if (i < 2) tree = w;
 else {
  if (dirPath[i - 2]) nodePath[i - 2].right = w;
  else nodePath[i - 2].left = w;
 }

 nodePath[i] = nodePath[i - 1];
 dirPath[i] = true;
 nodePath[i - 1] = w;
 if (maxDepth == i++) maxDepth++;

 w = nodePath[i - 1].left;
}

if ((w.pred() || w.left.black()) &&
  (w.succ() || w.right.black())) {
 w.black(false);
}
else {
 if (w.pred() || w.left.black()) {
  Entry KEY_VALUE_GENERIC y = w.right;

  y.black(true);
  w.black (false);
  w.right = y.left;
  y.left = w;
  w = nodePath[i - 1].left = y;

  if (w.pred()) {
   w.pred(false);
   w.left.succ(w);
  }
 }

 w.black(nodePath[i - 1].black());
 nodePath[i - 1].black(true);
 w.left.black(true);

 nodePath[i - 1].left = w.right;
 w.right = nodePath[i - 1];

 if (i < 2) tree = w;
 else {
  if (dirPath[i - 2]) nodePath[i - 2].right = w;
  else nodePath[i - 2].left = w;
 }

 if (w.succ()) {
```

```
      w.succ(false);
      nodePath[i - 1].pred(w);
     }
     break;
    }
   }
  }

  if (tree != null) tree.black(true);
 }

 modified = true;
 count--;
 // We clean up the node path, or we could have stale references later.
 while(maxDepth-- != 0) nodePath[maxDepth] = null;
 return p.value;
}


@Override
public boolean containsValue(final VALUE_TYPE v) {
 final ValueIterator i = new ValueIterator();
 VALUE_TYPE ev;

 int j = count;
 while(j-- != 0) {
  ev = i.NEXT_VALUE();
  if (VALUE_EQUALS(ev, v)) return true;
 }

 return false;
}


@Override
public void clear() {
 count = 0;
 tree = null;
 entries = null;
 values = null;
 keys = null;
 firstEntry = lastEntry = null;
}


/** This class represent an entry in a tree map.
 *
 * <p>We use the only "metadata", i.e., {@link Entry#info}, to store
```

```
 * information about color, predecessor status and successor status.
 *
 * <p>Note that since the class is recursive, it can be
 * considered equivalently a tree.
 */

 private static final class Entry KEY_VALUE_GENERIC extends ABSTRACT_MAP.BasicEntry
KEY_VALUE_GENERIC implements Cloneable {
  /** The the bit in this mask is true, the node is black. */
  private static final int BLACK_MASK = 1;
  /** If the bit in this mask is true, {@link #right} points to a successor. */
  private static final int SUCC_MASK = 1 << 31;
  /** If the bit in this mask is true, {@link #left} points to a predecessor. */
  private static final int PRED_MASK = 1 << 30;
  /** The pointers to the left and right subtrees. */
  Entry KEY_VALUE_GENERIC left, right;
  /** This integers holds different information in different bits (see {@link #SUCC_MASK} and {@link
#PRED_MASK}. */
  int info;

  Entry() {
   super(KEY_NULL, VALUE_NULL);
  }

  /** Creates a new entry with the given key and value.
   *
   * @param k a key.
   * @param v a value.
   */
  Entry(final KEY_GENERIC_TYPE k, final VALUE_GENERIC_TYPE v) {
   super(k, v);
   info = SUCC_MASK | PRED_MASK;
  }

  /** Returns the left subtree.
   *
   * @return the left subtree ({@code null} if the left
   * subtree is empty).
   */
  Entry KEY_VALUE_GENERIC left() {
   return (info & PRED_MASK) != 0 ? null : left;
  }

  /** Returns the right subtree.
   *
   * @return the right subtree ({@code null} if the right
   * subtree is empty).
   */
```

```
Entry KEY_VALUE_GENERIC right() {
 return (info & SUCC_MASK) != 0 ? null : right;
}

/** Checks whether the left pointer is really a predecessor.
 * @return true if the left pointer is a predecessor.
 */
boolean pred() {
 return (info & PRED_MASK) != 0;
}

/** Checks whether the right pointer is really a successor.
 * @return true if the right pointer is a successor.
 */
boolean succ() {
 return (info & SUCC_MASK) != 0;
}

/** Sets whether the left pointer is really a predecessor.
 * @param pred if true then the left pointer will be considered a predecessor.
 */
void pred(final boolean pred) {
 if (pred) info |= PRED_MASK;
 else info &= ~PRED_MASK;
}

/** Sets whether the right pointer is really a successor.
 * @param succ if true then the right pointer will be considered a successor.
 */
void succ(final boolean succ) {
 if (succ) info |= SUCC_MASK;
 else info &= ~SUCC_MASK;
}

/** Sets the left pointer to a predecessor.
 * @param pred the predecessr.
 */
void pred(final Entry KEY_VALUE_GENERIC pred) {
 info |= PRED_MASK;
 left = pred;
}

/** Sets the right pointer to a successor.
 * @param succ the successor.
 */
void succ(final Entry KEY_VALUE_GENERIC succ) {
 info |= SUCC_MASK;
 right = succ;
```

```
}

/** Sets the left pointer to the given subtree.
 * @param left the new left subtree.
 */
void left(final Entry KEY_VALUE_GENERIC left) {
 info &= ~PRED_MASK;
 this.left = left;
}

/** Sets the right pointer to the given subtree.
 * @param right the new right subtree.
 */
void right(final Entry KEY_VALUE_GENERIC right) {
 info &= ~SUCC_MASK;
 this.right = right;
}


/** Returns whether this node is black.
 * @return true iff this node is black.
 */
boolean black() {
 return (info & BLACK_MASK) != 0;
}

/** Sets whether this node is black.
 * @param black if true, then this node becomes black; otherwise, it becomes red..
 */
void black(final boolean black) {
 if (black) info |= BLACK_MASK;
 else info &= ~BLACK_MASK;
}

/** Computes the next entry in the set order.
 *
 * @return the next entry ({@code null}) if this is the last entry).
 */

Entry KEY_VALUE_GENERIC next() {
 Entry KEY_VALUE_GENERIC next = this.right;
 if ((info & SUCC_MASK) == 0) while ((next.info & PRED_MASK) == 0) next = next.left;
 return next;
}

/** Computes the previous entry in the set order.
 *
 * @return the previous entry ({@code null}) if this is the first entry).
```

```
 */

Entry KEY_VALUE_GENERIC prev() {
 Entry KEY_VALUE_GENERIC prev = this.left;
 if ((info & PRED_MASK) == 0) while ((prev.info & SUCC_MASK) == 0) prev = prev.right;
 return prev;
}

@Override
public VALUE_GENERIC_TYPE setValue(final VALUE_GENERIC_TYPE value) {
 final VALUE_GENERIC_TYPE oldValue = this.value;
 this.value = value;
 return oldValue;
}

@Override
SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
public Entry KEY_VALUE_GENERIC clone() {
 Entry KEY_VALUE_GENERIC c;
 try {
  c = (Entry KEY_VALUE_GENERIC)super.clone();
 }
 catch(CloneNotSupportedException cantHappen) {
  throw new InternalError();
 }

 c.key = key;
 c.value = value;
 c.info = info;

 return c;
}

@Override
@SuppressWarnings("unchecked")
public boolean equals(final Object o) {
 if (!(o instanceof Map.Entry)) return false;
 Map.Entry <KEY_GENERIC_CLASS, VALUE_GENERIC_CLASS> e = (Map.Entry <KEY_GENERIC_CLASS,
VALUE_GENERIC_CLASS>)o;

 return KEY_EQUALS(key, KEY_CLASS2TYPE(e.getKey())) && VALUE_EQUALS(value,
VALUE_CLASS2TYPE(e.getValue()));
 }

@Override
public int hashCode() {
 return KEY2JAVAHASH_NOT_NULL(key) ^ VALUE2JAVAHASH(value);
}
```

```
 @Override
 public String toString() {
  return key + "=>" + value;
 }

 /*
   public void prettyPrint() {
   prettyPrint(0);
   }

   public void prettyPrint(int level) {
   if (pred()) {
   for (int i = 0; i < level; i++)
   System.err.print("  ");
   System.err.println("pred: " + left);
   }
   else if (left != null)
   left.prettyPrint(level +1);
   for (int i = 0; i < level; i++)
   System.err.print("  ");
   System.err.println(key + "=" + value + " (" + balance() + ")");
   if (succ()) {
   for (int i = 0; i < level; i++)
   System.err.print("  ");
   System.err.println("succ: " + right);
   }
   else if (right != null)
   right.prettyPrint(level + 1);
   }*/
 }

 /*
   public void prettyPrint() {
   System.err.println("size: " + count);
   if (tree != null) tree.prettyPrint();
   }*/

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 @Override
 public boolean containsKey(final KEY_TYPE k) {
  return findKey(KEY_GENERIC_CAST k) != null;
 }

 @Override
 public int size() {
  return count;
```

```
}

@Override
public boolean isEmpty() {
 return count == 0;
}

SUPPRESS_WARNINGS_KEY_UNCHECKED
@Override
public VALUE_GENERIC_TYPE GET_VALUE(final KEY_TYPE k) {
 final Entry KEY_VALUE_GENERIC e = findKey(KEY_GENERIC_CAST k);
 return e == null ? defRetValue : e.value;
}

@Override
public KEY_GENERIC_TYPE FIRST_KEY() {
 if (tree == null) throw new NoSuchElementException();
 return firstEntry.key;
}

@Override
public KEY_GENERIC_TYPE LAST_KEY() {
 if (tree == null) throw new NoSuchElementException();
 return lastEntry.key;
}


/** An abstract iterator on the whole range.
 *
 * <p>This class can iterate in both directions on a threaded tree.
 */

private class TreeIterator {
 /** The entry that will be returned by the next call to {@link java.util.ListIterator#previous()} (or {@code null} if
no previous entry exists). */
 Entry KEY_VALUE_GENERIC prev;
 /** The entry that will be returned by the next call to {@link java.util.ListIterator#next()} (or {@code null} if no
next entry exists). */
 Entry KEY_VALUE_GENERIC next;
 /** The last entry that was returned (or {@code null} if we did not iterate or used {@link #remove()}). */
 Entry KEY_VALUE_GENERIC curr;
 /** The current index (in the sense of a {@link java.util.ListIterator}). Note that this value is not meaningful when
this {@link TreeIterator} has been created using the nonempty constructor.*/
 int index = 0;

 TreeIterator() {
  next = firstEntry;
 }
```

```
TreeIterator(final KEY_GENERIC_TYPE k) {
 if ((next = locateKey(k)) != null) {
  if (compare(next.key, k) <= 0) {
   prev = next;
   next = next.next();
  }
  else prev = next.prev();
 }
}

public boolean hasNext() { return next != null; }
public boolean hasPrevious() { return prev != null; }

void updateNext() {
 next = next.next();
}

Entry KEY_VALUE_GENERIC nextEntry() {
 if (! hasNext()) throw new NoSuchElementException();
 curr = prev = next;
 index++;
 updateNext();
 return curr;
}

void updatePrevious() {
 prev = prev.prev();
}

Entry KEY_VALUE_GENERIC previousEntry() {
 if (! hasPrevious()) throw new NoSuchElementException();
 curr = next = prev;
 index--;
 updatePrevious();
 return curr;
}

public int nextIndex() {
 return index;
}

public int previousIndex() {
 return index - 1;
}

public void remove() {
 if (curr == null) throw new IllegalStateException();
```

```
  /* If the last operation was a next(), we are removing an entry that preceeds
     the current index, and thus we must decrement it. */
  if (curr == prev) index--;
  next = prev = curr;
  updatePrevious();
  updateNext();
  RB_TREE_MAP.this.REMOVE_VALUE(curr.key);
  curr = null;
 }

 public int skip(final int n) {
  int i = n;
  while(i-- != 0 && hasNext()) nextEntry();
  return n - i - 1;
 }

 public int back(final int n) {
  int i = n;
  while(i-- != 0 && hasPrevious()) previousEntry();
  return n - i - 1;
 }
}


/** An iterator on the whole range.
 *
 * <p>This class can iterate in both directions on a threaded tree.
 */

 private class EntryIterator extends TreeIterator implements ObjectListIterator<MAP.Entry
KEY_VALUE_GENERIC> {
  EntryIterator() {}

  EntryIterator(final KEY_GENERIC_TYPE k) {
   super(k);
  }

  @Override
  public MAP.Entry KEY_VALUE_GENERIC next() { return nextEntry(); }
  @Override
  public MAP.Entry KEY_VALUE_GENERIC previous() { return previousEntry(); }
 }


 @Override
 public ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> ENTRYSET() {
  if (entries == null) entries = new AbstractObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC>() {
    final Comparator<? super MAP.Entry KEY_VALUE_GENERIC> comparator =  (Comparator<MAP.Entry
```

```
KEY_VALUE_GENERIC>) (x, y) -> RB_TREE_MAP.this.actualComparator.compare(x.ENTRY_GET_KEY(),
y.ENTRY_GET_KEY());

   @Override
   public Comparator<? super MAP.Entry KEY_VALUE_GENERIC> comparator() { return comparator; }

   @Override
   public ObjectBidirectionalIterator<MAP.Entry KEY_VALUE_GENERIC> iterator() { return new EntryIterator(); }

   @Override
   public ObjectBidirectionalIterator<MAP.Entry KEY_VALUE_GENERIC> iterator(final MAP.Entry
KEY_VALUE_GENERIC from) { return new EntryIterator(from.ENTRY_GET_KEY()); }

   @Override
   SUPPRESS_WARNINGS_KEY_UNCHECKED
   public boolean contains(final Object o) {
    if (!(o instanceof Map.Entry)) return false;
    final Map.Entry<?,?> e = (Map.Entry<?,?>)o;
#if KEYS_PRIMITIVE
    if (e.getKey() == null || ! (e.getKey() instanceof KEY_CLASS)) return false;
#endif
#if VALUES_PRIMITIVE
    if (e.getValue() == null || ! (e.getValue() instanceof VALUE_CLASS)) return false;
#endif
    final Entry KEY_VALUE_GENERIC f = findKey(KEY_OBJ2TYPE(KEY_GENERIC_CAST e.getKey()));
    return e.equals(f);
   }

   @Override
   SUPPRESS_WARNINGS_KEY_UNCHECKED
   public boolean remove(final Object o) {
    if (!(o instanceof Map.Entry)) return false;
    final Map.Entry<?,?> e = (Map.Entry<?,?>)o;
#if KEYS_PRIMITIVE
    if (e.getKey() == null || ! (e.getKey() instanceof KEY_CLASS)) return false;
#endif
#if VALUES_PRIMITIVE
    if (e.getValue() == null || ! (e.getValue() instanceof VALUE_CLASS)) return false;
#endif
    final Entry KEY_VALUE_GENERIC f = findKey(KEY_OBJ2TYPE(KEY_GENERIC_CAST e.getKey()));
    if (f == null || ! VALUE_EQUALS(f.ENTRY_GET_VALUE(), VALUE_OBJ2TYPE(e.getValue()))) return false;
    RB_TREE_MAP.this.REMOVE_VALUE(f.key);
    return true;
   }

   @Override
   public int size() { return count; }
```

```
    @Override
    public void clear() { RB_TREE_MAP.this.clear(); }

    @Override
    public MAP.Entry KEY_VALUE_GENERIC first() { return firstEntry; }

    @Override
    public MAP.Entry KEY_VALUE_GENERIC last() { return lastEntry; }

    @Override
    public ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> subSet(MAP.Entry KEY_VALUE_GENERIC
from, MAP.Entry KEY_VALUE_GENERIC to) { return subMap(from.ENTRY_GET_KEY(),
to.ENTRY_GET_KEY()).ENTRYSET(); }

    @Override
    public ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> headSet(MAP.Entry KEY_VALUE_GENERIC
to) { return headMap(to.ENTRY_GET_KEY()).ENTRYSET(); }

    @Override
    public ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> tailSet(MAP.Entry KEY_VALUE_GENERIC
from) { return tailMap(from.ENTRY_GET_KEY()).ENTRYSET(); }
   };

  return entries;
  }

  /** An iterator on the whole range of keys.
   *
   * <p>This class can iterate in both directions on the keys of a threaded tree. We
   * simply override the {@link java.util.ListIterator#next()}/{@link java.util.ListIterator#previous()} methods (and
possibly
   * their type-specific counterparts) so that they return keys instead of entries.
   */
  private final class KeyIterator extends TreeIterator implements KEY_LIST_ITERATOR KEY_GENERIC {
   public KeyIterator() {}
   public KeyIterator(final KEY_GENERIC_TYPE k) { super(k); }

   @Override
   public KEY_GENERIC_TYPE NEXT_KEY() { return nextEntry().key; }

   @Override
   public KEY_GENERIC_TYPE PREV_KEY() { return previousEntry().key; }
  };


  /** A keyset implementation using a more direct implementation for iterators. */
  private class KeySet extends ABSTRACT_SORTED_MAP KEY_VALUE_GENERIC.KeySet {
   @Override
```

```
 public KEY_BIDI_ITERATOR KEY_GENERIC iterator() { return new KeyIterator(); }
 @Override
 public KEY_BIDI_ITERATOR KEY_GENERIC iterator(final KEY_GENERIC_TYPE from) { return new
KeyIterator(from); }
 }

 /** Returns a type-specific sorted set view of the keys contained in this map.
  *
  * <p>In addition to the semantics of {@link java.util.Map#keySet()}, you can
  * safely cast the set returned by this call to a type-specific sorted
  * set interface.
  *
  * @return a type-specific sorted set view of the keys contained in this map.
  */
 @Override
 public SORTED_SET KEY_GENERIC keySet() {
  if (keys == null) keys = new KeySet();
  return keys;
 }

 /** An iterator on the whole range of values.
  *
  * <p>This class can iterate in both directions on the values of a threaded tree. We
  * simply override the {@link java.util.ListIterator#next()}/{@link java.util.ListIterator#previous()} methods (and
possibly
  * their type-specific counterparts) so that they return values instead of entries.
  */
 private final class ValueIterator extends TreeIterator implements VALUE_LIST_ITERATOR VALUE_GENERIC {
  @Override
  public VALUE_GENERIC_TYPE NEXT_VALUE() { return nextEntry().value; }

  @Override
  public VALUE_GENERIC_TYPE PREV_VALUE() { return previousEntry().value; }
 };

 /** Returns a type-specific collection view of the values contained in this map.
  *
  * <p>In addition to the semantics of {@link java.util.Map#values()}, you can
  * safely cast the collection returned by this call to a type-specific collection
  * interface.
  *
  * @return a type-specific collection view of the values contained in this map.
  */
 @Override
 public VALUE_COLLECTION VALUE_GENERIC values() {
  if (values == null) values = new VALUE_ABSTRACT_COLLECTION VALUE_GENERIC() {
    @Override
    public VALUE_ITERATOR VALUE_GENERIC iterator() { return new ValueIterator(); }
```

```
   @Override
   public boolean contains(final VALUE_TYPE k) { return containsValue(k); }
   @Override
   public int size() { return count; }
   @Override
   public void clear() { RB_TREE_MAP.this.clear(); }
  };

 return values;
 }

 @Override
 public KEY_COMPARATOR KEY_SUPER_GENERIC comparator() { return actualComparator; }

 @Override
 public SORTED_MAP KEY_VALUE_GENERIC headMap(KEY_GENERIC_TYPE to) { return new
Submap(KEY_NULL, true, to, false); }

 @Override
 public SORTED_MAP KEY_VALUE_GENERIC tailMap(KEY_GENERIC_TYPE from) { return new
Submap(from, false, KEY_NULL, true); }

 @Override
 public SORTED_MAP KEY_VALUE_GENERIC subMap(KEY_GENERIC_TYPE from, KEY_GENERIC_TYPE
to) { return new Submap(from, false, to, false); }

 /** A submap with given range.
  *
  * <p>This class represents a submap. One has to specify the left/right
  * limits (which can be set to -&infin; or &infin;). Since the submap is a
  * view on the map, at a given moment it could happen that the limits of
  * the range are not any longer in the main map. Thus, things such as
  * {@link java.util.SortedMap#firstKey()} or {@link java.util.Collection#size()} must be always computed
  * on-the-fly.
  */
 private final class Submap extends ABSTRACT_SORTED_MAP KEY_VALUE_GENERIC implements
java.io.Serializable {
  private static final long serialVersionUID = -7046029254386353129L;

  /** The start of the submap range, unless {@link #bottom} is true. */
  KEY_GENERIC_TYPE from;
  /** The end of the submap range, unless {@link #top} is true. */
  KEY_GENERIC_TYPE to;
  /** If true, the submap range starts from -&infin;. */
  boolean bottom;
  /** If true, the submap range goes to &infin;. */
  boolean top;
  /** Cached set of entries. */
```

```
 protected transient ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> entries;
 /** Cached set of keys. */
 protected transient SORTED_SET KEY_GENERIC keys;
 /** Cached collection of values. */
 protected transient VALUE_COLLECTION VALUE_GENERIC values;

 /** Creates a new submap with given key range.
  *
  * @param from the start of the submap range.
  * @param bottom if true, the first parameter is ignored and the range starts from -&infin;.
  * @param to the end of the submap range.
  * @param top if true, the third parameter is ignored and the range goes to &infin;.
  */
 public Submap(final KEY_GENERIC_TYPE from, final boolean bottom, final KEY_GENERIC_TYPE to, final
boolean top) {
  if (! bottom && ! top && RB_TREE_MAP.this.compare(from, to) > 0) throw new
IllegalArgumentException("Start key (" + from  + ") is larger than end key (" + to + ")");

  this.from = from;
  this.bottom = bottom;
  this.to = to;
  this.top = top;
  this.defRetValue = RB_TREE_MAP.this.defRetValue;
 }

 @Override
 public void clear() {
  final SubmapIterator i = new SubmapIterator();
  while(i.hasNext()) {
   i.nextEntry();
   i.remove();
  }
 }

 /** Checks whether a key is in the submap range.
  * @param k a key.
  * @return true if is the key is in the submap range.
  */
 final boolean in(final KEY_GENERIC_TYPE k) {
  return (bottom || RB_TREE_MAP.this.compare(k, from) >= 0) &&
   (top || RB_TREE_MAP.this.compare(k, to) < 0);
 }

 @Override
 public ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> ENTRYSET() {
  if (entries == null) entries = new AbstractObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC>() {
    @Override
    public ObjectBidirectionalIterator<MAP.Entry KEY_VALUE_GENERIC> iterator() {
```

```
        return new SubmapEntryIterator();
    }

    @Override
    public ObjectBidirectionalIterator<MAP.Entry KEY_VALUE_GENERIC> iterator(final MAP.Entry
KEY_VALUE_GENERIC from) {
        return new SubmapEntryIterator(from.ENTRY_GET_KEY());
    }

    @Override
    public Comparator<? super MAP.Entry KEY_VALUE_GENERIC> comparator() { return
RB_TREE_MAP.this.ENTRYSET().comparator(); }

    @Override
    SUPPRESS_WARNINGS_KEY_UNCHECKED
    public boolean contains(final Object o) {
        if (!(o instanceof Map.Entry)) return false;
        final Map.Entry<?,?> e = (Map.Entry<?,?>)o;
#if KEYS_PRIMITIVE
        if (e.getKey() == null || ! (e.getKey() instanceof KEY_CLASS)) return false;
#endif
#if VALUES_PRIMITIVE
        if (e.getValue() == null || ! (e.getValue() instanceof VALUE_CLASS)) return false;
#endif
        final RB_TREE_MAP.Entry KEY_VALUE_GENERIC f = findKey(KEY_OBJ2TYPE(KEY_GENERIC_CAST
e.getKey()));
        return f != null && in(f.key) && e.equals(f);
    }

    @Override
    SUPPRESS_WARNINGS_KEY_UNCHECKED
    public boolean remove(final Object o) {
        if (!(o instanceof Map.Entry)) return false;
        final Map.Entry<?,?> e = (Map.Entry<?,?>)o;
#if KEYS_PRIMITIVE
        if (e.getKey() == null || ! (e.getKey() instanceof KEY_CLASS)) return false;
#endif
#if VALUES_PRIMITIVE
        if (e.getValue() == null || ! (e.getValue() instanceof VALUE_CLASS)) return false;
#endif
        final RB_TREE_MAP.Entry KEY_VALUE_GENERIC f = findKey(KEY_OBJ2TYPE(KEY_GENERIC_CAST
e.getKey()));
        if (f != null && in(f.key)) Submap.this.REMOVE_VALUE(f.key);
        return f != null;
    }

    @Override
    public int size() {
```

```
    int c = 0;
    for(Iterator<?> i = iterator(); i.hasNext(); i.next()) c++;
    return c;
   }

   @Override
   public boolean isEmpty() { return ! new SubmapIterator().hasNext(); }

   @Override
   public void clear() { Submap.this.clear(); }

   @Override
   public MAP.Entry KEY_VALUE_GENERIC first() { return firstEntry(); }

   @Override
   public MAP.Entry KEY_VALUE_GENERIC last() { return lastEntry(); }

   @Override
   public ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> subSet(MAP.Entry KEY_VALUE_GENERIC
from, MAP.Entry KEY_VALUE_GENERIC to) { return subMap(from.ENTRY_GET_KEY(),
to.ENTRY_GET_KEY()).ENTRYSET(); }

   @Override
   public ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> headSet(MAP.Entry KEY_VALUE_GENERIC
to) { return headMap(to.ENTRY_GET_KEY()).ENTRYSET(); }

   @Override
   public ObjectSortedSet<MAP.Entry KEY_VALUE_GENERIC> tailSet(MAP.Entry KEY_VALUE_GENERIC
from) { return tailMap(from.ENTRY_GET_KEY()).ENTRYSET(); }
  };

  return entries;
 }

 private class KeySet extends ABSTRACT_SORTED_MAP KEY_VALUE_GENERIC.KeySet {
  @Override
  public KEY_BIDI_ITERATOR KEY_GENERIC iterator() { return new SubmapKeyIterator(); }
  @Override
  public KEY_BIDI_ITERATOR KEY_GENERIC iterator(final KEY_GENERIC_TYPE from) { return new
SubmapKeyIterator(from); }
 }

 @Override
 public SORTED_SET KEY_GENERIC keySet() {
  if (keys == null) keys = new KeySet();
  return keys;
 }
```

```
 @Override
 public VALUE_COLLECTION VALUE_GENERIC values() {
  if (values == null) values = new VALUE_ABSTRACT_COLLECTION VALUE_GENERIC() {
    @Override
    public VALUE_ITERATOR VALUE_GENERIC iterator() { return new SubmapValueIterator(); }
    @Override
    public boolean contains(final VALUE_TYPE k) { return containsValue(k); }
    @Override
    public int size() { return Submap.this.size(); }
    @Override
    public void clear() { Submap.this.clear(); }
   };

  return values;
 }

    @Override
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public boolean containsKey(final KEY_TYPE k) { return in(KEY_GENERIC_CAST k) &&
RB_TREE_MAP.this.containsKey(k); }

 @Override
 public boolean containsValue(final VALUE_TYPE v) {
  final SubmapIterator i = new SubmapIterator();
  VALUE_TYPE ev;

  while(i.hasNext()) {
   ev = i.nextEntry().value;
   if (VALUE_EQUALS(ev, v)) return true;
  }

  return false;
 }

 @Override
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public VALUE_GENERIC_TYPE GET_VALUE(final KEY_TYPE k) {
  final RB_TREE_MAP.Entry KEY_VALUE_GENERIC e;
  final KEY_GENERIC_TYPE kk = KEY_GENERIC_CAST k;
  return in(kk) && (e = findKey(kk)) != null ? e.value : this.defRetValue;
 }

 @Override
 public VALUE_GENERIC_TYPE put(final KEY_GENERIC_TYPE k, final VALUE_GENERIC_TYPE v) {
  modified = false;
  if (! in(k)) throw new IllegalArgumentException("Key (" + k + ") out of range [" + (bottom ? "-" :
String.valueOf(from)) + ", " + (top ? "-" : String.valueOf(to)) + ")");
  final VALUE_GENERIC_TYPE oldValue = RB_TREE_MAP.this.put(k, v);
```

```
 return modified ? this.defRetValue : oldValue;
}

@Override
SUPPRESS_WARNINGS_KEY_UNCHECKED
public VALUE_GENERIC_TYPE REMOVE_VALUE(final KEY_TYPE k) {
 modified = false;
 if (! in(KEY_GENERIC_CAST k)) return this.defRetValue;
 final VALUE_GENERIC_TYPE oldValue = RB_TREE_MAP.this.REMOVE_VALUE(k);
 return modified ? oldValue : this.defRetValue;
}

@Override
public int size() {
 final SubmapIterator i = new SubmapIterator();
 int n = 0;

 while(i.hasNext()) {
  n++;
  i.nextEntry();
 }

 return n;
}

@Override
public boolean isEmpty() { return ! new SubmapIterator().hasNext(); }

@Override
public KEY_COMPARATOR KEY_SUPER_GENERIC comparator() { return actualComparator; }

@Override
public SORTED_MAP KEY_VALUE_GENERIC headMap(final KEY_GENERIC_TYPE to) {
 if (top) return new Submap(from, bottom, to, false);
 return compare(to, this.to) < 0 ? new Submap(from, bottom, to, false) : this;
}

@Override
public SORTED_MAP KEY_VALUE_GENERIC tailMap(final KEY_GENERIC_TYPE from) {
 if (bottom) return new Submap(from, false, to, top);
 return compare(from, this.from) > 0 ? new Submap(from, false, to, top) : this;
}

@Override
public SORTED_MAP KEY_VALUE_GENERIC subMap(KEY_GENERIC_TYPE from, KEY_GENERIC_TYPE
to) {
 if (top && bottom) return new Submap(from, false, to, false);
 if (! top) to = compare(to, this.to) < 0 ? to : this.to;
```

```
 if (! bottom) from = compare(from, this.from) > 0 ? from : this.from;
 if (! top && ! bottom && from == this.from && to == this.to) return this;
 return new Submap(from, false, to, false);
}

/** Locates the first entry.
 *
 * @return the first entry of this submap, or {@code null} if the submap is empty.
 */
public RB_TREE_MAP.Entry KEY_VALUE_GENERIC firstEntry() {
 if (tree == null) return null;
 // If this submap goes to -infinity, we return the main map first entry; otherwise, we locate the start of the map.
 RB_TREE_MAP.Entry KEY_VALUE_GENERIC e;
 if (bottom) e = firstEntry;
 else {
  e = locateKey(from);
  // If we find either the start or something greater we're OK.
  if (compare(e.key, from) < 0) e = e.next();
 }
 // Finally, if this submap doesn't go to infinity, we check that the resulting key isn't greater than the end.
 if (e == null || ! top && compare(e.key, to) >= 0) return null;
 return e;
}

/** Locates the last entry.
 *
 * @return the last entry of this submap, or {@code null} if the submap is empty.
 */
public RB_TREE_MAP.Entry KEY_VALUE_GENERIC lastEntry() {
 if (tree == null) return null;
 // If this submap goes to infinity, we return the main map last entry; otherwise, we locate the end of the map.
 RB_TREE_MAP.Entry KEY_VALUE_GENERIC e;
 if (top) e = lastEntry;
 else {
  e = locateKey(to);
  // If we find something smaller than the end we're OK.
  if (compare(e.key, to) >= 0) e = e.prev();
 }
 // Finally, if this submap doesn't go to -infinity, we check that the resulting key isn't smaller than the start.
 if (e == null || ! bottom && compare(e.key, from) < 0) return null;
 return e;
}

@Override
public KEY_GENERIC_TYPE FIRST_KEY() {
 RB_TREE_MAP.Entry KEY_VALUE_GENERIC e = firstEntry();
 if (e == null) throw new NoSuchElementException();
 return e.key;
```

```
 }

 @Override
 public KEY_GENERIC_TYPE LAST_KEY() {
  RB_TREE_MAP.Entry KEY_VALUE_GENERIC e = lastEntry();
  if (e == null) throw new NoSuchElementException();
  return e.key;
 }

 /** An iterator for subranges.
  *
  * <p>This class inherits from {@link TreeIterator}, but overrides the methods that
  * update the pointer after a {@link java.util.ListIterator#next()} or {@link java.util.ListIterator#previous()}. If we
would
  * move out of the range of the submap we just overwrite the next or previous
  * entry with {@code null}.
  */
 private class SubmapIterator extends TreeIterator {
  SubmapIterator() {
   next = firstEntry();
  }

  SubmapIterator(final KEY_GENERIC_TYPE k) {
   this();

   if (next != null) {
    if (! bottom && compare(k, next.key) < 0) prev = null;
    else if (! top && compare(k, (prev = lastEntry()).key) >= 0) next = null;
    else {
     next = locateKey(k);

     if (compare(next.key, k) <= 0) {
      prev = next;
      next = next.next();
     }
     else prev = next.prev();
    }
   }
  }

  @Override
  void updatePrevious() {
   prev = prev.prev();
   if (! bottom && prev != null && RB_TREE_MAP.this.compare(prev.key, from) < 0) prev = null;
  }

  @Override
  void updateNext() {
```

```
  next = next.next();
  if (! top && next != null && RB_TREE_MAP.this.compare(next.key, to) >= 0) next = null;
  }
 }

 private class SubmapEntryIterator extends SubmapIterator implements ObjectListIterator<MAP.Entry
KEY_VALUE_GENERIC> {
  SubmapEntryIterator() {}

  SubmapEntryIterator(final KEY_GENERIC_TYPE k) {
   super(k);
  }

  @Override
  public MAP.Entry KEY_VALUE_GENERIC next() { return nextEntry(); }
  @Override
  public MAP.Entry KEY_VALUE_GENERIC previous() { return previousEntry(); }
 }


 /** An iterator on a subrange of keys.
  *
  * <p>This class can iterate in both directions on a subrange of the
  * keys of a threaded tree. We simply override the {@link
  * java.util.ListIterator#next()}/{@link java.util.ListIterator#previous()} methods (and possibly their
  * type-specific counterparts) so that they return keys instead of
  * entries.
  */
 private final class SubmapKeyIterator extends SubmapIterator implements KEY_LIST_ITERATOR
KEY_GENERIC {
  public SubmapKeyIterator() { super(); }
  public SubmapKeyIterator(KEY_GENERIC_TYPE from) { super(from); }

  @Override
  public KEY_GENERIC_TYPE NEXT_KEY() { return nextEntry().key; }
  @Override
  public KEY_GENERIC_TYPE PREV_KEY() { return previousEntry().key; }
 };

 /** An iterator on a subrange of values.
  *
  * <p>This class can iterate in both directions on the values of a
  * subrange of the keys of a threaded tree. We simply override the
  * {@link java.util.ListIterator#next()}/{@link java.util.ListIterator#previous()} methods (and possibly their
  * type-specific counterparts) so that they return values instead of
  * entries.
  */
 private final class SubmapValueIterator extends SubmapIterator implements VALUE_LIST_ITERATOR
```

```
VALUE_GENERIC {
  @Override
  public VALUE_GENERIC_TYPE NEXT_VALUE() { return nextEntry().value; }
  @Override
  public VALUE_GENERIC_TYPE PREV_VALUE() { return previousEntry().value; }
  };
 }


 /** Returns a deep copy of this tree map.
  *
  * <p>This method performs a deep copy of this tree map; the data stored in the
  * set, however, is not cloned. Note that this makes a difference only for object keys.
  *
  * @return a deep copy of this tree map.
  */
 @Override
 SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
 public RB_TREE_MAP KEY_VALUE_GENERIC clone() {
  RB_TREE_MAP KEY_VALUE_GENERIC c;
  try {
   c = (RB_TREE_MAP KEY_VALUE_GENERIC)super.clone();
  }
  catch(CloneNotSupportedException cantHappen) {
   throw new InternalError();
  }

  c.keys = null;
  c.values = null;
  c.entries = null;
  c.allocatePaths();

  if (count != 0) {
   // Also this apparently unfathomable code is derived from GNU libavl.
   Entry KEY_VALUE_GENERIC e, p, q, rp = new Entry KEY_VALUE_GENERIC_DIAMOND(), rq = new Entry
KEY_VALUE_GENERIC_DIAMOND();

   p = rp;
   rp.left(tree);

   q = rq;
   rq.pred(null);

   while(true) {
    if (! p.pred()) {
     e = p.left.clone();
     e.pred(q.left);
     e.succ(q);
```

```
    q.left(e);

    p = p.left;
    q = q.left;
   }
   else {
    while(p.succ()) {
     p = p.right;

     if (p == null) {
      q.right = null;
      c.tree = rq.left;

      c.firstEntry = c.tree;
      while(c.firstEntry.left != null) c.firstEntry = c.firstEntry.left;
      c.lastEntry = c.tree;
      while(c.lastEntry.right != null) c.lastEntry = c.lastEntry.right;

      return c;
     }
     q = q.right;
    }

    p = p.right;
    q = q.right;
   }

   if (! p.succ()) {
    e = p.right.clone();
    e.succ(q.right);
    e.pred(q);
    q.right(e);
   }
  }
 }

 return c;
}


private void writeObject(java.io.ObjectOutputStream s) throws java.io.IOException {
 int n = count;
 EntryIterator i = new EntryIterator();
 Entry KEY_VALUE_GENERIC e;

 s.defaultWriteObject();

 while(n-- != 0) {
```

```
  e = i.nextEntry();
  s.WRITE_KEY(e.key);
  s.WRITE_VALUE(e.value);
  }
 }


 /** Reads the given number of entries from the input stream, returning the corresponding tree.
  *
  * @param s the input stream.
  * @param n the (positive) number of entries to read.
  * @param pred the entry containing the key that preceeds the first key in the tree.
  * @param succ the entry containing the key that follows the last key in the tree.
  */
 SUPPRESS_WARNINGS_KEY_VALUE_UNCHECKED
 private Entry KEY_VALUE_GENERIC readTree(final java.io.ObjectInputStream s, final int n, final Entry
KEY_VALUE_GENERIC pred, final Entry KEY_VALUE_GENERIC succ) throws java.io.IOException,
ClassNotFoundException {
  if (n == 1) {
   final Entry KEY_VALUE_GENERIC top = new Entry
KEY_VALUE_GENERIC_DIAMOND(KEY_GENERIC_CAST s.READ_KEY(), VALUE_GENERIC_CAST
s.READ_VALUE());
   top.pred(pred);
   top.succ(succ);
   top.black(true);

   return top;
  }

  if (n == 2) {
   /* We handle separately this case so that recursion will
    *always* be on nonempty subtrees. */
   final Entry KEY_VALUE_GENERIC top = new Entry
KEY_VALUE_GENERIC_DIAMOND(KEY_GENERIC_CAST s.READ_KEY(), VALUE_GENERIC_CAST
s.READ_VALUE());
   top.black(true);
   top.right(new Entry KEY_VALUE_GENERIC_DIAMOND(KEY_GENERIC_CAST s.READ_KEY(),
VALUE_GENERIC_CAST s.READ_VALUE()));
   top.right.pred(top);
   top.pred(pred);
   top.right.succ(succ);

   return top;
  }

  // The right subtree is the largest one.
  final int rightN = n / 2, leftN = n - rightN - 1;
```

```
 final Entry KEY_VALUE_GENERIC top = new Entry KEY_VALUE_GENERIC_DIAMOND();

 top.left(readTree(s, leftN, pred, top));

 top.key = KEY_GENERIC_CAST s.READ_KEY();
 top.value = VALUE_GENERIC_CAST s.READ_VALUE();
 top.black(true);

 top.right(readTree(s, rightN, top, succ));

 if (n + 2 == ((n + 2)  & -(n + 2))) top.right.black(false); // Quick test for determining whether n + 2 is a power of 2.

 return top;
}

private void readObject(java.io.ObjectInputStream s) throws java.io.IOException, ClassNotFoundException {
 s.defaultReadObject();
 /* The storedComparator is now correctly set, but we must restore
    on-the-fly the actualComparator. */
 setActualComparator();
 allocatePaths();

 if (count != 0) {
  tree = readTree(s, count, null, null);
  Entry KEY_VALUE_GENERIC e;

  e = tree;
  while(e.left() != null) e = e.left();
  firstEntry = e;

  e = tree;
  while(e.right() != null) e = e.right();
  lastEntry = e;
 }
}

#ifdef ASSERTS_CODE
 private void checkNodePath() {
 for(int i = nodePath.length; i-- != 0;) assert nodePath[i] == null : i;
 }

 private static KEY_VALUE_GENERIC int checkTree(Entry KEY_VALUE_GENERIC e, int d, int D) {
  if (e == null) return 0;
  if (e.black()) d++;
  if (e.left() != null) D = checkTree(e.left(), d, D);
  if (e.right() != null) D = checkTree(e.right(), d, D);
  if (e.left() == null && e.right() == null) {
   if (D == -1) D = d;
```

```
  else if (D != d) throw new AssertionError("Mismatch between number of black nodes (" + D + " and " + d + ")");
 }
 return D;
}
#endif



#ifdef TEST

 private static long seed = System.currentTimeMillis();
 private static java.util.Random r = new java.util.Random(seed);

 private static KEY_TYPE genKey() {
#if KEY_CLASS_Byte || KEY_CLASS_Short || KEY_CLASS_Character
  return (KEY_TYPE)(r.nextInt());
#elif KEYS_PRIMITIVE
  return r.NEXT_KEY();
#else
  return Integer.toBinaryString(r.nextInt());
#endif
 }

 private static VALUE_TYPE genValue() {
#if VALUE_CLASS_Byte || VALUE_CLASS_Short || VALUE_CLASS_Character
  return (VALUE_TYPE)(r.nextInt());
#elif VALUES_PRIMITIVE
  return r.NEXT_VALUE();
#elif !VALUE_CLASS_Reference || KEY_CLASS_Reference
  return Integer.toBinaryString(r.nextInt());
#else
  return new java.io.Serializable() {};
#endif
 }


 private static java.text.NumberFormat format = new java.text.DecimalFormat("#,###.00");
 private static java.text.FieldPosition p = new java.text.FieldPosition(0);

 private static String format(double d) {
  StringBuffer s = new StringBuffer();
  return format.format(d, s, p).toString();
 }

 private static void speedTest(int n, boolean comp) {
  int i, j;
  RB_TREE_MAP m;
  java.util.TreeMap t;
```

```
KEY_TYPE k[] = new KEY_TYPE[n];
KEY_TYPE nk[] = new KEY_TYPE[n];
VALUE_TYPE v[] = new VALUE_TYPE[n];
long ms;

for(i = 0; i < n; i++) {
 k[i] = genKey();
 nk[i] = genKey();
 v[i] = genValue();
}

double totPut = 0, totYes = 0, totNo = 0, totAddTo = 0, totIterFor = 0, totIterBack = 0, totRemYes = 0, d, dd, ddd;

if (comp) { for(j = 0; j < 20; j++) {

 t = new java.util.TreeMap();


 /* We first add all pairs to t. */
 for(i = 0; i < n;  i++) t.put(KEY2OBJ(k[i]), VALUE2OBJ(v[i]));

 /* Then we remove the first half and put it back. */
 for(i = 0; i < n/2;  i++) t.remove(KEY2OBJ(k[i]));

 ms = System.currentTimeMillis();
 for(i = 0; i < n/2;  i++) t.put(KEY2OBJ(k[i]), VALUE2OBJ(v[i]));
 d = System.currentTimeMillis() - ms;

 /* Then we remove the other half and put it back again. */
 ms = System.currentTimeMillis();
 for(i = n/2; i < n;  i++) t.remove(KEY2OBJ(k[i]));
 dd = System.currentTimeMillis() - ms ;

 ms = System.currentTimeMillis();
 for(i = n/2; i < n;  i++) t.put(KEY2OBJ(k[i]), VALUE2OBJ(v[i]));
 d += System.currentTimeMillis() - ms;
 if (j > 2) totPut += n/d;
 System.out.print("Add: " + format(n/d) +" K/s ");

 /* Then we remove again the first half. */
 ms = System.currentTimeMillis();
 for(i = 0; i < n/2;  i++) t.remove(KEY2OBJ(k[i]));
 dd += System.currentTimeMillis() - ms ;
 if (j > 2) totRemYes += n/dd;
 System.out.print("RemYes: " + format(n/dd) +" K/s ");

 /* And then we put it back. */
 for(i = 0; i < n/2;  i++) t.put(KEY2OBJ(k[i]), VALUE2OBJ(v[i]));
```

```
  #if VALUES_PRIMITIVE && !VALUE_CLASS_Boolean
  /* we perform n/2 addTo() operations with get then put */
  ms = System.currentTimeMillis();
  for(i = 0; i < n/2; i++) t.put(KEY2OBJ(k[i]), (VALUE_TYPE) ((VALUE_CLASS) t.get(KEY2OBJ(k[i])) + i));
  ddd = System.currentTimeMillis() - ms;
  if (j > 2) totAddTo += n/ddd;
  System.out.print("AddTo: " + format(n/ddd) +" K/s ");
  #endif

  /* We check for pairs in t. */
  ms = System.currentTimeMillis();
  for(i = 0; i < n;  i++) t.containsKey(KEY2OBJ(k[i]));
  d = 1.0 * n / (System.currentTimeMillis() - ms);
  if (j > 2) totYes += d;
  System.out.print("Yes: " + format(d) +" K/s ");

  /* We check for pairs not in t. */
  ms = System.currentTimeMillis();
  for(i = 0; i < n;  i++) t.containsKey(KEY2OBJ(nk[i]));
  d = 1.0 * n / (System.currentTimeMillis() - ms);
  if (j > 2) totNo += d;
  System.out.print("No: " + format(d) +" K/s ");

  /* We iterate on t. */
  ms = System.currentTimeMillis();
  for(Iterator it = t.entrySet().iterator(); it.hasNext(); it.next());
  d = 1.0 * n / (System.currentTimeMillis() - ms);
  if (j > 2) totIterFor += d;
  System.out.print("IterFor: " + format(d) +" K/s ");

  System.out.println();
  }

  System.out.println();
  System.out.println("java.util Put: " + format(totPut/(j-3)) + " K/s RemYes: " + format(totRemYes/(j-3)) + " K/s Yes:
" + format(totYes/(j-3)) + " K/s No: " + format(totNo/(j-3))+ "K/s AddTo: " + format(totAddTo/(j-3)) + " K/s
IterFor: " + format(totIterFor/(j-3))  + " K/s");

  System.out.println();

  t = null;
  totPut = totYes = totNo = totIterFor = totIterBack = totRemYes = 0;

  }

  for(j = 0; j < 20; j++) {
```

```
m = new RB_TREE_MAP();


/* We first add all pairs to m. */
for(i = 0; i < n;  i++) m.put(k[i], v[i]);

/* Then we remove the first half and put it back. */
for(i = 0; i < n/2;  i++) m.remove(k[i]);

ms = System.currentTimeMillis();
for(i = 0; i < n/2;  i++) m.put(k[i], v[i]);
d = System.currentTimeMillis() - ms;

/* Then we remove the other half and put it back again. */
ms = System.currentTimeMillis();
for(i = n/2; i < n;  i++) m.remove(k[i]);
dd = System.currentTimeMillis() - ms ;

ms = System.currentTimeMillis();
for(i = n/2; i < n;  i++) m.put(k[i], v[i]);
d += System.currentTimeMillis() - ms;
if (j > 2) totPut += n/d;
System.out.print("Add: " + format(n/d) +" K/s ");

/* Then we remove again the first half. */
ms = System.currentTimeMillis();
for(i = 0; i < n/2;  i++) m.remove(k[i]);
dd += System.currentTimeMillis() - ms ;
if (j > 2) totRemYes += n/dd;
System.out.print("RemYes: " + format(n/dd) +" K/s ");

/* And then we put it back. */
for(i = 0; i < n/2;  i++) m.put(k[i], v[i]);

#if VALUES_PRIMITIVE && !VALUE_CLASS_Boolean
/* we perform n/2 addTo() operations with get then put */
ms = System.currentTimeMillis();
for(i = 0; i < n/2; i++) m.addTo(k[i], (VALUE_TYPE) i);
ddd = System.currentTimeMillis() - ms;
if (j > 2) totAddTo += n/ddd;
System.out.print("AddTo: " + format(n/ddd) +" K/s ");
#endif

/* We check for pairs in m. */
ms = System.currentTimeMillis();
for(i = 0; i < n;  i++) m.containsKey(k[i]);
d = 1.0 * n / (System.currentTimeMillis() - ms);
if (j > 2) totYes += d;
```

```
  System.out.print("Yes: " + format(d) +" K/s ");

  /* We check for pairs not in m. */
  ms = System.currentTimeMillis();
  for(i = 0; i < n;  i++) m.containsKey(nk[i]);
  d = 1.0 * n / (System.currentTimeMillis() - ms);
  if (j > 2) totNo += d;
  System.out.print("No: " + format(d) +" K/s ");

  /* We iterate on m. */
  java.util.ListIterator it = (java.util.ListIterator)m.entrySet().iterator();
  ms = System.currentTimeMillis();
  for(; it.hasNext(); it.next());
  d = 1.0 * n / (System.currentTimeMillis() - ms);
  if (j > 2) totIterFor += d;
  System.out.print("IterFor: " + format(d) +" K/s ");

  /* We iterate back on m. */
  ms = System.currentTimeMillis();
  for(; it.hasPrevious(); it.previous());
  d = 1.0 * n / (System.currentTimeMillis() - ms);
  if (j > 2) totIterBack += d;
  System.out.print("IterBack: " + format(d) +" K/s ");

  System.out.println();
  }


  System.out.println();
  System.out.println("fastutil Put: " + format(totPut/(j-3)) + " K/s RemYes: " + format(totRemYes/(j-3)) + " K/s Yes:
" + format(totYes/(j-3)) + " K/s No: " + format(totNo/(j-3))+ "K/s AddTo: " + format(totAddTo/(j-3)) + " K/s
IterFor: " + format(totIterFor/(j-3))  + " K/s");

  System.out.println();

  }


 private static boolean valEquals(Object o1, Object o2) {
  return o1 == null ? o2 == null : o1.equals(o2);
 }

 private static void fatal(String msg) {
  System.out.println(msg);
  System.exit(1);
 }

 private static void ensure(boolean cond, String msg) {
```

```
 if (cond) return;
 fatal(msg);
}

private static Object[] k, v, nk;
private static KEY_TYPE kt[];
private static KEY_TYPE nkt[];
private static VALUE_TYPE vt[];
private static RB_TREE_MAP topMap;

protected static void testMaps(SORTED_MAP m, SortedMap t, int n, int level) {
 long ms;
 boolean mThrowsIllegal, tThrowsIllegal, mThrowsNoElement, tThrowsNoElement;
 Object rt = null, rm = null;

 if (level > 4) return;


 /* Now we check that both maps agree on first/last keys. */

 mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

 try {
  m.firstKey();
 }
 catch (NoSuchElementException e) { mThrowsNoElement = true; }
 try {
  t.firstKey();
 }
 catch (NoSuchElementException e) { tThrowsNoElement = true; }

 ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): firstKey() divergence at
start in NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
 if (! mThrowsNoElement) ensure(t.firstKey().equals(m.firstKey()), "Error (" + level + ", " + seed + "): m and t differ
at start on their first key (" + m.firstKey() + ", " + t.firstKey() +")");

 mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

 try {
  m.lastKey();
 }
 catch (NoSuchElementException e) { mThrowsNoElement = true; }
 try {
  t.lastKey();
 }
 catch (NoSuchElementException e) { tThrowsNoElement = true; }

 ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): lastKey() divergence at start
```

```
in NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");


  if (! mThrowsNoElement) ensure(t.lastKey().equals(m.lastKey()), "Error (" + level + ", " + seed + "): m and t differ
at start on their last key (" + m.lastKey() + ", " + t.lastKey() +")");


  /* Now we check that m and t are equal. */
  if (!m.equals(t) || ! t.equals(m)) System.err.println("m: " + m + " t: " + t);

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) at start");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) at start");



  /* Now we check that m actually holds that data. */
  for(Iterator i=t.entrySet().iterator(); i.hasNext();) {
   java.util.Map.Entry e = (java.util.Map.Entry)i.next();
   ensure(valEquals(e.getValue(), m.get(e.getKey())), "Error (" + level + ", " + seed + "): m and t differ on an entry
("+e+") after insertion (iterating on t)");
  }

  /* Now we check that m actually holds that data, but iterating on m. */
  for(Iterator i=m.entrySet().iterator(); i.hasNext();) {
   Entry e = (Entry)i.next();
   ensure(valEquals(e.getValue(), t.get(e.getKey())), "Error (" + level + ", " + seed + "): m and t differ on an entry
("+e+") after insertion (iterating on m)");
  }

  /* Now we check that m actually holds the same keys. */
  for(Iterator i=t.keySet().iterator(); i.hasNext();) {
   Object o = i.next();
   ensure(m.containsKey(o), "Error (" + level + ", " + seed + "): m and t differ on a key ("+o+") after insertion
(iterating on t)");
   ensure(m.keySet().contains(o), "Error (" + level + ", " + seed + "): m and t differ on a key ("+o+", in keySet()) after
insertion (iterating on t)");
  }

  /* Now we check that m actually holds the same keys, but iterating on m. */
  for(Iterator i=m.keySet().iterator(); i.hasNext();) {
   Object o = i.next();
   ensure(t.containsKey(o), "Error (" + level + ", " + seed + "): m and t differ on a key after insertion (iterating on m)");
   ensure(t.keySet().contains(o), "Error (" + level + ", " + seed + "): m and t differ on a key (in keySet()) after insertion
(iterating on m)");
  }


  /* Now we check that m actually hold the same values. */
```

```
  for(Iterator i=t.values().iterator(); i.hasNext();) {
   Object o = i.next();
   ensure(m.containsValue(o), "Error (" + level + ", " + seed + "): m and t differ on a value after insertion (iterating on t)");
   ensure(m.values().contains(o), "Error (" + level + ", " + seed + "): m and t differ on a value (in values()) after insertion (iterating on t)");
  }

  /* Now we check that m actually hold the same values, but iterating on m. */
  for(Iterator i=m.values().iterator(); i.hasNext();) {
   Object o = i.next();
   ensure(t.containsValue(o), "Error (" + level + ", " + seed + "): m and t differ on a value after insertion (iterating on m)");
   ensure(t.values().contains(o), "Error (" + level + ", " + seed + "): m and t differ on a value (in values()) after insertion (iterating on m)");
  }

  /* Now we check that inquiries about random data give the same answer in m and t. For
     m we use the polymorphic method. */

  for(int i=0; i<n;  i++) {
   KEY_TYPE T = genKey();

   mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

   try {
    m.containsKey(KEY2OBJ(T));
   }
   catch (NoSuchElementException e) { mThrowsNoElement = true; }
   catch (IllegalArgumentException e) { mThrowsIllegal = true; }

   try {
    t.containsKey(KEY2OBJ(T));
   }
   catch (NoSuchElementException e) { tThrowsNoElement = true; }
   catch (IllegalArgumentException e) { tThrowsIllegal = true; }

   ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): containsKey() divergence in NoSuchElementException (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
   ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): containsKey() divergence in IllegalArgumentException (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
   if (!mThrowsNoElement && !mThrowsIllegal) {
    ensure(m.containsKey(KEY2OBJ(T)) == t.containsKey(KEY2OBJ(T)), "Error (" + level + ", " + seed + "): divergence in keys between t and m (polymorphic method)");

#if KEY_CLASS_Object && ! (VALUES_REFERENCE)
    if ((m.GET_VALUE(T) != VALUE_NULL) != ((t.get(KEY2OBJ(T)) == null ? VALUE_NULL : VALUE_OBJ2TYPE(t.get(KEY2OBJ(T)))) != VALUE_NULL) ||
```

```
    t.get(KEY2OBJ(T)) != null &&
    ! VALUE2OBJ(m.GET_VALUE(T)).equals(t.get(KEY2OBJ(T))))
#else
    if ((m.get(T) != VALUE_NULL) != ((t.get(KEY2OBJ(T)) == null ? VALUE_NULL :
VALUE_OBJ2TYPE(t.get(KEY2OBJ(T)))) != VALUE_NULL) ||
     t.get(KEY2OBJ(T)) != null &&
     ! m.get(KEY2OBJ(T)).equals(t.get(KEY2OBJ(T))))
#endif
     {
      System.out.println("Error (" + level + ", " + seed + "): divergence between t and m (polymorphic method)");
      System.exit(1);
     }
   }
  }

  /* Again, we check that inquiries about random data give the same answer in m and t, but
     for m we use the standard method. */

  for(int i=0; i<n;  i++) {
   KEY_TYPE T = genKey();

   mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

   try {
    m.get(KEY2OBJ(T));
   }
   catch (NoSuchElementException e) { mThrowsNoElement = true; }
   catch (IllegalArgumentException e) { mThrowsIllegal = true; }

   try {
    t.get(KEY2OBJ(T));
   }
   catch (NoSuchElementException e) { tThrowsNoElement = true; }
   catch (IllegalArgumentException e) { tThrowsIllegal = true; }

   ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): get() divergence in
NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
   ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): get() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
   if (!mThrowsNoElement && !mThrowsIllegal) ensure(valEquals(m.get(KEY2OBJ(T)), t.get(KEY2OBJ(T))),
"Error (" + level + ", " + seed + "): divergence between t and m (standard method)");
  }

  /* Now we put and remove random data in m and t, checking that the result is the same. */

  for(int i=0; i<20*n;  i++) {
   KEY_TYPE T = genKey();
   VALUE_TYPE U = genValue();
```

```
  mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

  try {
   rm = m.put(KEY2OBJ(T), VALUE2OBJ(U));
  }
  catch (NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }

  try {
   rt = t.put(KEY2OBJ(T), VALUE2OBJ(U));
  }
  catch (NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }


  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): put() divergence in
NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): put() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
  if (!mThrowsNoElement && !mThrowsIllegal) ensure(valEquals(rm, rt), "Error (" + level + ", " + seed + "):
divergence in put() between t and m (" + rt + ", " + rm + ")");

  T = genKey();

  mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

  try {
   rm = m.remove(KEY2OBJ(T));
  }
  catch (NoSuchElementException e) { mThrowsNoElement = true; }
  catch (IllegalArgumentException e) { mThrowsIllegal = true; }

  try {
   rt = t.remove(KEY2OBJ(T));
  }
  catch (NoSuchElementException e) { tThrowsNoElement = true; }
  catch (IllegalArgumentException e) { tThrowsIllegal = true; }


  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): remove() divergence in
NoSuchElementException for " + T + " (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
  ensure(mThrowsIllegal == tThrowsIllegal, "Error (" + level + ", " + seed + "): remove() divergence in
IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
  if (!mThrowsNoElement && !mThrowsIllegal) ensure(valEquals(rm, rt), "Error (" + level + ", " + seed + "):
divergence in remove() between t and m (" + rt + ", " + rm + ")");
  }
```

```
  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after removal");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after removal");

  /* Now we check that m actually holds the same data. */

  for(Iterator i=t.entrySet().iterator(); i.hasNext();) {
   java.util.Map.Entry e = (java.util.Map.Entry)i.next();
   ensure(valEquals(e.getValue(), m.get(e.getKey())), "Error (" + level + ", " + seed + "): m and t differ on an entry
("+e+") after removal (iterating on t)");
  }

  /* Now we check that m actually holds that data, but iterating on m. */

  for(Iterator i=m.entrySet().iterator(); i.hasNext();) {
   Entry e = (Entry)i.next();
   ensure(valEquals(e.getValue(), t.get(e.getKey())), "Error (" + level + ", " + seed + "): m and t differ on an entry
("+e+") after removal (iterating on m)");
  }

  /* Now we check that m actually holds the same keys. */

  for(Iterator i=t.keySet().iterator(); i.hasNext();) {
   Object o = i.next();
   ensure(m.containsKey(o), "Error (" + level + ", " + seed + "): m and t differ on a key ("+o+") after removal (iterating
on t)");
   ensure(m.keySet().contains(o), "Error (" + level + ", " + seed + "): m and t differ on a key ("+o+", in keySet()) after
removal (iterating on t)");
  }

  /* Now we check that m actually holds the same keys, but iterating on m. */

  for(Iterator i=m.keySet().iterator(); i.hasNext();) {
   Object o = i.next();
   ensure(t.containsKey(o), "Error (" + level + ", " + seed + "): m and t differ on a key after removal (iterating on m)");
   ensure(t.keySet().contains(o), "Error (" + level + ", " + seed + "): m and t differ on a key (in keySet()) after removal
(iterating on m)");
  }


  /* Now we check that m actually hold the same values. */

  for(Iterator i=t.values().iterator(); i.hasNext();) {
   Object o = i.next();
   ensure(m.containsValue(o), "Error (" + level + ", " + seed + "): m and t differ on a value after removal (iterating on
t)");
   ensure(m.values().contains(o), "Error (" + level + ", " + seed + "): m and t differ on a value (in values()) after
removal (iterating on t)");
  }
```

```
  /* Now we check that m actually hold the same values, but iterating on m. */

  for(Iterator i=m.values().iterator(); i.hasNext();) {
   Object o = i.next();
   ensure(t.containsValue(o), "Error (" + level + ", " + seed + "): m and t differ on a value after removal (iterating on
m)");
   ensure(t.values().contains(o), "Error (" + level + ", " + seed + "): m and t differ on a value (in values()) after removal
(iterating on m)");
  }

  /* Now we check that both maps agree on first/last keys. */

  mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

  try {
   m.firstKey();
  }
  catch (NoSuchElementException e) { mThrowsNoElement = true; }
  try {
   t.firstKey();
  }
  catch (NoSuchElementException e) { tThrowsNoElement = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): firstKey() divergence in
NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
  if (! mThrowsNoElement) ensure(t.firstKey().equals(m.firstKey()), "Error (" + level + ", " + seed + "): m and t differ
on their first key (" + m.firstKey() + ", " + t.firstKey() +")");

  mThrowsNoElement = mThrowsIllegal = tThrowsNoElement = tThrowsIllegal = false;

  try {
   m.lastKey();
  }
  catch (NoSuchElementException e) { mThrowsNoElement = true; }
  try {
   t.lastKey();
  }
  catch (NoSuchElementException e) { tThrowsNoElement = true; }

  ensure(mThrowsNoElement == tThrowsNoElement, "Error (" + level + ", " + seed + "): lastKey() divergence in
NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");

  if (! mThrowsNoElement) ensure(t.lastKey().equals(m.lastKey()), "Error (" + level + ", " + seed + "): m and t differ
on their last key (" + m.lastKey() + ", " + t.lastKey() +")");

  /* Now we check cloning. */
```

```
  if (level == 0) {
   ensure(m.equals(((RB_TREE_MAP)m).clone()), "Error (" + level + ", " + seed + "): m does not equal m.clone()");
   ensure(((RB_TREE_MAP)m).clone().equals(m), "Error (" + level + ", " + seed + "): m.clone() does not equal m");
   m = (RB_TREE_MAP)((RB_TREE_MAP)m).clone();
  }

  int h = m.hashCode();


  /* Now we save and read m. */

  SORTED_MAP m2 = null;

  try {
   java.io.File ff = new java.io.File("it.unimi.dsi.fastutil.test");
   java.io.OutputStream os = new java.io.FileOutputStream(ff);
   java.io.ObjectOutputStream oos = new java.io.ObjectOutputStream(os);

   oos.writeObject(m);
   oos.close();

   java.io.InputStream is = new java.io.FileInputStream(ff);
   java.io.ObjectInputStream ois = new java.io.ObjectInputStream(is);

   m2 = (SORTED_MAP)ois.readObject();
   ois.close();
   ff.delete();
  }
  catch(Exception e) {
   e.printStackTrace();
   System.exit(1);
  }

#if !VALUE_CLASS_Reference
  ensure(m2.hashCode() == h, "Error (" + level + ", " + seed + "): hashCode() changed after save/read");

  /* Now we check that m2 actually holds that data. */

  ensure(m2.equals(t), "Error (" + level + ", " + seed + "): ! m2.equals(t) after save/read");
  ensure(t.equals(m2), "Error (" + level + ", " + seed + "): ! t.equals(m2) after save/read");
  /* Now we take out of m everything, and check that it is empty. */
#else
  m2.clear();
  m2.putAll(m);
#endif

  for(Iterator i=t.keySet().iterator(); i.hasNext();) m2.remove(i.next());
```

```
 ensure(m2.isEmpty(), "Error (" + level + ", " + seed + "): m2 is not empty (as it should be)");

 /* Now we play with iterators. */

 {
  java.util.ListIterator i, j;
  Map.Entry E, F;
  Object J;
  i = (java.util.ListIterator)m.entrySet().iterator();
  j = new java.util.LinkedList(t.entrySet()).listIterator();

  for(int k = 0; k < 2*n; k++) {
   ensure(i.hasNext() == j.hasNext(), "Error (" + level + ", " + seed + "): divergence in hasNext()");
   ensure(i.hasPrevious() == j.hasPrevious(), "Error (" + level + ", " + seed + "): divergence in hasPrevious()");

   if (r.nextFloat() < .8 && i.hasNext()) {
    ensure((E=(Entry)i.next()).getKey().equals(J = (F=(Map.Entry)j.next()).getKey()), "Error (" + level + ", " + seed +
"): divergence in next()");

    if (r.nextFloat() < 0.3) {
     i.remove();
     j.remove();
     t.remove(J);
    }
    else if (r.nextFloat() < 0.3) {
     Object U = VALUE2OBJ(genValue());
     E.setValue(U);
                t.put(F.getKey(), U);
    }
   }
   else if (r.nextFloat() < .2 && i.hasPrevious()) {
    ensure((E=(Entry)i.previous()).getKey().equals(J = (F=(Map.Entry)j.previous()).getKey()), "Error (" + level + ", " +
seed + "): divergence in previous()");

    if (r.nextFloat() < 0.3) {
     i.remove();
     j.remove();
     t.remove(J);
    }
    else if (r.nextFloat() < 0.3) {
     Object U = VALUE2OBJ(genValue());
     E.setValue(U);
                t.put(F.getKey(), U);
    }
   }

   ensure(i.nextIndex() == j.nextIndex(), "Error (" + level + ", " + seed + "): divergence in nextIndex()");
   ensure(i.previousIndex() == j.previousIndex(), "Error (" + level + ", " + seed + "): divergence in previousIndex()");
```

```
  }
 }

 {
  boolean badPrevious = false;
  Object previous = null;
  it.unimi.dsi.fastutil.BidirectionalIterator i;
  java.util.ListIterator j;
  Object I, J;
  KEY_TYPE from = genKey();
  j = new java.util.LinkedList(t.keySet()).listIterator();
  while(j.hasNext()) {
   Object k = j.next();
   if (((Comparable)k).compareTo(KEY2OBJ(from)) > 0) {
    badPrevious = true;
    j.previous();
    break;
   }
   previous = k;
  }

  i = (it.unimi.dsi.fastutil.BidirectionalIterator)((SORTED_SET)m.keySet()).iterator(from);

  for(int k = 0; k < 2*n; k++) {
   ensure(i.hasNext() == j.hasNext(), "Error (" + level + ", " + seed + "): divergence in hasNext() (iterator with starting point " + from + ")");
   ensure(i.hasPrevious() == j.hasPrevious() || badPrevious && (i.hasPrevious() == (previous != null)), "Error (" + level + ", " + seed + "): divergence in hasPrevious() (iterator with starting point " + from + ")" + badPrevious);

   if (r.nextFloat() < .8 && i.hasNext()) {
    ensure((I = i.next()).equals(J = j.next()), "Error (" + level + ", " + seed + "): divergence in next() (" + I + ", " + J + ", iterator with starting point " + from + ")");
    //System.err.println("Done next " + I + " " + J + "  " + badPrevious);

    badPrevious = false;

    if (r.nextFloat() < 0.5) {
     //System.err.println("Removing in next");
     i.remove();
     j.remove();
     t.remove(J);
    }
   }
   else if (!badPrevious && r.nextFloat() < .2 && i.hasPrevious()) {
    ensure((I = i.previous()).equals(J = j.previous()), "Error (" + level + ", " + seed + "): divergence in previous() (" + I + ", " + J + ", iterator with starting point " + from + ")");
```

```
    if (r.nextFloat() < 0.5) {
     //System.err.println("Removing in prev");
     i.remove();
     j.remove();
     t.remove(J);
    }
   }
  }

 }

 /* Now we check that m actually holds that data. */

 ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after iteration");
 ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after iteration");

 /* Now we select a pair of keys and create a submap. */

 if (! m.isEmpty()) {
  java.util.ListIterator i;
  Object start = m.firstKey(), end = m.firstKey();
  for(i = (java.util.ListIterator)m.keySet().iterator(); i.hasNext() && r.nextFloat() < .3; start = end = i.next());
  for(; i.hasNext() && r.nextFloat() < .95; end = i.next());

  //System.err.println("Checking subMap from " + start + " to " + end + " (level=" + (level+1) + ")...");
  testMaps((SORTED_MAP)m.subMap((KEY_CLASS) start, (KEY_CLASS)end), t.subMap(start, end), n, level +
1);

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after subMap");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after subMap");

  //System.err.println("Checking headMap to " + end + " (level=" + (level+1) + ")...");
  testMaps((SORTED_MAP)m.headMap((KEY_CLASS)end), t.headMap(end), n, level + 1);

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after headMap");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after headMap");

  //System.err.println("Checking tailMap from " + start + " (level=" + (level+1) + ")...");
  testMaps((SORTED_MAP)m.tailMap((KEY_CLASS)start), t.tailMap(start), n, level + 1);

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after tailMap");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after tailMap");
 }


}
```

```
 private static void runTest(int n) {
  RB_TREE_MAP m = new RB_TREE_MAP();
  SortedMap t = new java.util.TreeMap();
  topMap = m;
  k = new Object[n];
  v = new Object[n];
  nk = new Object[n];
  kt = new KEY_TYPE[n];
  nkt = new KEY_TYPE[n];
  vt = new VALUE_TYPE[n];

  for(int i = 0; i < n; i++) {
#if KEY_CLASS_Object
   k[i] = kt[i] = genKey();
   nk[i] = nkt[i] = genKey();
#else
   k[i] = new KEY_CLASS(kt[i] = genKey());
   nk[i] = new KEY_CLASS(nkt[i] = genKey());
#endif
#if VALUES_REFERENCE
   v[i] = vt[i] = genValue();
#else
   v[i] = new VALUE_CLASS(vt[i] = genValue());
#endif
  }

  /* We add pairs to t. */
  for(int i = 0; i < n;  i++) t.put(k[i], v[i]);

  /* We add to m the same data */
  m.putAll(t);

  testMaps(m, t, n, 0);

  System.out.println("Test OK");
  return;
 }


 public static void main(String args[]) {
  int n  = Integer.parseInt(args[1]);
  if (args.length > 2) r = new java.util.Random(seed = Long.parseLong(args[2]));

  try {
   if ("speedTest".equals(args[0]) || "speedComp".equals(args[0])) speedTest(n, "speedComp".equals(args[0]));
   else if ("test".equals(args[0])) runTest(n);
  } catch(Throwable e) {
   e.printStackTrace(System.err);
```

```
  System.err.println("seed: " + seed);
 }
}

#endif


}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/RBTreeMap.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import java.util.Iterator;
#if defined JDK_PRIMITIVE_ITERATOR && !KEY_WIDENED
import java.util.PrimitiveIterator;
#endif
#if (!defined JDK_PRIMITIVE_ITERATOR || KEY_WIDENED) && KEYS_PRIMITIVE
import java.util.Objects;
#endif
#ifdef KEYS_PRIMITIVE
import java.util.function.Consumer;
#endif

/** A type-specific {@link Iterator}; provides an additional method to avoid (un)boxing, and
* the possibility to skip elements.
*
* @see Iterator
```

```
*/

#if defined JDK_PRIMITIVE_ITERATOR && !KEY_WIDENED
public interface KEY_ITERATOR KEY_GENERIC extends JDK_PRIMITIVE_ITERATOR {
#else
public interface KEY_ITERATOR KEY_GENERIC extends Iterator<KEY_GENERIC_CLASS> {
#endif

#if KEYS_PRIMITIVE
 /**
  * Returns the next element as a primitive type.
  *
  * @return the next element in the iteration.
  * @see Iterator#next()
  */

#if defined JDK_PRIMITIVE_ITERATOR && !KEY_WIDENED
 @Override
#endif
 KEY_TYPE NEXT_KEY();

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 default KEY_CLASS next() {
  return KEY_CLASS.valueOf(NEXT_KEY());
 }

#if !defined JDK_PRIMITIVE_ITERATOR || KEY_WIDENED
 /**
  * Performs the given action for each remaining element until all elements
  * have been processed or the action throws an exception.
  * @param action the action to be performed for each element.
  * @see java.util.Iterator#forEachRemaining(java.util.function.Consumer)
  * @since 8.0.0
  */
 default void forEachRemaining(final KEY_CONSUMER action) {
  Objects.requireNonNull(action);
  while (hasNext()) {
   action.accept(NEXT_KEY());
  }
 }
#endif

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
```

```
 @Override
 default void forEachRemaining(final Consumer<? super KEY_GENERIC_CLASS> action) {
#if defined JDK_PRIMITIVE_ITERATOR && !KEY_WIDENED
  forEachRemaining((JDK_PRIMITIVE_KEY_CONSUMER) action::accept);
#else
  forEachRemaining((KEY_CONSUMER) action::accept);
#endif
 }

#endif

 /** Skips the given number of elements.
  *
  * <p>The effect of this call is exactly the same as that of calling {@link #next()} for {@code n} times (possibly stopping if {@link #hasNext()} becomes false).
  *
  * @param n the number of elements to skip.
  * @return the number of elements actually skipped.
  * @see Iterator#next()
  */

 default int skip(final int n) {
  if (n < 0) throw new IllegalArgumentException("Argument must be nonnegative: " + n);
  int i = n;
  while(i-- != 0 && hasNext()) NEXT_KEY();
  return n - i - 1;
 }
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/Iterator.drv
No license file was found, but licenses were detected in source scan.

```
*/


package PACKAGE;

import it.unimi.dsi.fastutil.IndirectPriorityQueue;

/** A type-specific {@link IndirectPriorityQueue}.
*
* <p>Additionally, this interface strengthens {@link #comparator()}.
*/

public interface INDIRECT_PRIORITY_QUEUE extends IndirectPriorityQueue<KEY_CLASS> {

 /** Returns the type-specific comparator associated with this queue.
  *
  * <p>Note that this specification strengthens the one given in {@link IndirectPriorityQueue}.
  *
  * @return the comparator associated with this queue.
  * @see IndirectPriorityQueue#comparator()
  */
 @Override
 KEY_COMPARATOR comparator();
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/IndirectPriorityQueue.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;
```

```
import java.lang.Iterable;
#if KEYS_PRIMITIVE
import java.util.Objects;
import java.util.function.Consumer;

/** A type-specific {@link Iterable} that strengthens that specification of {@link #iterator()} and {@link
#forEach(Consumer)}.
*
* <p>Note that whenever there exist a primitive consumer in {@link java.util.function} (e.g., {@link
java.util.function.IntConsumer}),
* trying to access any version of {@link #forEach(Consumer)} using a lambda expression with untyped arguments
* will generate an ambiguous method error. This can be easily solved by specifying the type of the argument, as in
* <pre>
*    intIterable.forEach((int x) -&gt; { // Do something with x });
* </pre>
* <p>The same problem plagues, for example, {@link
java.util.PrimitiveIterator.OfInt#forEachRemaining(java.util.function.IntConsumer)}.
*
* <p><strong>Warning</strong>: Java will let you write &ldquo;colon&rdquo; {@code for} statements with
primitive-type
* loop variables; however, what is (unfortunately) really happening is that at each iteration an
* unboxing (and, in the case of {@code fastutil} type-specific data structures, a boxing) will be performed. Watch
out.
*
* @see Iterable
*/

#else

/** A type-specific {@link Iterable} that strengthens that specification of {@link #iterator()}.
*
* @see Iterable
*/

#endif


public interface KEY_ITERABLE KEY_GENERIC extends Iterable<KEY_GENERIC_CLASS> {

 /** Returns a type-specific iterator.
  *
  * <p>Note that this specification strengthens the one given in {@link Iterable#iterator()}.
  *
  * @return a type-specific iterator.
  * @see Iterable#iterator()
  */
```

```
 @Override
 KEY_ITERATOR KEY_GENERIC iterator();


#if KEYS_PRIMITIVE
 /**
  * Performs the given action for each element of this type-specific {@link java.lang.Iterable}
  * until all elements have been processed or the action throws an
  * exception.
  * @param action the action to be performed for each element.
  * @see java.lang.Iterable#forEach(java.util.function.Consumer)
  * @since 8.0.0
  */
#ifdef JDK_PRIMITIVE_KEY_CONSUMER
 @SuppressWarnings("overloads")
 default void forEach(final JDK_PRIMITIVE_KEY_CONSUMER action) {
#else
 default void forEach(final KEY_CONSUMER action) {
#endif
  Objects.requireNonNull(action);
  for(final KEY_ITERATOR iterator = iterator(); iterator.hasNext();)
   action.accept(iterator.NEXT_KEY());
 }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 default void forEach(final Consumer<? super KEY_GENERIC_CLASS> action) {
#ifdef JDK_PRIMITIVE_KEY_CONSUMER
#ifdef KEY_WIDENED
  forEach(new JDK_PRIMITIVE_KEY_CONSUMER() {
   public void accept(final KEY_TYPE_WIDENED key) {
    action.accept(KEY2OBJ((KEY_TYPE)key));
   }
  });
#else
  forEach((JDK_PRIMITIVE_KEY_CONSUMER) action::accept);
#endif
#else
  forEach((KEY_CONSUMER) action::accept);
#endif
 }
#endif

}
```

Found in path(s):

* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/Iterable.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import java.util.Arrays;
import java.util.Collection;
import java.util.Iterator;
import java.util.RandomAccess;
import java.util.NoSuchElementException;

#if KEYS_PRIMITIVE

/** A type-specific array-based list; provides some additional methods that use polymorphism to avoid (un)boxing.
*
* <p>This class implements a lightweight, fast, open, optimized,
* reuse-oriented version of array-based lists. Instances of this class
* represent a list with an array that is enlarged as needed when new entries
* are created (by doubling its current length), but is
* <em>never</em> made smaller (even on a {@link #clear()}). A family of
* {@linkplain #trim() trimming methods} lets you control the size of the
* backing array; this is particularly useful if you reuse instances of this class.
* Range checks are equivalent to those of {@link java.util}'s classes, but
* they are delayed as much as possible. The backing array is exposed by the
* {@link #elements()} method.
*
* <p>This class implements the bulk methods {@code removeElements()},
* {@code addElements()} and {@code getElements()} using
* high-performance system calls (e.g., {@link
* System#arraycopy(Object,int,Object,int,int) System.arraycopy()} instead of
```

```
* expensive loops.
*
* @see java.util.ArrayList
*/

public class ARRAY_LIST KEY_GENERIC extends ABSTRACT_LIST KEY_GENERIC implements
RandomAccess, Cloneable, java.io.Serializable {
 private static final long serialVersionUID = -7046029254386353130L;


#else

/** A type-specific array-based list; provides some additional methods that use polymorphism to avoid (un)boxing.
*
* <p>This class implements a lightweight, fast, open, optimized,
* reuse-oriented version of array-based lists. Instances of this class
* represent a list with an array that is enlarged as needed when new entries
* are created (by doubling the current length), but is
* <em>never</em> made smaller (even on a {@link #clear()}). A family of
* {@linkplain #trim() trimming methods} lets you control the size of the
* backing array; this is particularly useful if you reuse instances of this class.
* Range checks are equivalent to those of {@link java.util}'s classes, but
* they are delayed as much as possible.
*
* <p>The backing array is exposed by the {@link #elements()} method. If an instance
* of this class was created {@linkplain #wrap(Object[],int) by wrapping},
* backing-array reallocations will be performed using reflection, so that
* {@link #elements()} can return an array of the same type of the original array: the comments
* about efficiency made in {@link it.unimi.dsi.fastutil.objects.ObjectArrays} apply here.
* Moreover, you must take into consideration that assignment to an array
* not of type {@code Object[]} is slower due to type checking.
*
* <p>This class implements the bulk methods {@code removeElements()},
* {@code addElements()} and {@code getElements()} using
* high-performance system calls (e.g., {@link
* System#arraycopy(Object,int,Object,int,int) System.arraycopy()} instead of
* expensive loops.
*
* @see java.util.ArrayList
*/

public class ARRAY_LIST KEY_GENERIC extends ABSTRACT_LIST KEY_GENERIC implements
RandomAccess, Cloneable, java.io.Serializable {
 private static final long serialVersionUID = -7046029254386353131L;


#endif
```

```
 /** The initial default capacity of an array list. */
 public static final int DEFAULT_INITIAL_CAPACITY = 10;

#if ! KEYS_PRIMITIVE
 /** Whether the backing array was passed to {@code wrap()}. In
  * this case, we must reallocate with the same type of array. */
 protected final boolean wrapped;
#endif

 /** The backing array. */
 protected transient KEY_GENERIC_TYPE a[];

 /** The current actual size of the list (never greater than the backing-array length). */
 protected int size;

 /** Creates a new array list using a given array.
  *
  * <p>This constructor is only meant to be used by the wrapping methods.
  *
  * @param a the array that will be used to back this array list.
  */

 protected ARRAY_LIST(final KEY_GENERIC_TYPE a[], @SuppressWarnings("unused") boolean dummy) {
  this.a = a;
#if ! KEYS_PRIMITIVE
  this.wrapped = true;
#endif
 }

 /** Creates a new array list with given capacity.
  *
  * @param capacity the initial capacity of the array list (may be 0).
  */

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public ARRAY_LIST(final int capacity) {
  if (capacity < 0) throw new IllegalArgumentException("Initial capacity (" + capacity + ") is negative");
  if (capacity == 0) a = KEY_GENERIC_ARRAY_CAST ARRAYS.EMPTY_ARRAY;
  else a = KEY_GENERIC_ARRAY_CAST new KEY_TYPE[capacity];
#if ! KEYS_PRIMITIVE
  wrapped = false;
#endif
 }

 /** Creates a new array list with {@link #DEFAULT_INITIAL_CAPACITY} capacity. */

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public ARRAY_LIST() {
```

```
 a = KEY_GENERIC_ARRAY_CAST ARRAYS.DEFAULT_EMPTY_ARRAY; // We delay allocation
#if ! KEYS_PRIMITIVE
 wrapped = false;
#endif
 }

 /** Creates a new array list and fills it with a given collection.
  *
  * @param c a collection that will be used to fill the array list.
  */

 public ARRAY_LIST(final Collection<? extends KEY_GENERIC_CLASS> c) {
  this(c.size());
#if KEYS_PRIMITIVE
  size = ITERATORS.unwrap(ITERATORS.AS_KEY_ITERATOR(c.iterator()), a);
#else
  size = ITERATORS.unwrap(c.iterator(), a);
#endif
 }

 /** Creates a new array list and fills it with a given type-specific collection.
  *
  * @param c a type-specific collection that will be used to fill the array list.
  */

 public ARRAY_LIST(final COLLECTION KEY_EXTENDS_GENERIC c) {
  this(c.size());
  size = ITERATORS.unwrap(c.iterator(), a);
 }

 /** Creates a new array list and fills it with a given type-specific list.
  *
  * @param l a type-specific list that will be used to fill the array list.
  */

 public ARRAY_LIST(final LIST KEY_EXTENDS_GENERIC l) {
  this(l.size());
  l.getElements(0, a, 0, size = l.size());
 }

 /** Creates a new array list and fills it with the elements of a given array.
  *
  * @param a an array whose elements will be used to fill the array list.
  */

 public ARRAY_LIST(final KEY_GENERIC_TYPE a[]) {
  this(a, 0, a.length);
 }
```

```
 /** Creates a new array list and fills it with the elements of a given array.
  *
  * @param a an array whose elements will be used to fill the array list.
  * @param offset the first element to use.
  * @param length the number of elements to use.
  */

 public ARRAY_LIST(final KEY_GENERIC_TYPE a[], final int offset, final int length) {
  this(length);
  System.arraycopy(a, offset, this.a, 0, length);
  size = length;
 }

 /** Creates a new array list and fills it with the elements returned by an iterator..
  *
  * @param i an iterator whose returned elements will fill the array list.
  */

 public ARRAY_LIST(final Iterator<? extends KEY_GENERIC_CLASS> i) {
  this();
  while(i.hasNext()) this.add(KEY_CLASS2TYPE(i.next()));
 }

 /** Creates a new array list and fills it with the elements returned by a type-specific iterator..
  *
  * @param i a type-specific iterator whose returned elements will fill the array list.
  */

 public ARRAY_LIST(final KEY_ITERATOR KEY_EXTENDS_GENERIC i) {
  this();
  while(i.hasNext()) this.add(i.NEXT_KEY());
 }

#if KEYS_PRIMITIVE
 /** Returns the backing array of this list.
  *
  * @return the backing array.
  */

 public KEY_GENERIC_TYPE[] elements() {
  return a;
 }
#else
 /** Returns the backing array of this list.
  *
  * <p>If this array list was created by wrapping a given array, it is guaranteed
  * that the type of the returned array will be the same. Otherwise, the returned
```

```
 * array will be of type {@link Object Object[]} (in spite of the declared return type).
 *
 * <p><strong>Warning</strong>: This behaviour may cause (unfathomable)
 * run-time errors if a method expects an array
 * actually of type {@code K[]}, but this methods returns an array
 * of type {@link Object Object[]}.
 *
 * @return the backing array.
 */

 public K[] elements() {
  return a;
 }
#endif

 /** Wraps a given array into an array list of given size.
 *
 * <p>Note it is guaranteed
 * that the type of the array returned by {@link #elements()} will be the same
 * (see the comments in the class documentation).
 *
 * @param a an array to wrap.
 * @param length the length of the resulting array list.
 * @return a new array list of the given size, wrapping the given array.
 */

 public static KEY_GENERIC ARRAY_LIST KEY_GENERIC wrap(final KEY_GENERIC_TYPE a[], final int
length) {
  if (length > a.length) throw new IllegalArgumentException("The specified length (" + length + ") is greater than the
array size (" + a.length + ")");
  final ARRAY_LIST KEY_GENERIC l = new ARRAY_LIST KEY_GENERIC_DIAMOND(a, false);
  l.size = length;
  return l;
 }

 /** Wraps a given array into an array list.
 *
 * <p>Note it is guaranteed
 * that the type of the array returned by {@link #elements()} will be the same
 * (see the comments in the class documentation).
 *
 * @param a an array to wrap.
 * @return a new array list wrapping the given array.
 */

 public static KEY_GENERIC ARRAY_LIST KEY_GENERIC wrap(final KEY_GENERIC_TYPE a[]) {
  return wrap(a, a.length);
 }
```

```
 /** Ensures that this array list can contain the given number of entries without resizing.
  *
  * @param capacity the new minimum capacity for this array list.
  */
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public void ensureCapacity(final int capacity) {
  if (capacity <= a.length || (a == ARRAYS.DEFAULT_EMPTY_ARRAY && capacity <=
DEFAULT_INITIAL_CAPACITY)) return;
#if KEYS_PRIMITIVE
  a = ARRAYS.ensureCapacity(a, capacity, size);
#else
  if (wrapped) a = ARRAYS.ensureCapacity(a, capacity, size);
  else {
   if (capacity > a.length) {
    final Object t[] = new Object[capacity];
    System.arraycopy(a, 0, t, 0, size);
    a = (KEY_GENERIC_TYPE[])t;
   }
  }
#endif
  assert size <= a.length;
 }

 /** Grows this array list, ensuring that it can contain the given number of entries without resizing,
  * and in case increasing the current capacity at least by a factor of 50%.
  *
  * @param capacity the new minimum capacity for this array list.
  */
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 private void grow(int capacity) {
  if (capacity <= a.length) return;
  if (a != ARRAYS.DEFAULT_EMPTY_ARRAY)
   capacity = (int)Math.max(Math.min((long)a.length + (a.length >> 1),
it.unimi.dsi.fastutil.Arrays.MAX_ARRAY_SIZE), capacity);
  else if (capacity < DEFAULT_INITIAL_CAPACITY) capacity = DEFAULT_INITIAL_CAPACITY;
#if KEYS_PRIMITIVE
  a = ARRAYS.forceCapacity(a, capacity, size);
#else
  if (wrapped) a = ARRAYS.forceCapacity(a, capacity, size);
  else {
   final Object t[] = new Object[capacity];
   System.arraycopy(a, 0, t, 0, size);
   a = (KEY_GENERIC_TYPE[])t;
  }
#endif
  assert size <= a.length;
```

```
 }

 @Override
 public void add(final int index, final KEY_GENERIC_TYPE k) {
  ensureIndex(index);
  grow(size + 1);
  if (index != size) System.arraycopy(a, index, a, index + 1, size - index);
  a[index] = k;
  size++;
  assert size <= a.length;
 }

 @Override
 public boolean add(final KEY_GENERIC_TYPE k) {
  grow(size + 1);
  a[size++] = k;
  assert size <= a.length;
  return true;
 }

 @Override
 public KEY_GENERIC_TYPE GET_KEY(final int index) {
  if (index >= size) throw new IndexOutOfBoundsException("Index (" + index + ") is greater than or equal to list size
(" + size + ")");
  return a[index];
 }

 @Override
 public int indexOf(final KEY_TYPE k) {
  for(int i = 0; i < size; i++) if (KEY_EQUALS(k, a[i])) return i;
  return -1;
 }


 @Override
 public int lastIndexOf(final KEY_TYPE k) {
  for(int i = size; i-- != 0;) if (KEY_EQUALS(k, a[i])) return i;
  return -1;
 }

 @Override
 public KEY_GENERIC_TYPE REMOVE_KEY(final int index) {
  if (index >= size) throw new IndexOutOfBoundsException("Index (" + index + ") is greater than or equal to list size
(" + size + ")");
  final KEY_GENERIC_TYPE old = a[index];
  size--;
  if (index != size) System.arraycopy(a, index + 1, a, index, size - index);
#if KEYS_REFERENCE
```

```
  a[size] = null;
#endif
  assert size <= a.length;
  return old;
  }

  @Override
  public boolean REMOVE(final KEY_TYPE k) {
  int index = indexOf(k);
  if (index == -1) return false;
  REMOVE_KEY(index);
  assert size <= a.length;
  return true;
  }

  @Override
  public KEY_GENERIC_TYPE set(final int index, final KEY_GENERIC_TYPE k) {
  if (index >= size) throw new IndexOutOfBoundsException("Index (" + index + ") is greater than or equal to list size
(" + size + ")");
  KEY_GENERIC_TYPE old = a[index];
  a[index] = k;
  return old;
  }

  @Override
  public void clear() {
#if KEYS_REFERENCE
  Arrays.fill(a, 0, size, null);
#endif
  size = 0;
  assert size <= a.length;
  }

  @Override
  public int size() {
  return size;
  }

  @Override
  public void size(final int size) {
  if (size > a.length) a = ARRAYS.forceCapacity(a, size, this.size);
  if (size > this.size) Arrays.fill(a, this.size, size, KEY_NULL);
#if KEYS_REFERENCE
  else Arrays.fill(a, size, this.size, KEY_NULL);
#endif
  this.size = size;
  }
```

```
@Override
public boolean isEmpty() {
 return size == 0;
}

/** Trims this array list so that the capacity is equal to the size.
 *
 * @see java.util.ArrayList#trimToSize()
 */
public void trim() {
 trim(0);
}

/** Trims the backing array if it is too large.
 *
 * If the current array length is smaller than or equal to
 * {@code n}, this method does nothing. Otherwise, it trims the
 * array length to the maximum between {@code n} and {@link #size()}.
 *
 * <p>This method is useful when reusing lists.  {@linkplain #clear() Clearing a
 * list} leaves the array length untouched. If you are reusing a list
 * many times, you can call this method with a typical
 * size to avoid keeping around a very large array just
 * because of a few large transient lists.
 *
 * @param n the threshold for the trimming.
 */

SUPPRESS_WARNINGS_KEY_UNCHECKED
public void trim(final int n) {
 // TODO: use Arrays.trim() and preserve type only if necessary
 if (n >= a.length || size == a.length) return;
 final KEY_GENERIC_TYPE t[] = KEY_GENERIC_ARRAY_CAST new KEY_TYPE[Math.max(n, size)];
 System.arraycopy(a, 0, t, 0, size);
 a = t;
 assert size <= a.length;
}


  /** Copies element of this type-specific list into the given array using optimized system calls.
 *
 * @param from the start index (inclusive).
 * @param a the destination array.
 * @param offset the offset into the destination array where to store the first element copied.
 * @param length the number of elements to be copied.
 */
@Override
public void getElements(final int from, final KEY_TYPE[] a, final int offset, final int length) {
```

```
 ARRAYS.ensureOffsetLength(a, offset, length);
 System.arraycopy(this.a, from, a, offset, length);
 }

 /** Removes elements of this type-specific list using optimized system calls.
  *
  * @param from the start index (inclusive).
  * @param to the end index (exclusive).
  */
 @Override
 public void removeElements(final int from, final int to) {
  it.unimi.dsi.fastutil.Arrays.ensureFromTo(size, from, to);
  System.arraycopy(a, to, a, from, size - to);
  size -= (to - from);
#if KEYS_REFERENCE
  int i = to - from;
  while(i-- != 0) a[size + i] = null;
#endif
 }


 /** Adds elements to this type-specific list using optimized system calls.
  *
  * @param index the index at which to add elements.
  * @param a the array containing the elements.
  * @param offset the offset of the first element to add.
  * @param length the number of elements to add.
  */
 @Override
 public void addElements(final int index, final KEY_GENERIC_TYPE a[], final int offset, final int length) {
  ensureIndex(index);
  ARRAYS.ensureOffsetLength(a, offset, length);
  grow(size + length);
  System.arraycopy(this.a, index, this.a, index + length, size - index);
  System.arraycopy(a, offset, this.a, index, length);
  size += length;
 }



#if KEYS_PRIMITIVE

 @Override
 public KEY_TYPE[] toArray(KEY_TYPE a[]) {
  if (a == null || a.length < size) a = new KEY_TYPE[size];
  System.arraycopy(this.a, 0, a, 0, size);
  return a;
 }
```

```
 @Override
 public boolean addAll(int index, final COLLECTION c) {
  ensureIndex(index);
  int n = c.size();
  if (n == 0) return false;
  grow(size + n);
  if (index != size) System.arraycopy(a, index, a, index + n, size - index);
  final KEY_ITERATOR i = c.iterator();
  size += n;
  while(n-- != 0) a[index++] = i.NEXT_KEY();
  assert size <= a.length;
  return true;
 }

 @Override
 public boolean addAll(final int index, final LIST l) {
  ensureIndex(index);
  final int n = l.size();
  if (n == 0) return false;
  grow(size + n);
  if (index != size) System.arraycopy(a, index, a, index + n, size - index);
  l.getElements(0, a, index, n);
  size += n;
  assert size <= a.length;
  return true;
 }

 @Override
 public boolean removeAll(final COLLECTION c) {
  final KEY_TYPE[] a = this.a;
  int j = 0;
  for(int i = 0; i < size; i++)
   if (! c.contains(a[i])) a[j++] = a[i];
#if KEYS_REFERENCE
  Arrays.fill(a, j, size, null);
#endif
  final boolean modified = size != j;
  size = j;
  return modified;
 }

#endif

 @Override
 public boolean removeAll(final Collection<?> c) {
  final KEY_TYPE[] a = this.a;
  int j = 0;
```

```
  for(int i = 0; i < size; i++)
   if (! c.contains(KEY2OBJ(a[i]))) a[j++] = a[i];
#if KEYS_REFERENCE
  Arrays.fill(a, j, size, null);
#endif
  final boolean modified = size != j;
  size = j;
  return modified;
 }

 @Override
 public KEY_LIST_ITERATOR KEY_GENERIC listIterator(final int index) {
  ensureIndex(index);

  return new KEY_LIST_ITERATOR KEY_GENERIC() {
    int pos = index, last = -1;

    @Override
    public boolean hasNext() { return pos < size; }
    @Override
    public boolean hasPrevious() { return pos > 0; }
    @Override
    public KEY_GENERIC_TYPE NEXT_KEY() { if (! hasNext()) throw new NoSuchElementException(); return
a[last = pos++]; }
    @Override
    public KEY_GENERIC_TYPE PREV_KEY() { if (! hasPrevious()) throw new NoSuchElementException(); return
a[last = --pos]; }
    @Override
    public int nextIndex() { return pos; }
    @Override
    public int previousIndex() { return pos - 1; }
    @Override
    public void add(KEY_GENERIC_TYPE k) {
     ARRAY_LIST.this.add(pos++, k);
     last = -1;
    }
    @Override
    public void set(KEY_GENERIC_TYPE k) {
     if (last == -1) throw new IllegalStateException();
     ARRAY_LIST.this.set(last, k);
    }
    @Override
    public void remove() {
     if (last == -1) throw new IllegalStateException();
     ARRAY_LIST.this.REMOVE_KEY(last);
     /* If the last operation was a next(), we are removing an element *before* us, and we must decrease pos
correspondingly. */
     if (last < pos) pos--;
```

```
    last = -1;
   }
  };
 }

 @Override
 public ARRAY_LIST KEY_GENERIC clone() {
  ARRAY_LIST KEY_GENERIC c = new ARRAY_LIST KEY_GENERIC_DIAMOND(size);
  System.arraycopy(a, 0, c.a, 0, size);
  c.size = size;
  return c;
 }

#if KEY_CLASS_Object
 private boolean valEquals(final K a, final K b) { return a == null ? b == null : a.equals(b); }
#endif

 /** Compares this type-specific array list to another one.
  *
  * <p>This method exists only for sake of efficiency. The implementation
  * inherited from the abstract implementation would already work.
  *
  * @param l a type-specific array list.
  * @return true if the argument contains the same elements of this type-specific array list.
  */
 public boolean equals(final ARRAY_LIST KEY_GENERIC l) {
  if (l == this) return true;
  int s = size();
  if (s != l.size()) return false;
  final KEY_GENERIC_TYPE[] a1 = a;
  final KEY_GENERIC_TYPE[] a2 = l.a;

#if KEY_CLASS_Object
  while(s-- !=  0) if (! valEquals(a1[s], a2[s])) return false;
#else
  while(s-- !=  0) if (a1[s] != a2[s]) return false;
#endif
  return true;
 }


#if ! KEY_CLASS_Reference

 /** Compares this array list to another array list.
  *
  * <p>This method exists only for sake of efficiency. The implementation
  * inherited from the abstract implementation would already work.
  *
```

```
 * @param l an array list.
 * @return a negative integer,
 * zero, or a positive integer as this list is lexicographically less than, equal
 * to, or greater than the argument.
 */
SUPPRESS_WARNINGS_KEY_UNCHECKED
public int compareTo(final ARRAY_LIST KEY_EXTENDS_GENERIC l) {
 final int s1 = size(), s2 = l.size();
 final KEY_GENERIC_TYPE a1[] = a, a2[] = l.a;
 KEY_GENERIC_TYPE e1, e2;
 int r, i;

 for(i = 0; i < s1 && i < s2; i++) {
  e1 = a1[i];
  e2 = a2[i];
  if ((r = KEY_CMP(e1, e2)) != 0) return r;
 }

 return i < s2 ? -1 : (i < s1 ? 1 : 0);
}
#endif


 private void writeObject(java.io.ObjectOutputStream s) throws java.io.IOException {
  s.defaultWriteObject();
  for(int i = 0; i < size; i++) s.WRITE_KEY(a[i]);
 }

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 private void readObject(java.io.ObjectInputStream s) throws java.io.IOException, ClassNotFoundException {
  s.defaultReadObject();
  a = KEY_GENERIC_ARRAY_CAST new KEY_TYPE[size];
  for(int i = 0; i < size; i++) a[i] = KEY_GENERIC_CAST s.READ_KEY();
 }


#ifdef TEST

 private static long seed = System.currentTimeMillis();
 private static java.util.Random r = new java.util.Random(seed);

 private static KEY_TYPE genKey() {
#if KEY_CLASS_Byte || KEY_CLASS_Short || KEY_CLASS_Character
  return (KEY_TYPE)(r.nextInt());
#elif KEYS_PRIMITIVE
  return r.NEXT_KEY();
#elif KEY_CLASS_Object
  return Integer.toBinaryString(r.nextInt());
```

```
#else
  return new java.io.Serializable() {};
#endif
 }

 private static java.text.NumberFormat format = new java.text.DecimalFormat("#,###.00");
 private static java.text.FieldPosition p = new java.text.FieldPosition(0);

 private static String format(double d) {
  StringBuffer s = new StringBuffer();
  return format.format(d, s, p).toString();
 }

 private static void speedTest(int n, boolean comp) {
  System.out.println("There are presently no speed tests for this class.");
 }


 private static void fatal(String msg) {
  System.out.println(msg);
  System.exit(1);
 }

 private static void ensure(boolean cond, String msg) {
  if (cond) return;
  fatal(msg);
 }

 private static Object[] k, v, nk;
 private static KEY_TYPE kt[];
 private static KEY_TYPE nkt[];
 private static ARRAY_LIST topList;

 protected static void testLists(LIST m, java.util.List t, int n, int level) {
  long ms;
  Exception mThrowsIllegal, tThrowsIllegal, mThrowsOutOfBounds, tThrowsOutOfBounds;
  Object rt = null;
  KEY_TYPE rm = KEY_NULL;

  if (level > 4) return;


  /* Now we check that both sets agree on random keys. For m we use the polymorphic method. */

  for(int i = 0; i < n; i++) {
   int p = r.nextInt() % (n * 2);

   KEY_TYPE T = genKey();
```

```
  mThrowsOutOfBounds = tThrowsOutOfBounds  = null;

  try {
   m.set(p, T);
  }
  catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }
  try {
   t.set(p, KEY2OBJ(T));
  }
  catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }

  ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + level + ", " + seed + "): set() divergence at start in IndexOutOfBoundsException for index " + p + "  (" + mThrowsOutOfBounds + ", " + tThrowsOutOfBounds + ")");
  if (mThrowsOutOfBounds == null) ensure(t.get(p).equals(KEY2OBJ(m.GET_KEY(p))), "Error (" + level + ", " + seed + "): m and t differ after set() on position " + p + " (" + m.GET_KEY(p) + ", " + t.get(p) + ")");

  p = r.nextInt() % (n * 2);
  mThrowsOutOfBounds = tThrowsOutOfBounds  = null;

  try {
   m.GET_KEY(p);
  }
  catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }
  try {
   t.get(p);
  }
  catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }

  ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + level + ", " + seed + "): get() divergence at start in IndexOutOfBoundsException for index " + p + "  (" + mThrowsOutOfBounds + ", " + tThrowsOutOfBounds + ")");
  if (mThrowsOutOfBounds == null) ensure(t.get(p).equals(KEY2OBJ(m.GET_KEY(p))), "Error (" + level + ", " + seed + "): m and t differ aftre get() on position " + p + " (" + m.GET_KEY(p) + ", " + t.get(p) + ")");

 }

 /* Now we check that both sets agree on random keys. For m we use the standard method. */

 for(int i = 0; i < n; i++) {
  int p = r.nextInt() % (n * 2);

  mThrowsOutOfBounds = tThrowsOutOfBounds  = null;

  try {
   m.get(p);
  }
  catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }
```

```
  try {
   t.get(p);
  }
  catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }

  ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + level + ", " + seed + "):
get() divergence at start in IndexOutOfBoundsException for index " + p + "  (" + mThrowsOutOfBounds + ", " +
tThrowsOutOfBounds + ")");
  if (mThrowsOutOfBounds == null) ensure(t.get(p).equals(m.get(p)), "Error (" + level + ", " + seed + "): m and t
differ at start on position " + p + " (" + m.get(p) + ", " + t.get(p) + ")");

 }

 /* Now we check that m and t are equal. */
 if (!m.equals(t) || ! t.equals(m)) System.err.println("m: " + m + " t: " + t);

 ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) at start");
 ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) at start");



 /* Now we check that m actually holds that data. */
 for(Iterator i=t.iterator(); i.hasNext();) {
  ensure(m.contains(i.next()), "Error (" + level + ", " + seed + "): m and t differ on an entry after insertion (iterating on
t)");
 }

 /* Now we check that m actually holds that data, but iterating on m. */
 for(Iterator i=m.listIterator(); i.hasNext();) {
  ensure(t.contains(i.next()), "Error (" + level + ", " + seed + "): m and t differ on an entry after insertion (iterating on
m)");
 }

 /* Now we check that inquiries about random data give the same answer in m and t. For
    m we use the polymorphic method. */

 for(int i=0; i<n;  i++) {
  KEY_TYPE T = genKey();
  ensure(m.contains(T) == t.contains(KEY2OBJ(T)), "Error (" + level + ", " + seed + "): divergence in content
between t and m (polymorphic method)");
 }

 /* Again, we check that inquiries about random data give the same answer in m and t, but
    for m we use the standard method. */

 for(int i=0; i<n;  i++) {
  KEY_TYPE T = genKey();
  ensure(m.contains(KEY2OBJ(T)) == t.contains(KEY2OBJ(T)), "Error (" + level + ", " + seed + "): divergence in
```

```
content between t and m (polymorphic method)");
  }

  /* Now we add and remove random data in m and t, checking that the result is the same. */

  for(int i=0; i<2*n;  i++) {
   KEY_TYPE T = genKey();

   try {
    m.add(T);
   }
   catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }

   try {
    t.add(KEY2OBJ(T));
   }
   catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }

   T = genKey();
   int p = r.nextInt() % (2 * n + 1);

   mThrowsOutOfBounds = tThrowsOutOfBounds  = null;

   try {
    m.add(p, T);
   }
   catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }

   try {
    t.add(p, KEY2OBJ(T));
   }
   catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }


   ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + level + ", " + seed + "):
add() divergence in IndexOutOfBoundsException for index " + p + " for " + T + " (" + mThrowsOutOfBounds + ", "
+ tThrowsOutOfBounds + ")");

   p = r.nextInt() % (2 * n + 1);

   mThrowsOutOfBounds = tThrowsOutOfBounds  = null;

   try {
    rm = m.REMOVE_KEY(p);
   }
   catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }

   try {
```

```
   rt = t.remove(p);
   }
   catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }


   ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + level + ", " + seed + "):
remove() divergence in IndexOutOfBoundsException for index " + p + " (" + mThrowsOutOfBounds + ", " +
tThrowsOutOfBounds + ")");
   if (mThrowsOutOfBounds == null) ensure(rt.equals(KEY2OBJ(rm)), "Error (" + level + ", " + seed + "): divergence
in remove() between t and m (" + rt + ", " + rm + ")");
   }

   ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after add/remove");
   ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after add/remove");

   /* Now we add random data in m and t using addAll on a collection, checking that the result is the same. */

   for(int i=0; i<n;  i++) {
    int p = r.nextInt() % (2 * n + 1);
    Collection m1 = new java.util.ArrayList();
    int s = r.nextInt(n / 2 + 1);
    for(int j = 0; j < s; j++) m1.add(KEY2OBJ(genKey()));

    mThrowsOutOfBounds = tThrowsOutOfBounds  = null;

    try {
     m.addAll(p, m1);
    }
    catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }

    try {
     t.addAll(p, m1);
    }
    catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }

    ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + level + ", " + seed + "):
addAll() divergence in IndexOutOfBoundsException for index " + p + " for " + m1 + " (" + mThrowsOutOfBounds
+ ", " + tThrowsOutOfBounds + ")");

    ensure(m.equals(t), "Error (" + level + ", " + seed + m + t + "): ! m.equals(t) after addAll");
    ensure(t.equals(m), "Error (" + level + ", " + seed + m + t + "): ! t.equals(m) after addAll");
    }

   if (m.size() > n) {
    m.size(n);
    while(t.size() != n) t.remove(t.size() -1);
    }
```

```
  /* Now we add random data in m and t using addAll on a type-specific collection, checking that the result is the same. */

  for(int i=0; i<n;  i++) {
   int p = r.nextInt() % (2 * n + 1);
   COLLECTION m1 = new ARRAY_LIST();
   Collection t1 = new java.util.ArrayList();
   int s = r.nextInt(n / 2 + 1);
   for(int j = 0; j < s; j++) {
    KEY_TYPE x = genKey();
    m1.add(x);
    t1.add(KEY2OBJ(x));
   }

   mThrowsOutOfBounds = tThrowsOutOfBounds  = null;

   try {
    m.addAll(p, m1);
   }
   catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }

   try {
    t.addAll(p, t1);
   }
   catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }

   ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + level + ", " + seed + "): polymorphic addAll() divergence in IndexOutOfBoundsException for index " + p + " for " + m1 + " (" + mThrowsOutOfBounds + ", " + tThrowsOutOfBounds + ")");

   ensure(m.equals(t), "Error (" + level + ", " + seed + m + t + "): ! m.equals(t) after polymorphic addAll");
   ensure(t.equals(m), "Error (" + level + ", " + seed + m + t + "): ! t.equals(m) after polymorphic addAll");
  }

  if (m.size() > n) {
   m.size(n);
   while(t.size() != n) t.remove(t.size() -1);
  }

  /* Now we add random data in m and t using addAll on a list, checking that the result is the same. */

  for(int i=0; i<n;  i++) {
   int p = r.nextInt() % (2 * n + 1);
   LIST m1 = new ARRAY_LIST();
   Collection t1 = new java.util.ArrayList();
   int s = r.nextInt(n / 2 + 1);
   for(int j = 0; j < s; j++) {
    KEY_TYPE x = genKey();
```

```
 m1.add(x);
 t1.add(KEY2OBJ(x));
 }

 mThrowsOutOfBounds = tThrowsOutOfBounds  = null;

 try {
  m.addAll(p, m1);
 }
 catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }

 try {
  t.addAll(p, t1);
 }
 catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }

 ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + level + ", " + seed + "): list
addAll() divergence in IndexOutOfBoundsException for index " + p + " for " + m1 + " (" + mThrowsOutOfBounds
+ ", " + tThrowsOutOfBounds + ")");

 ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after list addAll");
 ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after list addAll");
 }

 /* Now we add random data in m and t using addElements, checking that the result is the same. */

 for(int i=0; i<n;  i++) {
 int p = r.nextInt() % (2 * n + 1);
 Collection t1 = new java.util.ArrayList();
 int s = r.nextInt(n / 2 + 1);
 KEY_TYPE a[] = new KEY_TYPE [s];
 for(int j = 0; j < s; j++) {
  KEY_TYPE x = genKey();
  t1.add(KEY2OBJ(x));
  a[j] = x;
 }

 mThrowsOutOfBounds = tThrowsOutOfBounds  = null;

 try {
  m.addElements(p, a);
 }
 catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }

 try {
  t.addAll(p, t1);
 }
 catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }
```

```
  ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + level + ", " + seed + "): list
addElements() divergence in IndexOutOfBoundsException for index " + p + " for " + t1 + " (" +
mThrowsOutOfBounds + ", " + tThrowsOutOfBounds + ")");

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after list addElements");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after list addElements");
  }

  if (m.size() > n) {
  m.size(n);
  while(t.size() != n) t.remove(t.size() -1);
  }

  /* Now we check that m actually holds the same data. */

  for(Iterator i=t.iterator(); i.hasNext();) {
  ensure(m.contains(i.next()), "Error (" + level + ", " + seed + "): m and t differ on an entry after removal (iterating on
t)");
  }

  /* Now we check that m actually holds that data, but iterating on m. */

  for(Iterator i=m.listIterator(); i.hasNext();) {
  ensure(t.contains(i.next()), "Error (" + level + ", " + seed + "): m and t differ on an entry after removal (iterating on
m)");
  }

  /* Now we check that both sets agree on random keys. For m we use the standard method. */

  for(int i = 0; i < n; i++) {
  int p = r.nextInt() % (n * 2);

  mThrowsOutOfBounds = tThrowsOutOfBounds  = null;

  try {
   m.get(p);
  }
  catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }
  try {
   t.get(p);
  }
  catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }

  ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + level + ", " + seed + "):
get() divergence in IndexOutOfBoundsException for index " + p + "  (" + mThrowsOutOfBounds + ", " +
tThrowsOutOfBounds + ")");
  if (mThrowsOutOfBounds == null) ensure(t.get(p).equals(m.get(p)), "Error (" + level + ", " + seed + "): m and t
```

```
differ on position " + p + " (" + m.get(p) + ", " + t.get(p) +")");

  }

  /* Now we inquiry about the content with indexOf()/lastIndexOf(). */

  for(int i=0; i<10*n;  i++) {
   KEY_TYPE T = genKey();
   ensure(m.indexOf(KEY2OBJ(T)) == t.indexOf(KEY2OBJ(T)),
     "Error (" + level + ", " + seed + "): indexOf() divergence for " + T + "  (" + m.indexOf(KEY2OBJ(T)) + ", " +
t.indexOf(KEY2OBJ(T)) + ")");
   ensure(m.lastIndexOf(KEY2OBJ(T)) == t.lastIndexOf(KEY2OBJ(T)),
     "Error (" + level + ", " + seed + "): lastIndexOf() divergence for " + T + "  (" + m.lastIndexOf(KEY2OBJ(T)) + ", "
+ t.lastIndexOf(KEY2OBJ(T)) + ")");
   ensure(m.indexOf(T) == t.indexOf(KEY2OBJ(T)),
     "Error (" + level + ", " + seed + "): polymorphic indexOf() divergence for " + T + "  (" + m.indexOf(T) + ", " +
t.indexOf(KEY2OBJ(T)) + ")");
   ensure(m.lastIndexOf(T) == t.lastIndexOf(KEY2OBJ(T)),
     "Error (" + level + ", " + seed + "): polymorphic lastIndexOf() divergence for " + T + "  (" + m.lastIndexOf(T) + ", "
+ t.lastIndexOf(KEY2OBJ(T)) + ")");
  }

  /* Now we check cloning. */

  if (level == 0) {
   ensure(m.equals(((ARRAY_LIST)m).clone()), "Error (" + level + ", " + seed + "): m does not equal m.clone()");
   ensure(((ARRAY_LIST)m).clone().equals(m), "Error (" + level + ", " + seed + "): m.clone() does not equal m");
  }

  /* Now we play with constructors. */
  ensure(m.equals(new ARRAY_LIST((Collection)m)), "Error (" + level + ", " + seed + "): m does not equal new
(Collection m)");
  ensure((new ARRAY_LIST((Collection)m)).equals(m), "Error (" + level + ", " + seed + "): new (Collection m)does
not equal m");
  ensure(m.equals(new ARRAY_LIST((COLLECTION)m)), "Error (" + level + ", " + seed + "): m does not equal
new (type-specific Collection m)");
  ensure((new ARRAY_LIST((COLLECTION)m)).equals(m), "Error (" + level + ", " + seed + "): new (type-specific
Collection m) does not equal m");
  ensure(m.equals(new ARRAY_LIST((LIST)m)), "Error (" + level + ", " + seed + "): m does not equal new (type-
specific List m)");
  ensure((new ARRAY_LIST((LIST)m)).equals(m), "Error (" + level + ", " + seed + "): new (type-specific List m)
does not equal m");
  ensure(m.equals(new ARRAY_LIST(m.listIterator())), "Error (" + level + ", " + seed + "): m does not equal new
(m.listIterator())");
  ensure((new ARRAY_LIST(m.listIterator())).equals(m), "Error (" + level + ", " + seed + "): new (m.listIterator())
does not equal m");
  ensure(m.equals(new ARRAY_LIST(m.iterator())), "Error (" + level + ", " + seed + "): m does not equal new
(m.type_specific_iterator())");
```

```
  ensure((new ARRAY_LIST(m.iterator())).equals(m), "Error (" + level + ", " + seed + "): new
(m.type_specific_iterator()) does not equal m");

  /* Now we play with conversion to array, wrapping and copying. */
  ensure(m.equals(new ARRAY_LIST(m.TO_KEY_ARRAY())), "Error (" + level + ", " + seed + "): m does not
equal new (toArray(m))");
  ensure((new ARRAY_LIST(m.TO_KEY_ARRAY())).equals(m), "Error (" + level + ", " + seed + "): new
(toArray(m)) does not equal m");
  ensure(m.equals(wrap(m.TO_KEY_ARRAY())), "Error (" + level + ", " + seed + "): m does not equal wrap
(toArray(m))");
  ensure((wrap(m.TO_KEY_ARRAY())).equals(m), "Error (" + level + ", " + seed + "): wrap (toArray(m)) does not
equal m");


  int h = m.hashCode();

  /* Now we save and read m. */

  LIST m2 = null;

  try {
   java.io.File ff = new java.io.File("it.unimi.dsi.fastutil.test");
   java.io.OutputStream os = new java.io.FileOutputStream(ff);
   java.io.ObjectOutputStream oos = new java.io.ObjectOutputStream(os);

   oos.writeObject(m);
   oos.close();

   java.io.InputStream is = new java.io.FileInputStream(ff);
   java.io.ObjectInputStream ois = new java.io.ObjectInputStream(is);

   m2 = (LIST)ois.readObject();
   ois.close();
   ff.delete();
  }
  catch(Exception e) {
   e.printStackTrace();
   System.exit(1);
  }

#if ! KEY_CLASS_Reference
  ensure(m2.hashCode() == h, "Error (" + level + ", " + seed + "): hashCode() changed after save/read");

  /* Now we check that m2 actually holds that data. */

  ensure(m2.equals(t), "Error (" + level + ", " + seed + "): ! m2.equals(t) after save/read");
  ensure(t.equals(m2), "Error (" + level + ", " + seed + "): ! t.equals(m2) after save/read");
  /* Now we take out of m everything, and check that it is empty. */
```

```
  for(Iterator i=t.iterator(); i.hasNext();) m2.remove(i.next());

  ensure(m2.isEmpty(), "Error (" + level + ", " + seed + "): m2 is not empty (as it should be)");
#endif

  /* Now we play with iterators. */

  {
   KEY_LIST_ITERATOR i;
   java.util.ListIterator j;
   Object J;
   i = m.listIterator();
   j = t.listIterator();

   for(int k = 0; k < 2*n; k++) {
    ensure(i.hasNext() == j.hasNext(), "Error (" + level + ", " + seed + "): divergence in hasNext()");
    ensure(i.hasPrevious() == j.hasPrevious(), "Error (" + level + ", " + seed + "): divergence in hasPrevious()");

    if (r.nextFloat() < .8 && i.hasNext()) {
     ensure(i.next().equals(J = j.next()), "Error (" + level + ", " + seed + "): divergence in next()");

     if (r.nextFloat() < 0.2) {
      i.remove();
      j.remove();
     }
     else if (r.nextFloat() < 0.2) {
      KEY_TYPE T = genKey();
      i.set(T);
      j.set(KEY2OBJ(T));
     }
     else if (r.nextFloat() < 0.2) {
      KEY_TYPE T = genKey();
      i.add(T);
      j.add(KEY2OBJ(T));
     }
    }
    else if (r.nextFloat() < .2 && i.hasPrevious()) {
     ensure(i.previous().equals(J = j.previous()), "Error (" + level + ", " + seed + "): divergence in previous()");

     if (r.nextFloat() < 0.2) {
      i.remove();
      j.remove();
     }
     else if (r.nextFloat() < 0.2) {
      KEY_TYPE T = genKey();
      i.set(T);
      j.set(KEY2OBJ(T));
```

```
    }
    else if (r.nextFloat() < 0.2) {
     KEY_TYPE T = genKey();
     i.add(T);
     j.add(KEY2OBJ(T));
    }
   }

   ensure(i.nextIndex() == j.nextIndex(), "Error (" + level + ", " + seed + "): divergence in nextIndex()");
   ensure(i.previousIndex() == j.previousIndex(), "Error (" + level + ", " + seed + "): divergence in previousIndex()");

  }

 }

 {
  Object previous = null;
  Object I, J;
  int from = r.nextInt(m.size() +1);
  KEY_LIST_ITERATOR i;
  java.util.ListIterator j;
  i = m.listIterator(from);
  j = t.listIterator(from);

  for(int k = 0; k < 2*n; k++) {
   ensure(i.hasNext() == j.hasNext(), "Error (" + level + ", " + seed + "): divergence in hasNext() (iterator with starting
point " + from + ")");
   ensure(i.hasPrevious() == j.hasPrevious() , "Error (" + level + ", " + seed + "): divergence in hasPrevious() (iterator
with starting point " + from + ")");

   if (r.nextFloat() < .8 && i.hasNext()) {
    ensure((I = i.next()).equals(J = j.next()), "Error (" + level + ", " + seed + "): divergence in next() (" + I + ", " + J + ",
iterator with starting point " + from + ")");
    //System.err.println("Done next " + I + " " + J + "  " + badPrevious);

    if (r.nextFloat() < 0.2) {
     //System.err.println("Removing in next");
     i.remove();
     j.remove();
    }
    else if (r.nextFloat() < 0.2) {
     KEY_TYPE T = genKey();
     i.set(T);
     j.set(KEY2OBJ(T));
    }
    else if (r.nextFloat() < 0.2) {
     KEY_TYPE T = genKey();
     i.add(T);
```

```
     j.add(KEY2OBJ(T));
    }
   }
   else if (r.nextFloat() < .2 && i.hasPrevious()) {
    ensure((I = i.previous()).equals(J = j.previous()), "Error (" + level + ", " + seed + "): divergence in previous() (" + I +
", " + J + ", iterator with starting point " + from + ")");

    if (r.nextFloat() < 0.2) {
     //System.err.println("Removing in prev");
     i.remove();
     j.remove();
    }
    else if (r.nextFloat() < 0.2) {
     KEY_TYPE T = genKey();
     i.set(T);
     j.set(KEY2OBJ(T));
    }
    else if (r.nextFloat() < 0.2) {
     KEY_TYPE T = genKey();
     i.add(T);
     j.add(KEY2OBJ(T));
    }
   }
  }

  }

  /* Now we check that m actually holds that data. */

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after iteration");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after iteration");

  /* Now we select a pair of keys and create a subset. */

  if (! m.isEmpty()) {
   int start = r.nextInt(m.size());
   int end = start + r.nextInt(m.size() - start);
   //System.err.println("Checking subList from " + start + " to " + end + " (level=" + (level+1) + ")...");
   testLists(m.subList(start, end), t.subList(start, end), n, level + 1);

   ensure(m.equals(t), "Error (" + level + ", " + seed + m + t + "): ! m.equals(t) after subList");
   ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after subList");

  }

  m.clear();
  t.clear();
  ensure(m.isEmpty(), "Error (" + level + ", " + seed + "): m is not empty after clear()");
```

```
 }


 protected static void runTest(int n) {
  ARRAY_LIST m = new ARRAY_LIST();
  java.util.ArrayList t = new java.util.ArrayList();
  topList = m;
  k = new Object[n];
  nk = new Object[n];
  kt = new KEY_TYPE[n];
  nkt = new KEY_TYPE[n];

  for(int i = 0; i < n; i++) {
#if KEYS_REFERENCE
   k[i] = kt[i] = genKey();
   nk[i] = nkt[i] = genKey();
#else
   k[i] = new KEY_CLASS(kt[i] = genKey());
   nk[i] = new KEY_CLASS(nkt[i] = genKey());
#endif
  }

  /* We add pairs to t. */
  for(int i = 0; i < n;  i++) t.add(k[i]);

  /* We add to m the same data */
  m.addAll(t);

  testLists(m, t, n, 0);

  System.out.println("Test OK");
  return;
 }


 public static void main(String args[]) {
  int n  = Integer.parseInt(args[1]);
  if (args.length > 2) r = new java.util.Random(seed = Long.parseLong(args[2]));


  try {
   if ("speedTest".equals(args[0]) || "speedComp".equals(args[0])) speedTest(n, "speedComp".equals(args[0]));
   else if ("test".equals(args[0])) runTest(n);
  } catch(Throwable e) {
   e.printStackTrace(System.err);
   System.err.println("seed: " + seed);
  }
 }
```

```
#endif

}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/ArrayList.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2003-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

/** A class providing static methods and objects that do useful things with type-specific priority queues.
*
* @see it.unimi.dsi.fastutil.PriorityQueue
*/

public final class PRIORITY_QUEUES {

 private PRIORITY_QUEUES() {}

 /** A synchronized wrapper class for priority queues. */

 public static class SynchronizedPriorityQueue KEY_GENERIC implements PRIORITY_QUEUE KEY_GENERIC
{
  protected final PRIORITY_QUEUE KEY_GENERIC q;
  protected final Object sync;

  protected SynchronizedPriorityQueue(final PRIORITY_QUEUE KEY_GENERIC q, final Object sync) {
   this.q = q;
   this.sync = sync;
```

```
  }

  protected SynchronizedPriorityQueue(final PRIORITY_QUEUE KEY_GENERIC q) {
   this.q = q;
   this.sync = this;
  }

  @Override
  public void enqueue(KEY_GENERIC_TYPE x) { synchronized(sync) { q.enqueue(x); } }

  @Override
  public KEY_GENERIC_TYPE DEQUEUE() { synchronized(sync) { return q.DEQUEUE(); } }

  @Override
  public KEY_GENERIC_TYPE FIRST() { synchronized(sync) { return q.FIRST(); } }

  @Override
  public KEY_GENERIC_TYPE LAST() { synchronized(sync) { return q.LAST(); } }

  @Override
  public boolean isEmpty() { synchronized(sync) { return q.isEmpty(); } }

  @Override
  public int size() { synchronized(sync) { return q.size(); } }

  @Override
  public void clear() { synchronized(sync) { q.clear(); } }

  @Override
  public void changed() { synchronized(sync) { q.changed(); } }

  @Override
  public KEY_COMPARATOR KEY_SUPER_GENERIC comparator() { synchronized(sync) { return
q.comparator(); } }

#if KEYS_PRIMITIVE

  @Deprecated
  @Override
  public void enqueue(KEY_CLASS x) { synchronized(sync) { q.enqueue(x); } }

  @Deprecated
  @Override
  public KEY_CLASS dequeue() { synchronized(sync) { return q.dequeue(); } }

  @Deprecated
  @Override
  public KEY_CLASS first() { synchronized(sync) { return q.first(); } }
```

```
  @Deprecated
  @Override
  public KEY_CLASS last() { synchronized(sync) { return q.last(); } }

#endif
  @Override
  public int hashCode() { synchronized(sync) { return q.hashCode(); } }

  @Override
  public boolean equals(final Object o) { if (o == this) return true; synchronized(sync) { return q.equals(o); } }

  private void writeObject(java.io.ObjectOutputStream s) throws java.io.IOException {
   synchronized(sync) { s.defaultWriteObject(); }
  }
 }


 /** Returns a synchronized type-specific priority queue backed by the specified type-specific priority queue.
  *
  * @param q the priority queue to be wrapped in a synchronized priority queue.
  * @return a synchronized view of the specified priority queue.
  */
 public static KEY_GENERIC PRIORITY_QUEUE KEY_GENERIC synchronize(final PRIORITY_QUEUE KEY_GENERIC q) { return new SynchronizedPriorityQueue(q); }

 /** Returns a synchronized type-specific priority queue backed by the specified type-specific priority queue, using an assigned object to synchronize.
  *
  * @param q the priority queue to be wrapped in a synchronized priority queue.
  * @param sync an object that will be used to synchronize the access to the priority queue.
  * @return a synchronized view of the specified priority queue.
  */

 public static KEY_GENERIC PRIORITY_QUEUE KEY_GENERIC synchronize(final PRIORITY_QUEUE KEY_GENERIC q, final Object sync) { return new SynchronizedPriorityQueue(q, sync); }


}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/PriorityQueues.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
```

```
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

/**  An abstract class facilitating the creation of type-specific iterators.
*
* @deprecated As of fastutil 8 this class is no longer necessary, as its previous abstract
* methods are now default methods of the type-specific interface.
*/

@Deprecated
public abstract class KEY_ABSTRACT_ITERATOR KEY_GENERIC implements KEY_ITERATOR
KEY_GENERIC {
 protected KEY_ABSTRACT_ITERATOR() {}
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/AbstractIterator.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2003-2017 Paolo Boldi and Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
```

```
package PACKAGE;

import java.util.NoSuchElementException;

import it.unimi.dsi.fastutil.PriorityQueue;

/** A type-specific {@link PriorityQueue}; provides some additional methods that use polymorphism to avoid (un)boxing.
*
* <p>Additionally, this interface strengthens {@link #comparator()}.
*/

public interface PRIORITY_QUEUE extends PriorityQueue<KEY_CLASS> {

 /** Enqueues a new element.
  * @see PriorityQueue#enqueue(Object)
  * @param x the element to enqueue.
  */

 void enqueue(KEY_GENERIC_TYPE x);

 /** Dequeues the {@linkplain #first() first} element from the queue.
  * @see #dequeue()
  * @return the dequeued element.
  * @throws NoSuchElementException if the queue is empty.
  */

 KEY_GENERIC_TYPE DEQUEUE();

 /** Returns the first element of the queue.
  * @see #first()
  * @return the first element.
  * @throws NoSuchElementException if the queue is empty.
  */

 KEY_GENERIC_TYPE FIRST();

 /** Returns the last element of the queue, that is, the element the would be dequeued last (optional operation).
  * <p>This default implementation just throws an {@link UnsupportedOperationException}.
  * @see #last()
  * @return the last element.
  * @throws NoSuchElementException if the queue is empty.
  */

 default KEY_GENERIC_TYPE LAST() { throw new UnsupportedOperationException(); }
```

```
/** Returns the comparator associated with this priority queue, or null if it uses its elements' natural ordering.
 *
 * <p>Note that this specification strengthens the one given in {@link PriorityQueue#comparator()}.
 * @see PriorityQueue#comparator()
 * @return the comparator associated with this priority queue.
 */
@Override
KEY_COMPARATOR comparator();

/** {@inheritDoc}
 * <p>This default implementation delegates to the corresponding type-specific method.
 * @deprecated Please use the corresponding type-specific method instead. */
@Deprecated
@Override
default void enqueue(final KEY_GENERIC_CLASS x) { enqueue(x.KEY_VALUE()); }

/** {@inheritDoc}
 * <p>This default implementation delegates to the corresponding type-specific method.
 * @deprecated Please use the corresponding type-specific method instead. */
@Deprecated
@Override
default KEY_GENERIC_CLASS dequeue() { return KEY2OBJ(DEQUEUE()); }

/** {@inheritDoc}
 * <p>This default implementation delegates to the corresponding type-specific method.
 * @deprecated Please use the corresponding type-specific method instead. */
@Deprecated
@Override
default KEY_GENERIC_CLASS first() { return KEY2OBJ(FIRST()); }

/** {@inheritDoc}
 * <p>This default implementation delegates to the corresponding type-specific method.
 * @deprecated Please use the corresponding type-specific method instead. */
@Deprecated
@Override
default KEY_GENERIC_CLASS last() { return KEY2OBJ(LAST()); }
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/PriorityQueue.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
```

```
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

/** An abstract class providing basic methods for implementing a type-specific stack interface.
*
* @deprecated As of fastutil 8 this class is no longer necessary, as its previous abstract
* methods are now default methods of the type-specific interface.
*/
@Deprecated
public abstract class ABSTRACT_STACK KEY_GENERIC extends
it.unimi.dsi.fastutil.AbstractStack<KEY_GENERIC_CLASS> implements STACK KEY_GENERIC {
 protected ABSTRACT_STACK() {}
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/AbstractStack.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2003-2017 Paolo Boldi and Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-

a775574/src/it/unimi/dsi/fastutil/PriorityQueue.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/IndirectPriorityQueue.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/AbstractPriorityQueue.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/AbstractIndirectPriorityQueue.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2003-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*   http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/PriorityQueues.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/IndirectPriorityQueues.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*   http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

```
package PACKAGE;

import java.util.Collection;
import java.util.Iterator;
import java.util.RandomAccess;
import java.util.NoSuchElementException;
import it.unimi.dsi.fastutil.BigArrays;

#if KEYS_PRIMITIVE

/** A type-specific big list based on a big array; provides some additional methods that use polymorphism to avoid (un)boxing.
*
* <p>This class implements a lightweight, fast, open, optimized,
* reuse-oriented version of big-array-based big lists. Instances of this class
* represent a big list with a big array that is enlarged as needed when new entries
* are created (by doubling the current length), but is
* <em>never</em> made smaller (even on a {@link #clear()}). A family of
* {@linkplain #trim() trimming methods} lets you control the size of the
* backing big array; this is particularly useful if you reuse instances of this class.
* Range checks are equivalent to those of {@link java.util}'s classes, but
* they are delayed as much as possible. The backing big array is exposed by the
* {@link #elements()} method.
*
* <p>This class implements the bulk methods {@code removeElements()},
* {@code addElements()} and {@code getElements()} using
* high-performance system calls (e.g., {@link
* System#arraycopy(Object,int,Object,int,int) System.arraycopy()} instead of
* expensive loops.
*
* @see java.util.ArrayList
*/

public class BIG_ARRAY_BIG_LIST KEY_GENERIC extends ABSTRACT_BIG_LIST KEY_GENERIC implements RandomAccess, Cloneable, java.io.Serializable {
 private static final long serialVersionUID = -7046029254386353130L;


#else

/** A type-specific big-array-based big list; provides some additional methods that use polymorphism to avoid (un)boxing.
*
* <p>This class implements a lightweight, fast, open, optimized,
* reuse-oriented version of big-array-based big lists. Instances of this class
* represent a big list with a big array that is enlarged as needed when new entries
* are created (by doubling the current length), but is
```

```
* <em>never</em> made smaller (even on a {@link #clear()}). A family of
* {@linkplain #trim() trimming methods} lets you control the size of the
* backing big array; this is particularly useful if you reuse instances of this class.
* Range checks are equivalent to those of {@link java.util}'s classes, but
* they are delayed as much as possible.
*
* <p>The backing big array is exposed by the {@link #elements()} method. If an instance
* of this class was created {@linkplain #wrap(Object[][],long) by wrapping},
* backing-array reallocations will be performed using reflection, so that
* {@link #elements()} can return a big array of the same type of the original big array; the comments
* about efficiency made in {@link it.unimi.dsi.fastutil.objects.ObjectArrays} apply here.
*
* <p>This class implements the bulk methods {@code removeElements()},
* {@code addElements()} and {@code getElements()} using
* high-performance system calls (e.g., {@link
* System#arraycopy(Object,int,Object,int,int) System.arraycopy()} instead of
* expensive loops.
*
* @see java.util.ArrayList
*/

public class BIG_ARRAY_BIG_LIST KEY_GENERIC extends ABSTRACT_BIG_LIST KEY_GENERIC
implements RandomAccess, Cloneable, java.io.Serializable {
 private static final long serialVersionUID = -7046029254386353131L;


#endif

 /** The initial default capacity of a big-array big list. */
 public static final int DEFAULT_INITIAL_CAPACITY = 10;

#if ! KEYS_PRIMITIVE
 /** Whether the backing big array was passed to {@code wrap()}. In
  * this case, we must reallocate with the same type of big array. */
 protected final boolean wrapped;
#endif

 /** The backing big array. */
 protected transient KEY_GENERIC_TYPE a[][];

 /** The current actual size of the big list (never greater than the backing-array length). */
 protected long size;

 /** Creates a new big-array big list using a given array.
  *
  * <p>This constructor is only meant to be used by the wrapping methods.
  *
  * @param a the big array that will be used to back this big-array big list.
```

```
 */

 protected BIG_ARRAY_BIG_LIST(final KEY_GENERIC_TYPE a[][], @SuppressWarnings("unused") boolean
dummy) {
  this.a = a;
#if ! KEYS_PRIMITIVE
  this.wrapped = true;
#endif
 }

 /** Creates a new big-array big list with given capacity.
  *
  * @param capacity the initial capacity of the array list (may be 0).
  */

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public BIG_ARRAY_BIG_LIST(final long capacity) {
  if (capacity < 0) throw new IllegalArgumentException("Initial capacity (" + capacity + ") is negative");
  if (capacity == 0) a = KEY_GENERIC_BIG_ARRAY_CAST BIG_ARRAYS.EMPTY_BIG_ARRAY;
  else a = KEY_GENERIC_BIG_ARRAY_CAST BIG_ARRAYS.newBigArray(capacity);
#if ! KEYS_PRIMITIVE
  wrapped = false;
#endif
 }

 /** Creates a new big-array big list with {@link #DEFAULT_INITIAL_CAPACITY} capacity. */

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public BIG_ARRAY_BIG_LIST() {
  a = KEY_GENERIC_BIG_ARRAY_CAST BIG_ARRAYS.DEFAULT_EMPTY_BIG_ARRAY; // We delay
allocation
#if ! KEYS_PRIMITIVE
  wrapped = false;
#endif
 }

 /** Creates a new big-array big list and fills it with a given type-specific collection.
  *
  * @param c a type-specific collection that will be used to fill the array list.
  */

 public BIG_ARRAY_BIG_LIST(final COLLECTION KEY_EXTENDS_GENERIC c) {
  this(c.size());
  for(KEY_ITERATOR KEY_EXTENDS_GENERIC i = c.iterator(); i.hasNext();) add(i.NEXT_KEY());
 }

 /** Creates a new big-array big list and fills it with a given type-specific list.
  *
```

```
 * @param l a type-specific list that will be used to fill the array list.
 */

public BIG_ARRAY_BIG_LIST(final BIG_LIST KEY_EXTENDS_GENERIC l) {
 this(l.size64());
 l.getElements(0, a, 0, size = l.size64());
}

/** Creates a new big-array big list and fills it with the elements of a given big array.
 *
 * <p>Note that this constructor makes it easy to build big lists from literal arrays
 * declared as <code><var>type</var>[][] {{ <var>init_values</var> }}</code>.
 * The only constraint is that the number of initialisation values is
 * below {@link it.unimi.dsi.fastutil.BigArrays#SEGMENT_SIZE}.
 *
 * @param a a big array whose elements will be used to fill the array list.
 */

public BIG_ARRAY_BIG_LIST(final KEY_GENERIC_TYPE a[][]) {
 this(a, 0, BIG_ARRAYS.length(a));
}

/** Creates a new big-array big list and fills it with the elements of a given big array.
 *
 * <p>Note that this constructor makes it easy to build big lists from literal arrays
 * declared as <code><var>type</var>[][] {{ <var>init_values</var> }}</code>.
 * The only constraint is that the number of initialisation values is
 * below {@link it.unimi.dsi.fastutil.BigArrays#SEGMENT_SIZE}.
 *
 * @param a a big array whose elements will be used to fill the array list.
 * @param offset the first element to use.
 * @param length the number of elements to use.
 */

public BIG_ARRAY_BIG_LIST(final KEY_GENERIC_TYPE a[][], final long offset, final long length) {
 this(length);
 BIG_ARRAYS.copy(a, offset, this.a, 0, length);
 size = length;
}

/** Creates a new big-array big list and fills it with the elements returned by an iterator..
 *
 * @param i an iterator whose returned elements will fill the array list.
 */

public BIG_ARRAY_BIG_LIST(final Iterator<? extends KEY_GENERIC_CLASS> i) {
 this();
 while(i.hasNext()) this.add(KEY_CLASS2TYPE(i.next()));
```

```
 }

 /** Creates a new big-array big list and fills it with the elements returned by a type-specific iterator..
  *
  * @param i a type-specific iterator whose returned elements will fill the array list.
  */

 public BIG_ARRAY_BIG_LIST(final KEY_ITERATOR KEY_EXTENDS_GENERIC i) {
  this();
  while(i.hasNext()) this.add(i.NEXT_KEY());
 }

#if KEYS_PRIMITIVE
 /** Returns the backing big array of this big list.
  *
  * @return the backing big array.
  */

 public KEY_GENERIC_TYPE[][] elements() {
  return a;
 }
#else
 /** Returns the backing big array of this big list.
  *
  * <p>If this big-array big list was created by wrapping a given big array, it is guaranteed
  * that the type of the returned big array will be the same. Otherwise, the returned
  * big array will be an big array of objects.
  *
  * @return the backing big array.
  */

 public KEY_GENERIC_TYPE[][] elements() {
  return a;
 }
#endif

 /** Wraps a given big array into a big-array list of given size.
  *
  * @param a a big array to wrap.
  * @param length the length of the resulting big-array list.
  * @return a new big-array list of the given size, wrapping the given big array.
  */

 public static KEY_GENERIC BIG_ARRAY_BIG_LIST KEY_GENERIC wrap(final KEY_GENERIC_TYPE a[][],
final long length) {
  if (length > BIG_ARRAYS.length(a)) throw new IllegalArgumentException("The specified length (" + length + ")
is greater than the array size (" + BIG_ARRAYS.length(a) + ")");
  final BIG_ARRAY_BIG_LIST KEY_GENERIC l = new BIG_ARRAY_BIG_LIST
```

```
KEY_GENERIC_DIAMOND(a, false);
 l.size = length;
 return l;
 }

 /** Wraps a given big array into a big-array big list.
  *
  * @param a a big array to wrap.
  * @return a new big-array big list wrapping the given array.
  */

 public static KEY_GENERIC BIG_ARRAY_BIG_LIST KEY_GENERIC wrap(final KEY_GENERIC_TYPE
a[][]) {
 return wrap(a, BIG_ARRAYS.length(a));
 }



 /** Ensures that this big-array big list can contain the given number of entries without resizing.
  *
  * @param capacity the new minimum capacity for this big-array big list.
  */
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public void ensureCapacity(final long capacity) {
 if (capacity <= a.length || a == BIG_ARRAYS.DEFAULT_EMPTY_BIG_ARRAY) return;
#if KEYS_PRIMITIVE
 a = BIG_ARRAYS.forceCapacity(a, capacity, size);
#else
 if (wrapped) a = BIG_ARRAYS.forceCapacity(a, capacity, size);
 else {
  if (capacity > BIG_ARRAYS.length(a)) {
   final Object t[][] = BIG_ARRAYS.newBigArray(capacity);
   BIG_ARRAYS.copy(a, 0, t, 0, size);
   a = (KEY_GENERIC_TYPE[][])t;
  }
 }
#endif
 assert size <= BIG_ARRAYS.length(a);
 }

 /** Grows this big-array big list, ensuring that it can contain the given number of entries without resizing,
  * and in case increasing current capacity at least by a factor of 50%.
  *
  * @param capacity the new minimum capacity for this big-array big list.
  */
 SUPPRESS_WARNINGS_KEY_UNCHECKED
 private void grow(long capacity) {
 final long oldLength = BIG_ARRAYS.length(a);
 if (capacity <= oldLength) return;
```

```
 if (a != BIG_ARRAYS.DEFAULT_EMPTY_BIG_ARRAY) capacity = Math.max(oldLength + (oldLength >> 1),
capacity);
 else if (capacity < DEFAULT_INITIAL_CAPACITY) capacity = DEFAULT_INITIAL_CAPACITY;
#if KEYS_PRIMITIVE
 a = BIG_ARRAYS.forceCapacity(a, capacity, size);
#else
 if (wrapped) a =  BIG_ARRAYS.forceCapacity(a, capacity, size);
 else {
  final Object t[][] = BIG_ARRAYS.newBigArray(capacity);
  BIG_ARRAYS.copy(a, 0, t, 0, size);
  a = (KEY_GENERIC_TYPE[][])t;
 }
#endif
 assert size <= BIG_ARRAYS.length(a);
}

@Override
public void add(final long index, final KEY_GENERIC_TYPE k) {
 ensureIndex(index);
 grow(size + 1);
 if (index != size) BIG_ARRAYS.copy(a, index, a, index + 1, size - index);
 BIG_ARRAYS.set(a, index, k);
 size++;
 assert size <= BIG_ARRAYS.length(a);
}

@Override
public boolean add(final KEY_GENERIC_TYPE k) {
 grow(size + 1);
 BIG_ARRAYS.set(a, size++, k);
 assert size <= BIG_ARRAYS.length(a);
 return true;
}

@Override
public KEY_GENERIC_TYPE GET_KEY(final long index) {
 if (index >= size) throw new IndexOutOfBoundsException("Index (" + index + ") is greater than or equal to list size
(" + size + ")");
 return BIG_ARRAYS.get(a, index);
}

@Override
public long indexOf(final KEY_TYPE k) {
 for(long i = 0; i < size; i++) if (KEY_EQUALS(k, BIG_ARRAYS.get(a, i))) return i;
 return -1;
}

@Override
```

```
 public long lastIndexOf(final KEY_TYPE k) {
  for(long i = size; i-- != 0;) if (KEY_EQUALS(k, BIG_ARRAYS.get(a, i))) return i;
  return -1;
 }

 @Override
 public KEY_GENERIC_TYPE REMOVE_KEY(final long index) {
  if (index >= size) throw new IndexOutOfBoundsException("Index (" + index + ") is greater than or equal to list size
(" + size + ")");
  final KEY_GENERIC_TYPE old = BIG_ARRAYS.get(a, index);
  size--;
  if (index != size) BIG_ARRAYS.copy(a, index + 1, a, index, size - index);
#if KEYS_REFERENCE
  BIG_ARRAYS.set(a, size, null);
#endif
  assert size <= BIG_ARRAYS.length(a);
  return old;
 }

 @Override
 public boolean REMOVE(final KEY_TYPE k) {
  final long index = indexOf(k);
  if (index == -1) return false;
  REMOVE_KEY(index);
  assert size <= BIG_ARRAYS.length(a);
  return true;
 }

 @Override
 public KEY_GENERIC_TYPE set(final long index, final KEY_GENERIC_TYPE k) {
  if (index >= size) throw new IndexOutOfBoundsException("Index (" + index + ") is greater than or equal to list size
(" + size + ")");
  KEY_GENERIC_TYPE old = BIG_ARRAYS.get(a, index);
  BIG_ARRAYS.set(a, index, k);
  return old;
 }

#if KEYS_PRIMITIVE
 @Override
 public boolean removeAll(final COLLECTION c) {
  KEY_GENERIC_TYPE[] s = null, d = null;
  int ss = -1, sd = BigArrays.SEGMENT_SIZE, ds = -1, dd = BigArrays.SEGMENT_SIZE;
  for (long i = 0; i < size; i++) {
   if (sd == BigArrays.SEGMENT_SIZE) {
    sd = 0;
    s = a[++ss];
   }
   if (!c.contains(s[sd])) {
```

```
    if (dd == BigArrays.SEGMENT_SIZE) {
     d = a[++ds];
     dd = 0;
    }
    d[dd++] = s[sd];
   }
   sd++;
  }
  final long j = BigArrays.index(ds, dd);
#if KEYS_REFERENCE
  BIG_ARRAYS.fill(a, j, size, null);
#endif
  final boolean modified = size != j;
  size = j;
  return modified;
 }

#endif

 @Override
 public boolean removeAll(final Collection<?> c) {
  KEY_GENERIC_TYPE[] s = null, d = null;
  int ss = -1, sd = BigArrays.SEGMENT_SIZE, ds = -1, dd = BigArrays.SEGMENT_SIZE;
  for (long i = 0; i < size; i++) {
   if (sd == BigArrays.SEGMENT_SIZE) {
    sd = 0;
    s = a[++ss];
   }
   if (!c.contains(KEY2OBJ(s[sd]))) {
    if (dd == BigArrays.SEGMENT_SIZE) {
     d = a[++ds];
     dd = 0;
    }
    d[dd++] = s[sd];
   }
   sd++;
  }
  final long j = BigArrays.index(ds, dd);
#if KEYS_REFERENCE
  BIG_ARRAYS.fill(a, j, size, null);
#endif
  final boolean modified = size != j;
  size = j;
  return modified;
 }

 @Override
 public void clear() {
```

```
#if KEYS_REFERENCE
 BIG_ARRAYS.fill(a, 0, size, null);
#endif
 size = 0;
 assert size <= BIG_ARRAYS.length(a);
 }

 @Override
 public long size64() {
 return size;
 }

 @Override
 public void size(final long size) {
 if (size > BIG_ARRAYS.length(a)) a = BIG_ARRAYS.forceCapacity(a, size, this.size);
 if (size > this.size) BIG_ARRAYS.fill(a, this.size, size, KEY_NULL);
#if KEYS_REFERENCE
 else BIG_ARRAYS.fill(a, size, this.size, KEY_NULL);
#endif
 this.size = size;
 }

 @Override
 public boolean isEmpty() {
 return size == 0;
 }

 /** Trims this big-array big list so that the capacity is equal to the size.
 *
 * @see java.util.ArrayList#trimToSize()
 */
 public void trim() {
 trim(0);
 }

 /** Trims the backing big array if it is too large.
 *
 * If the current big array length is smaller than or equal to
 * {@code n}, this method does nothing. Otherwise, it trims the
 * big-array length to the maximum between {@code n} and {@link #size64()}.
 *
 * <p>This method is useful when reusing big lists.  {@linkplain #clear() Clearing a
 * big list} leaves the big-array length untouched. If you are reusing a big list
 * many times, you can call this method with a typical
 * size to avoid keeping around a very large big array just
 * because of a few large transient big lists.
 *
 * @param n the threshold for the trimming.
```

```
 */

public void trim(final long n) {
 final long arrayLength = BIG_ARRAYS.length(a);
 if (n >= arrayLength || size == arrayLength) return;
 a = BIG_ARRAYS.trim(a, Math.max(n, size));
 assert size <= BIG_ARRAYS.length(a);
}


  /** Copies element of this type-specific list into the given big array using optimized system calls.
 *
 * @param from the start index (inclusive).
 * @param a the destination big array.
 * @param offset the offset into the destination array where to store the first element copied.
 * @param length the number of elements to be copied.
 */

@Override
public void getElements(final long from, final KEY_TYPE[][] a, final long offset, final long length) {
 BIG_ARRAYS.copy(this.a, from, a, offset, length);
}

/** Removes elements of this type-specific list using optimized system calls.
 *
 * @param from the start index (inclusive).
 * @param to the end index (exclusive).
 */
@Override
public void removeElements(final long from, final long to) {
 BigArrays.ensureFromTo(size, from, to);
 BIG_ARRAYS.copy(a, to, a, from, size - to);
 size -= (to - from);
#if KEYS_REFERENCE
 BIG_ARRAYS.fill(a, size, size + to - from, null);
#endif
 }


/** Adds elements to this type-specific list using optimized system calls.
 *
 * @param index the index at which to add elements.
 * @param a the big array containing the elements.
 * @param offset the offset of the first element to add.
 * @param length the number of elements to add.
 */
@Override
public void addElements(final long index, final KEY_GENERIC_TYPE a[][], final long offset, final long length) {
```

```
 ensureIndex(index);
 BIG_ARRAYS.ensureOffsetLength(a, offset, length);
 grow(size + length);
 BIG_ARRAYS.copy(this.a, index, this.a, index + length, size - index);
 BIG_ARRAYS.copy(a, offset, this.a, index, length);
 size += length;
}

@Override
public KEY_BIG_LIST_ITERATOR KEY_GENERIC listIterator(final long index) {
 ensureIndex(index);

 return new KEY_BIG_LIST_ITERATOR KEY_GENERIC() {
  long pos = index, last = -1;

  @Override
  public boolean hasNext() { return pos < size; }
  @Override
  public boolean hasPrevious() { return pos > 0; }
  @Override
  public KEY_GENERIC_TYPE NEXT_KEY() { if (! hasNext()) throw new NoSuchElementException(); return
BIG_ARRAYS.get(a, last = pos++); }
  @Override
  public KEY_GENERIC_TYPE PREV_KEY() { if (! hasPrevious()) throw new NoSuchElementException(); return
BIG_ARRAYS.get(a, last = --pos); }
  @Override
  public long nextIndex() { return pos; }
  @Override
  public long previousIndex() { return pos - 1; }
  @Override
  public void add(KEY_GENERIC_TYPE k) {
   BIG_ARRAY_BIG_LIST.this.add(pos++, k);
   last = -1;
  }
  @Override
  public void set(KEY_GENERIC_TYPE k) {
   if (last == -1) throw new IllegalStateException();
   BIG_ARRAY_BIG_LIST.this.set(last, k);
  }
  @Override
  public void remove() {
   if (last == -1) throw new IllegalStateException();
   BIG_ARRAY_BIG_LIST.this.REMOVE_KEY(last);
   /* If the last operation was a next(), we are removing an element *before* us, and we must decrease pos
correspondingly. */
   if (last < pos) pos--;
   last = -1;
  }
```

```
  };
 }

 @Override
 public BIG_ARRAY_BIG_LIST KEY_GENERIC clone() {
  BIG_ARRAY_BIG_LIST KEY_GENERIC c = new BIG_ARRAY_BIG_LIST
KEY_GENERIC_DIAMOND(size);
  BIG_ARRAYS.copy(a, 0, c.a, 0, size);
  c.size = size;
  return c;
 }

#if KEY_CLASS_Object
 private boolean valEquals(final K a, final K b) {
  return a == null ? b == null : a.equals(b);
 }
#endif

 /** Compares this type-specific big-array list to another one.
  *
  * <p>This method exists only for sake of efficiency. The implementation
  * inherited from the abstract implementation would already work.
  *
  * @param l a type-specific big-array list.
  * @return true if the argument contains the same elements of this type-specific big-array list.
  */
 public boolean equals(final BIG_ARRAY_BIG_LIST KEY_GENERIC l) {
  if (l == this) return true;
  long s = size64();
  if (s != l.size64()) return false;
  final KEY_GENERIC_TYPE[][] a1 = a;
  final KEY_GENERIC_TYPE[][] a2 = l.a;

#if KEY_CLASS_Object
  while(s-- !=  0) if (! valEquals(BIG_ARRAYS.get(a1, s), BIG_ARRAYS.get(a2, s))) return false;
#else
  while(s-- !=  0) if (BIG_ARRAYS.get(a1, s) != BIG_ARRAYS.get(a2, s)) return false;
#endif
  return true;
 }


#if ! KEY_CLASS_Reference

 /** Compares this big list to another big list.
  *
  * <p>This method exists only for sake of efficiency. The implementation
  * inherited from the abstract implementation would already work.
```

```
 *
 * @param l a big list.
 * @return a negative integer,
 * zero, or a positive integer as this big list is lexicographically less than, equal
 * to, or greater than the argument.
 */
SUPPRESS_WARNINGS_KEY_UNCHECKED
public int compareTo(final BIG_ARRAY_BIG_LIST KEY_EXTENDS_GENERIC l) {
 final long s1 = size64(), s2 = l.size64();
 final KEY_GENERIC_TYPE a1[][] = a, a2[][] = l.a;
 KEY_GENERIC_TYPE e1, e2;
 int r, i;

 for(i = 0; i < s1 && i < s2; i++) {
  e1 = BIG_ARRAYS.get(a1, i);
  e2 = BIG_ARRAYS.get(a2, i);
  if ((r = KEY_CMP(e1, e2)) != 0) return r;
 }

 return i < s2 ? -1 : (i < s1 ? 1 : 0);
}
#endif


 private void writeObject(java.io.ObjectOutputStream s) throws java.io.IOException {
  s.defaultWriteObject();
  for(int i = 0; i < size; i++) s.WRITE_KEY(BIG_ARRAYS.get(a, i));
 }

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 private void readObject(java.io.ObjectInputStream s) throws java.io.IOException, ClassNotFoundException {
  s.defaultReadObject();
  a = KEY_GENERIC_BIG_ARRAY_CAST BIG_ARRAYS.newBigArray(size);
  for(int i = 0; i < size; i++) BIG_ARRAYS.set(a, i, KEY_GENERIC_CAST s.READ_KEY());
 }


#ifdef TEST

 private static long seed = System.currentTimeMillis();
 private static java.util.Random r = new java.util.Random(seed);

 private static KEY_TYPE genKey() {
#if KEY_CLASS_Byte || KEY_CLASS_Short || KEY_CLASS_Character
  return (KEY_TYPE)(r.nextInt());
#elif KEYS_PRIMITIVE
  return r.NEXT_KEY();
#elif KEY_CLASS_Object
```

```
  return Integer.toBinaryString(r.nextInt());
#else
  return new java.io.Serializable() {};
#endif
 }

 private static java.text.NumberFormat format = new java.text.DecimalFormat("#,###.00");
 private static java.text.FieldPosition p = new java.text.FieldPosition(0);

 private static String format(double d) {
  StringBuffer s = new StringBuffer();
  return format.format(d, s, p).toString();
 }

 private static void speedTest(int n, boolean comp) {
  System.out.println("There are presently no speed tests for this class.");
 }


 private static void fatal(String msg) {
  System.out.println(msg);
  System.exit(1);
 }

 private static void ensure(boolean cond, String msg) {
  if (cond) return;
  fatal(msg);
 }

 private static Object[] k, v, nk;
 private static KEY_TYPE kt[];
 private static KEY_TYPE nkt[];
 private static BIG_ARRAY_BIG_LIST topList;

 protected static void testLists(BIG_LIST m, BIG_LIST t, int n, int level) {
  long ms;
  Exception mThrowsIllegal, tThrowsIllegal, mThrowsOutOfBounds, tThrowsOutOfBounds;
  Object rt = null;
  KEY_TYPE rm = KEY_NULL;

  if (level > 4) return;


  /* Now we check that both sets agree on random keys. For m we use the polymorphic method. */

  for(int i = 0; i < n; i++) {
   int p = r.nextInt() % (n * 2);
```

```
  KEY_TYPE T = genKey();
  mThrowsOutOfBounds = tThrowsOutOfBounds  = null;

  try {
   m.set(p, T);
  }
  catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }
  try {
   t.set(p, KEY2OBJ(T));
  }
  catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }

  ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + level + ", " + seed + "): set() divergence at start in IndexOutOfBoundsException for index " + p + "  (" + mThrowsOutOfBounds + ", " + tThrowsOutOfBounds + ")");
  if (mThrowsOutOfBounds == null) ensure(t.get(p).equals(KEY2OBJ(m.GET_KEY(p))), "Error (" + level + ", " + seed + "): m and t differ after set() on position " + p + " (" + m.GET_KEY(p) + ", " + t.get(p) + ")");

  p = r.nextInt() % (n * 2);
  mThrowsOutOfBounds = tThrowsOutOfBounds  = null;

  try {
   m.GET_KEY(p);
  }
  catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }
  try {
   t.get(p);
  }
  catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }

  ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + level + ", " + seed + "): get() divergence at start in IndexOutOfBoundsException for index " + p + "  (" + mThrowsOutOfBounds + ", " + tThrowsOutOfBounds + ")");
  if (mThrowsOutOfBounds == null) ensure(t.get(p).equals(KEY2OBJ(m.GET_KEY(p))), "Error (" + level + ", " + seed + "): m and t differ aftre get() on position " + p + " (" + m.GET_KEY(p) + ", " + t.get(p) + ")");

 }

 /* Now we check that both sets agree on random keys. For m we use the standard method. */

 for(int i = 0; i < n; i++) {
  int p = r.nextInt() % (n * 2);

  mThrowsOutOfBounds = tThrowsOutOfBounds  = null;

  try {
   m.get(p);
  }
```

```
  catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }
  try {
   t.get(p);
  }
  catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }

  ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + level + ", " + seed + "): get() divergence at start in IndexOutOfBoundsException for index " + p + "  (" + mThrowsOutOfBounds + ", " + tThrowsOutOfBounds + ")");
  if (mThrowsOutOfBounds == null) ensure(t.get(p).equals(m.get(p)), "Error (" + level + ", " + seed + "): m and t differ at start on position " + p + " (" + m.get(p) + ", " + t.get(p) + ")");

 }

 /* Now we check that m and t are equal. */
 if (!m.equals(t) || ! t.equals(m)) System.err.println("m: " + m + " t: " + t);

 ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) at start");
 ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) at start");



 /* Now we check that m actually holds that data. */
 for(Iterator i=t.iterator(); i.hasNext();) {
  ensure(m.contains(i.next()), "Error (" + level + ", " + seed + "): m and t differ on an entry after insertion (iterating on t)");
 }

 /* Now we check that m actually holds that data, but iterating on m. */
 for(Iterator i=m.listIterator(); i.hasNext();) {
  ensure(t.contains(i.next()), "Error (" + level + ", " + seed + "): m and t differ on an entry after insertion (iterating on m)");
 }

 /* Now we check that inquiries about random data give the same answer in m and t. For
    m we use the polymorphic method. */

 for(int i=0; i<n;  i++) {
  KEY_TYPE T = genKey();
  ensure(m.contains(T) == t.contains(KEY2OBJ(T)), "Error (" + level + ", " + seed + "): divergence in content between t and m (polymorphic method)");
 }

 /* Again, we check that inquiries about random data give the same answer in m and t, but
    for m we use the standard method. */

 for(int i=0; i<n;  i++) {
  KEY_TYPE T = genKey();
```

```
  ensure(m.contains(KEY2OBJ(T)) == t.contains(KEY2OBJ(T)), "Error (" + level + ", " + seed + "): divergence in content between t and m (polymorphic method)");
 }

 /* Now we add and remove random data in m and t, checking that the result is the same. */

 for(int i=0; i<2*n;  i++) {
  KEY_TYPE T = genKey();

  try {
   m.add(T);
  }
  catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }

  try {
   t.add(KEY2OBJ(T));
  }
  catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }

  T = genKey();
  int p = r.nextInt() % (2 * n + 1);

  mThrowsOutOfBounds = tThrowsOutOfBounds  = null;

  try {
   m.add(p, T);
  }
  catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }

  try {
   t.add(p, KEY2OBJ(T));
  }
  catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }


  ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + level + ", " + seed + "): add() divergence in IndexOutOfBoundsException for index " + p + " for " + T + " (" + mThrowsOutOfBounds + ", " + tThrowsOutOfBounds + ")");

  p = r.nextInt() % (2 * n + 1);

  mThrowsOutOfBounds = tThrowsOutOfBounds  = null;

  try {
   rm = m.REMOVE_KEY(p);
  }
  catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }
```

```
  try {
   rt = t.remove(p);
  }
  catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }


  ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + level + ", " + seed + "): remove() divergence in IndexOutOfBoundsException for index " + p + " (" + mThrowsOutOfBounds + ", " + tThrowsOutOfBounds + ")");
  if (mThrowsOutOfBounds == null) ensure(rt.equals(KEY2OBJ(rm)), "Error (" + level + ", " + seed + "): divergence in remove() between t and m (" + rt + ", " + rm + ")");
  }

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after add/remove");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after add/remove");

  /* Now we add random data in m and t using addAll on a collection, checking that the result is the same. */

  for(int i=0; i<n;  i++) {
   int p = r.nextInt() % (2 * n + 1);
   java.util.Collection m1 = new java.util.ArrayList();
   int s = r.nextInt(n / 2 + 1);
   for(int j = 0; j < s; j++) m1.add(KEY2OBJ(genKey()));

   mThrowsOutOfBounds = tThrowsOutOfBounds  = null;

   try {
    m.addAll(p, m1);
   }
   catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }

   try {
    t.addAll(p, m1);
   }
   catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }

   ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + level + ", " + seed + "): addAll() divergence in IndexOutOfBoundsException for index " + p + " for " + m1 + " (" + mThrowsOutOfBounds + ", " + tThrowsOutOfBounds + ")");

   ensure(m.equals(t), "Error (" + level + ", " + seed + m + t + "): ! m.equals(t) after addAll");
   ensure(t.equals(m), "Error (" + level + ", " + seed + m + t + "): ! t.equals(m) after addAll");
  }

  if (m.size64() > n) {
   m.size(n);
   while(t.size64() != n) t.remove(t.size64() -1);
  }
```

```
  /* Now we add random data in m and t using addAll on a type-specific collection, checking that the result is the same. */

  for(int i=0; i<n;  i++) {
  int p = r.nextInt() % (2 * n + 1);
  COLLECTION m1 = new BIG_ARRAY_BIG_LIST();
  java.util.Collection t1 = new java.util.ArrayList();
  int s = r.nextInt(n / 2 + 1);
  for(int j = 0; j < s; j++) {
   KEY_TYPE x = genKey();
   m1.add(x);
   t1.add(KEY2OBJ(x));
  }

  mThrowsOutOfBounds = tThrowsOutOfBounds  = null;

  try {
   m.addAll(p, m1);
  }
  catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }

  try {
   t.addAll(p, t1);
  }
  catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }

  ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + level + ", " + seed + "): polymorphic addAll() divergence in IndexOutOfBoundsException for index " + p + " for " + m1 + " (" + mThrowsOutOfBounds + ", " + tThrowsOutOfBounds + ")");

  ensure(m.equals(t), "Error (" + level + ", " + seed + m + t + "): ! m.equals(t) after polymorphic addAll");
  ensure(t.equals(m), "Error (" + level + ", " + seed + m + t + "): ! t.equals(m) after polymorphic addAll");
  }

  if (m.size64() > n) {
  m.size(n);
  while(t.size64() != n) t.remove(t.size64() -1);
  }

  /* Now we add random data in m and t using addAll on a list, checking that the result is the same. */

  for(int i=0; i<n;  i++) {
  int p = r.nextInt() % (2 * n + 1);
  BIG_LIST m1 = new BIG_ARRAY_BIG_LIST();
  java.util.Collection t1 = new java.util.ArrayList();
  int s = r.nextInt(n / 2 + 1);
  for(int j = 0; j < s; j++) {
```

```
   KEY_TYPE x = genKey();
   m1.add(x);
   t1.add(KEY2OBJ(x));
  }

  mThrowsOutOfBounds = tThrowsOutOfBounds  = null;

  try {
   m.addAll(p, m1);
  }
  catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }

  try {
   t.addAll(p, t1);
  }
  catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }

  ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + level + ", " + seed + "): list
addAll() divergence in IndexOutOfBoundsException for index " + p + " for " + m1 + " (" + mThrowsOutOfBounds
+ ", " + tThrowsOutOfBounds + ")");

  ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after list addAll");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after list addAll");
  }

  /* Now we check that both sets agree on random keys. For m we use the standard method. */

  for(int i = 0; i < n; i++) {
   int p = r.nextInt() % (n * 2);

  mThrowsOutOfBounds = tThrowsOutOfBounds  = null;

  try {
   m.get(p);
  }
  catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }
  try {
   t.get(p);
  }
  catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }

  ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + level + ", " + seed + "):
get() divergence in IndexOutOfBoundsException for index " + p + "  (" + mThrowsOutOfBounds + ", " +
tThrowsOutOfBounds + ")");
  if (mThrowsOutOfBounds == null) ensure(t.get(p).equals(m.get(p)), "Error (" + level + ", " + seed + "): m and t
differ on position " + p + " (" + m.get(p) + ", " + t.get(p) +")");

  }
```

```
  /* Now we inquiry about the content with indexOf()/lastIndexOf(). */

  for(int i=0; i<10*n;  i++) {
   KEY_TYPE T = genKey();
   ensure(m.indexOf(KEY2OBJ(T)) == t.indexOf(KEY2OBJ(T)),
     "Error (" + level + ", " + seed + "): indexOf() divergence for " + T + "  (" + m.indexOf(KEY2OBJ(T)) + ", " +
t.indexOf(KEY2OBJ(T)) + ")");
   ensure(m.lastIndexOf(KEY2OBJ(T)) == t.lastIndexOf(KEY2OBJ(T)),
     "Error (" + level + ", " + seed + "): lastIndexOf() divergence for " + T + "  (" + m.lastIndexOf(KEY2OBJ(T)) + ", "
+ t.lastIndexOf(KEY2OBJ(T)) + ")");
   ensure(m.indexOf(T) == t.indexOf(KEY2OBJ(T)),
     "Error (" + level + ", " + seed + "): polymorphic indexOf() divergence for " + T + "  (" + m.indexOf(T) + ", " +
t.indexOf(KEY2OBJ(T)) + ")");
   ensure(m.lastIndexOf(T) == t.lastIndexOf(KEY2OBJ(T)),
     "Error (" + level + ", " + seed + "): polymorphic lastIndexOf() divergence for " + T + "  (" + m.lastIndexOf(T) + ", "
+ t.lastIndexOf(KEY2OBJ(T)) + ")");
  }

  /* Now we check cloning. */

  if (level == 0) {
   ensure(m.equals(((BIG_ARRAY_BIG_LIST)m).clone()), "Error (" + level + ", " + seed + "): m does not equal
m.clone()");
   ensure(((BIG_ARRAY_BIG_LIST)m).clone().equals(m), "Error (" + level + ", " + seed + "): m.clone() does not
equal m");
  }

  /* Now we play with constructors. */
  ensure(m.equals(new BIG_ARRAY_BIG_LIST((COLLECTION)m)), "Error (" + level + ", " + seed + "): m does
not equal new (type-specific Collection m)");
  ensure((new BIG_ARRAY_BIG_LIST((COLLECTION)m)).equals(m), "Error (" + level + ", " + seed + "): new
(type-specific nCollection m) does not equal m");
  ensure(m.equals(new BIG_ARRAY_BIG_LIST((BIG_LIST)m)), "Error (" + level + ", " + seed + "): m does not
equal new (type-specific List m)");
  ensure((new BIG_ARRAY_BIG_LIST((BIG_LIST)m)).equals(m), "Error (" + level + ", " + seed + "): new (type-
specific List m) does not equal m");
  ensure(m.equals(new BIG_ARRAY_BIG_LIST(m.listIterator())), "Error (" + level + ", " + seed + "): m does not
equal new (m.listIterator())");
  ensure((new BIG_ARRAY_BIG_LIST(m.listIterator())).equals(m), "Error (" + level + ", " + seed + "): new
(m.listIterator()) does not equal m");
  ensure(m.equals(new BIG_ARRAY_BIG_LIST(m.iterator())), "Error (" + level + ", " + seed + "): m does not equal
new (m.type_specific_iterator())");
  ensure((new BIG_ARRAY_BIG_LIST(m.iterator())).equals(m), "Error (" + level + ", " + seed + "): new
(m.type_specific_iterator()) does not equal m");


  int h = m.hashCode();
```

```
  /* Now we save and read m. */

  BIG_LIST m2 = null;

  try {
   java.io.File ff = new java.io.File("it.unimi.dsi.fastutil.test");
   java.io.OutputStream os = new java.io.FileOutputStream(ff);
   java.io.ObjectOutputStream oos = new java.io.ObjectOutputStream(os);

   oos.writeObject(m);
   oos.close();

   java.io.InputStream is = new java.io.FileInputStream(ff);
   java.io.ObjectInputStream ois = new java.io.ObjectInputStream(is);

   m2 = (BIG_LIST)ois.readObject();
   ois.close();
   ff.delete();
  }
  catch(Exception e) {
   e.printStackTrace();
   System.exit(1);
  }

#if ! KEY_CLASS_Reference
  ensure(m2.hashCode() == h, "Error (" + level + ", " + seed + "): hashCode() changed after save/read");

  /* Now we check that m2 actually holds that data. */

  ensure(m2.equals(t), "Error (" + level + ", " + seed + "): ! m2.equals(t) after save/read");
  ensure(t.equals(m2), "Error (" + level + ", " + seed + "): ! t.equals(m2) after save/read");
  /* Now we take out of m everything, and check that it is empty. */

  for(Iterator i=t.iterator(); i.hasNext();) m2.remove(i.next());

  ensure(m2.isEmpty(), "Error (" + level + ", " + seed + "): m2 is not empty (as it should be)");
#endif

  /* Now we play with iterators. */

  {
   KEY_BIG_LIST_ITERATOR i;
   KEY_BIG_LIST_ITERATOR j;
   Object J;
   i = m.listIterator();
   j = t.listIterator();
```

```
for(int k = 0; k < 2*n; k++) {
 ensure(i.hasNext() == j.hasNext(), "Error (" + level + ", " + seed + "): divergence in hasNext()");
 ensure(i.hasPrevious() == j.hasPrevious(), "Error (" + level + ", " + seed + "): divergence in hasPrevious()");

 if (r.nextFloat() < .8 && i.hasNext()) {
  ensure(i.next().equals(J = j.next()), "Error (" + level + ", " + seed + "): divergence in next()");

  if (r.nextFloat() < 0.2) {
   i.remove();
   j.remove();
  }
  else if (r.nextFloat() < 0.2) {
   KEY_TYPE T = genKey();
   i.set(T);
   j.set(KEY2OBJ(T));
  }
  else if (r.nextFloat() < 0.2) {
   KEY_TYPE T = genKey();
   i.add(T);
   j.add(KEY2OBJ(T));
  }
 }
 else if (r.nextFloat() < .2 && i.hasPrevious()) {
  ensure(i.previous().equals(J = j.previous()), "Error (" + level + ", " + seed + "): divergence in previous()");

  if (r.nextFloat() < 0.2) {
   i.remove();
   j.remove();
  }
  else if (r.nextFloat() < 0.2) {
   KEY_TYPE T = genKey();
   i.set(T);
   j.set(KEY2OBJ(T));
  }
  else if (r.nextFloat() < 0.2) {
   KEY_TYPE T = genKey();
   i.add(T);
   j.add(KEY2OBJ(T));
  }
 }

 ensure(i.nextIndex() == j.nextIndex(), "Error (" + level + ", " + seed + "): divergence in nextIndex()");
 ensure(i.previousIndex() == j.previousIndex(), "Error (" + level + ", " + seed + "): divergence in previousIndex()");

 }

}
```

```
 {
  Object previous = null;
  Object I, J;
  long from = m.isEmpty() ? 0 : (r.nextLong() & 0x7FFFFFFFFFFFFFFFL) % m.size64();
  KEY_BIG_LIST_ITERATOR i;
  KEY_BIG_LIST_ITERATOR j;
  i = m.listIterator(from);
  j = t.listIterator(from);

  for(int k = 0; k < 2*n; k++) {
   ensure(i.hasNext() == j.hasNext(), "Error (" + level + ", " + seed + "): divergence in hasNext() (iterator with starting point " + from + ")");
   ensure(i.hasPrevious() == j.hasPrevious() , "Error (" + level + ", " + seed + "): divergence in hasPrevious() (iterator with starting point " + from + ")");

   if (r.nextFloat() < .8 && i.hasNext()) {
    ensure((I = i.next()).equals(J = j.next()), "Error (" + level + ", " + seed + "): divergence in next() (" + I + ", " + J + ", iterator with starting point " + from + ")");
    //System.err.println("Done next " + I + " " + J + "  " + badPrevious);

    if (r.nextFloat() < 0.2) {
     //System.err.println("Removing in next");
     i.remove();
     j.remove();
    }
    else if (r.nextFloat() < 0.2) {
     KEY_TYPE T = genKey();
     i.set(T);
     j.set(KEY2OBJ(T));
    }
    else if (r.nextFloat() < 0.2) {
     KEY_TYPE T = genKey();
     i.add(T);
     j.add(KEY2OBJ(T));
    }
   }
   else if (r.nextFloat() < .2 && i.hasPrevious()) {
    ensure((I = i.previous()).equals(J = j.previous()), "Error (" + level + ", " + seed + "): divergence in previous() (" + I + ", " + J + ", iterator with starting point " + from + ")");

    if (r.nextFloat() < 0.2) {
     //System.err.println("Removing in prev");
     i.remove();
     j.remove();
    }
    else if (r.nextFloat() < 0.2) {
     KEY_TYPE T = genKey();
     i.set(T);
```

```
    j.set(KEY2OBJ(T));
   }
   else if (r.nextFloat() < 0.2) {
    KEY_TYPE T = genKey();
    i.add(T);
    j.add(KEY2OBJ(T));
   }
  }
 }

 }

 /* Now we check that m actually holds that data. */

 ensure(m.equals(t), "Error (" + level + ", " + seed + "): ! m.equals(t) after iteration");
 ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after iteration");

 /* Now we select a pair of keys and create a subset. */

 if (! m.isEmpty()) {
  long start = (r.nextLong() & 0x7FFFFFFFFFFFFFFFL) % m.size64();
  long end = start + (r.nextLong() & 0x7FFFFFFFFFFFFFFFL) % (m.size64() - start);
  //System.err.println("Checking subList from " + start + " to " + end + " (level=" + (level+1) + ")...");
  testLists(m.subList(start, end), t.subList(start, end), n, level + 1);

  ensure(m.equals(t), "Error (" + level + ", " + seed + m + t + "): ! m.equals(t) after subList");
  ensure(t.equals(m), "Error (" + level + ", " + seed + "): ! t.equals(m) after subList");

 }

 m.clear();
 t.clear();
 ensure(m.isEmpty(), "Error (" + level + ", " + seed + "): m is not empty after clear()");
}


protected static void runTest(int n) {
 BIG_ARRAY_BIG_LIST m = new BIG_ARRAY_BIG_LIST();
 BIG_LIST t = BIG_LISTS.asBigList(new ARRAY_LIST());
 topList = m;
 k = new Object[n];
 nk = new Object[n];
 kt = new KEY_TYPE[n];
 nkt = new KEY_TYPE[n];

 for(int i = 0; i < n; i++) {
#if KEYS_REFERENCE
  k[i] = kt[i] = genKey();
```

```
   nk[i] = nkt[i] = genKey();
#else
   k[i] = new KEY_CLASS(kt[i] = genKey());
   nk[i] = new KEY_CLASS(nkt[i] = genKey());
#endif
  }

  /* We add pairs to t. */
#if KEYS_PRIMITIVE
  for(int i = 0; i < n;  i++) t.add((KEY_GENERIC_CLASS)k[i]);
#else
  for(int i = 0; i < n;  i++) t.add(k[i]);
#endif

  /* We add to m the same data */
  m.addAll(t);

  testLists(m, t, n, 0);

  System.out.println("Test OK");
  return;
 }


 public static void main(String args[]) {
  int n  = Integer.parseInt(args[1]);
  if (args.length > 2) r = new java.util.Random(seed = Long.parseLong(args[2]));


  try {
   if ("speedTest".equals(args[0]) || "speedComp".equals(args[0])) speedTest(n, "speedComp".equals(args[0]));
   else if ("test".equals(args[0])) runTest(n);
  } catch(Throwable e) {
   e.printStackTrace(System.err);
   System.err.println("seed: " + seed);
  }
 }

#endif

}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/BigArrayBigList.drv
No license file was found, but licenses were detected in source scan.

```
/*
```

```
* Copyright (C) 2003-2017 Paolo Boldi and Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

#if KEY_CLASS_Object
import java.util.Comparator;
import it.unimi.dsi.fastutil.IndirectPriorityQueue;
#endif

import java.util.NoSuchElementException;

import it.unimi.dsi.fastutil.ints.IntArrays;

/** A type-specific heap-based semi-indirect priority queue.
*
* <p>Instances of this class use as reference list a <em>reference array</em>,
* which must be provided to each constructor. The priority queue is
* represented using a heap. The heap is enlarged as needed, but it is never
* shrunk. Use the {@link #trim()} method to reduce its size, if necessary.
*
* <p>This implementation allows one to enqueue several time the same index, but
* you must be careful when calling {@link #changed()}.
*/

public class HEAP_SEMI_INDIRECT_PRIORITY_QUEUE KEY_GENERIC implements
INDIRECT_PRIORITY_QUEUE KEY_GENERIC {

 /** The reference array. */
 protected final KEY_GENERIC_TYPE refArray[];

 /** The semi-indirect heap. */
 protected int heap[] = IntArrays.EMPTY_ARRAY;

 /** The number of elements in this queue. */
```

```
 protected int size;

 /** The type-specific comparator used in this queue. */
 protected KEY_COMPARATOR KEY_SUPER_GENERIC c;

 /** Creates a new empty queue without elements with a given capacity and comparator.
  *
  * @param refArray the reference array.
  * @param capacity the initial capacity of this queue.
  * @param c the comparator used in this queue, or {@code null} for the natural order.
  */
 public HEAP_SEMI_INDIRECT_PRIORITY_QUEUE(KEY_GENERIC_TYPE[] refArray, int capacity,
KEY_COMPARATOR KEY_SUPER_GENERIC c) {
  if (capacity > 0) this.heap = new int[capacity];
  this.refArray = refArray;
  this.c = c;
 }

 /** Creates a new empty queue with given capacity and using the natural order.
  *
  * @param refArray the reference array.
  * @param capacity the initial capacity of this queue.
  */
 public HEAP_SEMI_INDIRECT_PRIORITY_QUEUE(KEY_GENERIC_TYPE[] refArray, int capacity) {
  this(refArray, capacity, null);
 }

 /** Creates a new empty queue with capacity equal to the length of the reference array and a given comparator.
  *
  * @param refArray the reference array.
  * @param c the comparator used in this queue, or {@code null} for the natural order.
  */
 public HEAP_SEMI_INDIRECT_PRIORITY_QUEUE(KEY_GENERIC_TYPE[] refArray,
KEY_COMPARATOR KEY_SUPER_GENERIC c) {
  this(refArray, refArray.length, c);
 }

 /** Creates a new empty queue with capacity equal to the length of the reference array and using the natural order.
  * @param refArray the reference array.
  */
 public HEAP_SEMI_INDIRECT_PRIORITY_QUEUE(final KEY_GENERIC_TYPE[] refArray) {
  this(refArray, refArray.length, null);
 }



 /** Wraps a given array in a queue using a given comparator.
  *
  * <p>The queue returned by this method will be backed by the given array.
```

```
 * The first {@code size} element of the array will be rearranged so to form a heap (this is
 * more efficient than enqueing the elements of {@code a} one by one).
 *
 * @param refArray the reference array.
 * @param a an array of indices into {@code refArray}.
 * @param size the number of elements to be included in the queue.
 * @param c the comparator used in this queue, or {@code null} for the natural order.
 */
 public HEAP_SEMI_INDIRECT_PRIORITY_QUEUE(final KEY_GENERIC_TYPE[] refArray, final int[] a, int
size, final KEY_COMPARATOR KEY_SUPER_GENERIC c) {
 this(refArray, 0, c);
 this.heap = a;
 this.size = size;
 SEMI_INDIRECT_HEAPS.makeHeap(refArray, a, size, c);
 }


 /** Wraps a given array in a queue using a given comparator.
 *
 * <p>The queue returned by this method will be backed by the given array.
 * The elements of the array will be rearranged so to form a heap (this is
 * more efficient than enqueing the elements of {@code a} one by one).
 *
 * @param refArray the reference array.
 * @param a an array of indices into {@code refArray}.
 * @param c the comparator used in this queue, or {@code null} for the natural order.
 */
 public HEAP_SEMI_INDIRECT_PRIORITY_QUEUE(final KEY_GENERIC_TYPE[] refArray, final int[] a, final
KEY_COMPARATOR KEY_SUPER_GENERIC c) {
 this(refArray, a, a.length, c);
 }

 /** Wraps a given array in a queue using the natural order.
 *
 * <p>The queue returned by this method will be backed by the given array.
 * The first {@code size} element of the array will be rearranged so to form a heap (this is
 * more efficient than enqueing the elements of {@code a} one by one).
 *
 * @param refArray the reference array.
 * @param a an array of indices into {@code refArray}.
 * @param size the number of elements to be included in the queue.
 */
 public HEAP_SEMI_INDIRECT_PRIORITY_QUEUE(final KEY_GENERIC_TYPE[] refArray, final int[] a, int
size) {
 this(refArray, a, size, null);
 }
```

```
 /** Wraps a given array in a queue using the natural order.
  *
  * <p>The queue returned by this method will be backed by the given array.
  * The elements of the array will be rearranged so to form a heap (this is
  * more efficient than enqueing the elements of {@code a} one by one).
  *
  * @param refArray the reference array.
  * @param a an array of indices into {@code refArray}.
  */
 public HEAP_SEMI_INDIRECT_PRIORITY_QUEUE(final KEY_GENERIC_TYPE[] refArray, final int[] a) {
  this(refArray, a, a.length);
 }

 /** Ensures that the given index is a valid reference.
  *
  * @param index an index in the reference array.
  * @throws IndexOutOfBoundsException if the given index is negative or larger than the reference array length.
  */
 protected void ensureElement(final int index) {
  if (index < 0)  throw new IndexOutOfBoundsException("Index (" + index + ") is negative");
  if (index >= refArray.length) throw new IndexOutOfBoundsException("Index (" + index + ") is larger than or equal
to reference array size (" + refArray.length + ")");
 }

 @Override
 public void enqueue(int x) {
  ensureElement(x);

  if (size == heap.length) heap = IntArrays.grow(heap, size + 1);

  heap[size++] = x;
  SEMI_INDIRECT_HEAPS.upHeap(refArray, heap, size, size - 1, c);
 }

 @Override
 public int dequeue() {
  if (size == 0) throw new NoSuchElementException();
  final int result = heap[0];
  heap[0] = heap[--size];
  if (size != 0) SEMI_INDIRECT_HEAPS.downHeap(refArray, heap, size, 0, c);
  return result;
 }

 @Override
 public int first() {
  if (size == 0) throw new NoSuchElementException();
  return heap[0];
 }
```

```
/** {@inheritDoc}
 *
 * <p>The caller <strong>must</strong> guarantee that when this method is called the
 * index of the first element appears just once in the queue. Failure to do so
 * will bring the queue in an inconsistent state, and will cause
 * unpredictable behaviour.
 */
@Override
public void changed() {
 SEMI_INDIRECT_HEAPS.downHeap(refArray, heap, size, 0, c);
}

/** Rebuilds this heap in a bottom-up fashion (in linear time). */

@Override
public void allChanged() { SEMI_INDIRECT_HEAPS.makeHeap(refArray, heap, size, c); }

@Override
public int size() { return size; }

@Override
public void clear() { size = 0; }

/** Trims the backing array so that it has exactly {@link #size()} elements. */

public void trim() {
 heap = IntArrays.trim(heap, size);
}

@Override
public KEY_COMPARATOR KEY_SUPER_GENERIC comparator() { return c; }

/** Writes in the provided array the <em>front</em> of the queue, that is, the set of indices
 * whose elements have the same priority as the top.
 * @param a an array whose initial part will be filled with the frnot (must be sized as least as the heap size).
 * @return the number of elements of the front.
 */
@Override
public int front(final int[] a) { return c == null ? SEMI_INDIRECT_HEAPS.front(refArray, heap, size, a) :
SEMI_INDIRECT_HEAPS.front(refArray, heap, size, a, c); }

@Override
public String toString() {
 StringBuffer s = new StringBuffer();
 s.append("[");
 for (int i = 0; i < size; i++) {
  if (i != 0) s.append(", ");
```

```
  s.append(refArray[heap [i]]);
 }
 s.append("]");
 return s.toString();
}

#ifdef TEST

 /** The original class, now just used for testing. */

 private static class TestQueue {

  /** The reference array */
  private KEY_TYPE refArray[];
  /** Its length */
  private int N;
  /** The number of elements in the heaps */
  private int n;
  /** The two comparators */
  private KEY_COMPARATOR primaryComp, secondaryComp;
  /** Two indirect heaps are used, called {@code primary} and {@code secondary}. Each of them contains
   a permutation of {@code n} among the indices 0, 1, ..., {@code N}-1 in such a way that the corresponding
   objects be sorted with respect to the two comparators.
   We also need an array {@code inSec[]} so that {@code inSec[k]} is the index of {@code secondary}
   containing {@code k}.
  */
  private int primary[], secondary[], inSec[];

  /** Builds a double indirect priority queue.
   *  @param refArray The reference array.
   *  @param primaryComp The primary comparator.
   *  @param secondaryComp The secondary comparator.
   */
  public TestQueue(KEY_TYPE refArray[], KEY_COMPARATOR primaryComp, KEY_COMPARATOR
secondaryComp) {
   this.refArray = refArray;
   this.N = refArray.length;
   assert this.N != 0;
   this.n = 0;
   this.primaryComp = primaryComp;
   this.secondaryComp = secondaryComp;
   this.primary = new int[N];
   this.secondary = new int[N];
   this.inSec = new int[N];
   java.util.Arrays.fill(inSec, -1);
  }

  /** Adds an index to the queue. Notice that the index should not be already present in the queue.
```

```
   *  @param i The index to be added
   */
  public void add(int i) {
   if (i < 0 || i >= refArray.length) throw new IndexOutOfBoundsException();
   //if (inSec[i] >= 0) throw new IllegalArgumentException();
   primary[n] = i;
   n++;
   swimPrimary(n-1);
  }

  /** Heapify the primary heap.
   *  @param i The index of the heap to be heapified.
   */
  private void heapifyPrimary(int i) {
   int dep = primary[i];
   int child;

   while ((child = 2*i+1) < n) {
    if (child+1 < n && primaryComp.compare(refArray[primary[child+1]], refArray[primary[child]]) < 0) child++;
    if (primaryComp.compare(refArray[dep], refArray[primary[child]]) <= 0) break;
    primary[i] = primary[child];
    i = child;
   }
   primary[i] = dep;
  }

  /** Heapify the secondary heap.
   *  @param i The index of the heap to be heapified.
   */
  private void heapifySecondary(int i) {
   int dep = secondary[i];
   int child;

   while ((child = 2*i+1) < n) {
    if (child+1 < n && secondaryComp.compare(refArray[secondary[child+1]], refArray[secondary[child]]) < 0)
child++;
    if (secondaryComp.compare(refArray[dep], refArray[secondary[child]]) <= 0) break;
    secondary[i] = secondary[child]; inSec[secondary[i]] = i;
    i = child;
   }
   secondary[i] = dep; inSec[secondary[i]] = i;
  }

  /** Swim and heapify the primary heap.
   *  @param i The index to be moved.
   */
  private void swimPrimary(int i) {
   int dep = primary[i];
```

```
 int parent;

 while (i != 0 && (parent = (i - 1) / 2) >= 0) {
  if (primaryComp.compare(refArray[primary[parent]], refArray[dep]) <= 0) break;
  primary[i] = primary[parent];
  i = parent;
 }
 primary[i] = dep;
 heapifyPrimary(i);
}

/** Swim and heapify the secondary heap.
 *  @param i The index to be moved.
 */
private void swimSecondary(int i) {
 int dep = secondary[i];
 int parent;

 while (i != 0 && (parent = (i - 1) / 2) >= 0) {
  if (secondaryComp.compare(refArray[secondary[parent]], refArray[dep]) <= 0) break;
  secondary[i] = secondary[parent]; inSec[secondary[i]] = i;
  i = parent;
 }
 secondary[i] = dep; inSec[secondary[i]] = i;
 heapifySecondary(i);
}

/** Returns the minimum element with respect to the primary comparator.
 @return the minimum element.
*/
public int top() {
 if (n == 0) throw new NoSuchElementException();
 return primary[0];
}

/** Returns the minimum element with respect to the secondary comparator.
 @return the minimum element.
*/
public int secTop() {
 if (n == 0) throw new NoSuchElementException();
 return secondary[0];
}

/** Removes the minimum element with respect to the primary comparator.
 *  @return the removed element.
 */
public void remove() {
 if (n == 0) throw new NoSuchElementException();
```

```
  int result = primary[0];
  // Copy a leaf
  primary[0] = primary[n-1];
  n--;
  heapifyPrimary(0);
  return;
 }

 public void clear() {
  while(size() != 0) remove();
 }

 /** Signals that the minimum element with respect to the comparator has changed.
  */
 public void change() {
  heapifyPrimary(0);
 }

 /** Returns the number of elements in the queue.
  *  @return the size of the queue
  */
 public int size() {
  return n;
 }

 public String toString() {
  String s = "[";
  for (int i = 0; i < n; i++)
   s += refArray[primary[i]]+", ";
  return s+ "]";
 }
}

 private static long seed = System.currentTimeMillis();
 private static java.util.Random r = new java.util.Random(seed);

 private static KEY_TYPE genKey() {
#if KEY_CLASS_Byte || KEY_CLASS_Short || KEY_CLASS_Character
  return (KEY_TYPE)(r.nextInt());
#elif KEYS_PRIMITIVE
  return r.NEXT_KEY();
#elif KEY_CLASS_Object
  return Integer.toBinaryString(r.nextInt());
#else
  return new java.io.Serializable() {};
#endif
 }
```

```
 private static java.text.NumberFormat format = new java.text.DecimalFormat("#,###.00");
 private static java.text.FieldPosition p = new java.text.FieldPosition(0);

 private static String format(double d) {
  StringBuffer s = new StringBuffer();
  return format.format(d, s, p).toString();
 }

 private static void speedTest(int n, boolean comp) {
  System.out.println("There are presently no speed tests for this class.");
 }


 private static void fatal(String msg) {
  System.out.println(msg);
  System.exit(1);
 }

 private static void ensure(boolean cond, String msg) {
  if (cond) return;
  fatal(msg);
 }

 private static boolean heapEqual(int[] a, int[] b, int sizea, int sizeb) {
  if (sizea != sizeb) return false;
  while(sizea-- != 0) if (a[sizea] != b[sizea]) return false;
  return true;
 }

 protected static void runTest(int n) {
  long ms;
  Exception mThrowsIllegal, tThrowsIllegal, mThrowsOutOfBounds, tThrowsOutOfBounds, mThrowsNoElement,
tThrowsNoElement;
  int rm = 0, rt = 0;
  KEY_TYPE[] refArray = new KEY_TYPE[n];

  for(int i = 0; i < n; i++) refArray[i] = genKey();

  HEAP_SEMI_INDIRECT_PRIORITY_QUEUE m = new
HEAP_SEMI_INDIRECT_PRIORITY_QUEUE(refArray, COMPARATORS.NATURAL_COMPARATOR);
  TestQueue t = new TestQueue(refArray, COMPARATORS.NATURAL_COMPARATOR,
COMPARATORS.OPPOSITE_COMPARATOR);

  /* We add pairs to t. */
  for(int i = 0; i < n / 2;  i++) {
   t.add(i);
   m.enqueue(i);
  }
```

```
  ensure(heapEqual(m.heap, t.primary, m.size(), t.size()), "Error (" + seed + "): m and t differ after creation (" + m + ", " + t + ")");

  /* Now we add and remove random data in m and t, checking that the result is the same. */

  for(int i=0; i<2*n;  i++) {
   if (r.nextDouble() < 0.01) {
    t.clear();
    m.clear();
    for(int j = 0; j < n / 2;  j++) {
     t.add(j);
     m.enqueue(j);
    }
   }

   int T = r.nextInt(2 * n);

   mThrowsNoElement = tThrowsNoElement = mThrowsOutOfBounds = tThrowsOutOfBounds = mThrowsIllegal = tThrowsIllegal = null;

   try {
    m.enqueue(T);
   }
   catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }
   catch (IllegalArgumentException e) { mThrowsIllegal = e; }

   try {
    t.add(T);
   }
   catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }
   catch (IllegalArgumentException e) { tThrowsIllegal = e; }

   ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + seed + "): enqueue() divergence in IndexOutOfBoundsException for " + T + " (" + mThrowsOutOfBounds + ", " + tThrowsOutOfBounds + ")");
   ensure((mThrowsIllegal == null) == (tThrowsIllegal == null), "Error (" + seed + "): enqueue() divergence in IllegalArgumentException for " + T + " (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");

   ensure(heapEqual(m.heap, t.primary, m.size(), t.size()), "Error (" + seed + "): m and t differ after enqueue (" + m + ", " + t + ")");

   if (m.size() != 0) {
    ensure(m.first() == t.top(), "Error (" + seed + "): m and t differ in first element after enqueue (" + m.first() + ", " + t.top() + ")");
   }

   mThrowsNoElement = tThrowsNoElement = mThrowsOutOfBounds = tThrowsOutOfBounds = mThrowsIllegal =
```

```
tThrowsIllegal = null;

  try {
   rm = m.dequeue();
  }
  catch (IndexOutOfBoundsException e) { mThrowsOutOfBounds = e; }
  catch (IllegalArgumentException e) { mThrowsIllegal = e; }
  catch (NoSuchElementException e) { mThrowsNoElement = e; }

  try {
   rt = t.top();
   t.remove();
  }
  catch (IndexOutOfBoundsException e) { tThrowsOutOfBounds = e; }
  catch (IllegalArgumentException e) { tThrowsIllegal = e; }
  catch (NoSuchElementException e) { tThrowsNoElement = e; }

  ensure((mThrowsOutOfBounds == null) == (tThrowsOutOfBounds == null), "Error (" + seed + "): dequeue()
divergence in IndexOutOfBoundsException (" + mThrowsOutOfBounds + ", " + tThrowsOutOfBounds + ")");
  ensure((mThrowsIllegal == null) == (tThrowsIllegal == null), "Error (" + seed + "): dequeue() divergence in
IllegalArgumentException  (" + mThrowsIllegal + ", " + tThrowsIllegal + ")");
  ensure((mThrowsNoElement == null) == (tThrowsNoElement == null), "Error (" + seed + "): dequeue() divergence
in NoSuchElementException  (" + mThrowsNoElement + ", " + tThrowsNoElement + ")");
  if (mThrowsOutOfBounds == null) ensure(rt == rm , "Error (" + seed + "): divergence in dequeue() between t and m
(" + rt + ", " + rm + ")");



  ensure(heapEqual(m.heap, t.primary, m.size(), t.size()), "Error (" + seed + "): m and t differ after dequeue (" + m +
", " + t + ")");

  if (m.size() != 0) {
   ensure(m.first() == t.top(), "Error (" + seed + "): m and t differ in first element after dequeue (" + m.first() + ", " +
t.top() + ")");
  }

  if (m.size() != 0 && ((new it.unimi.dsi.fastutil.ints.IntOpenHashSet(m.heap, 0, m.size())).size() == m.size())) {

   refArray[m.first()] = genKey();

   m.changed();
   t.change();

   ensure(heapEqual(m.heap, t.primary, m.size(), t.size()), "Error (" + seed + "): m and t differ after change (" + m + ",
" + t + ")");

   if (m.size() != 0) {
    ensure(m.first() == t.top(), "Error (" + seed + "): m and t differ in first element after change (" + m.first() + ", " +
t.top() + ")");
```

```
   }
  }
 }


 /* Now we check that m actually holds the same data. */

 m.clear();
 ensure(m.isEmpty(), "Error (" + seed + "): m is not empty after clear()");

 System.out.println("Test OK");
}



public static void main(String args[]) {
 int n  = Integer.parseInt(args[1]);
 if (args.length > 2) r = new java.util.Random(seed = Long.parseLong(args[2]));


 try {
  if ("speedTest".equals(args[0]) || "speedComp".equals(args[0])) speedTest(n, "speedComp".equals(args[0]));
  else if ("test".equals(args[0])) runTest(n);
 } catch(Throwable e) {
  e.printStackTrace(System.err);
  System.err.println("seed: " + seed);
 }
}

#endif
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/HeapSemiIndirectPriorityQueue.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
```

```
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import VALUE_PACKAGE.VALUE_COLLECTION;

#if !KEY_CLASS_Object
import it.unimi.dsi.fastutil.objects.ObjectSet;
import it.unimi.dsi.fastutil.objects.ObjectIterator;
#endif

import java.util.function.Consumer;
import java.util.Map;

/** A type-specific {@link Map}; provides some additional methods that use polymorphism to avoid (un)boxing,
and handling of a default return value.
*
* <p>Besides extending the corresponding type-specific {@linkplain it.unimi.dsi.fastutil.Function function}, this
interface strengthens {@link Map#entrySet()},
* {@link #keySet()} and {@link #values()}. Moreover, a number of methods, such as {@link #size()}, {@link
#defaultReturnValue()}, etc., are un-defaulted
* as their function default do not make sense for a map.
* Maps returning entry sets of type {@link FastEntrySet} support also fast iteration.
*
* <p>A submap or subset may or may not have an
* independent default return value (which however must be initialized to the
* default return value of the originator).
*
* @see Map
*/

public interface MAP KEY_VALUE_GENERIC extends FUNCTION KEY_VALUE_GENERIC,
Map<KEY_GENERIC_CLASS,VALUE_GENERIC_CLASS> {

 /** An entry set providing fast iteration.
  *
  * <p>In some cases (e.g., hash-based classes) iteration over an entry set requires the creation
  * of a large number of {@link java.util.Map.Entry} objects. Some {@code fastutil}
  * maps might return {@linkplain Map#entrySet() entry set} objects of type {@code FastEntrySet}: in this case,
{@link #fastIterator() fastIterator()}
  * will return an iterator that is guaranteed not to create a large number of objects, <em>possibly
  * by returning always the same entry</em> (of course, mutated), and {@link #fastForEach(Consumer)} will apply
  * the provided consumer to all elements of the entry set, <em>which might be represented
  * always by the same entry</em> (of course, mutated).
```

```
 */

 interface FastEntrySet KEY_VALUE_GENERIC extends ObjectSet<MAP.Entry KEY_VALUE_GENERIC> {
  /** Returns a fast iterator over this entry set; the iterator might return always the same entry instance, suitably
mutated.
   *
   * @return a fast iterator over this entry set; the iterator might return always the same {@link java.util.Map.Entry}
instance, suitably mutated.
   */
  ObjectIterator<MAP.Entry KEY_VALUE_GENERIC> fastIterator();

  /** Iterates quickly over this entry set; the iteration might happen always on the same entry instance, suitably
mutated.
   *
   * <p>This default implementation just delegates to {@link #forEach(Consumer)}.
   *
   * @param consumer a consumer that will by applied to the entries of this set; the entries might be represented
   * by the same entry instance, suitably mutated.
   * @since 8.1.0
   */
  default void fastForEach(final Consumer<? super MAP.Entry KEY_VALUE_GENERIC> consumer) {
   forEach(consumer);
  }
 }

 /**
  * Returns the number of key/value mappings in this map.  If the
  * map contains more than {@link Integer#MAX_VALUE} elements, returns {@link Integer#MAX_VALUE}.
  *
  * @return the number of key-value mappings in this map.
  * @see it.unimi.dsi.fastutil.Size64
  */

 @Override
 int size();

 /** Removes all of the mappings from this map (optional operation).
  * The map will be empty after this call returns.
  *
  * @throws UnsupportedOperationException if the <tt>clear</tt> operation is not supported by this map
  */

 @Override
 default void clear() { throw new UnsupportedOperationException(); }

#if VALUES_PRIMITIVE
 /** Sets the default return value (optional operation).
  *
```

```
 * This value must be returned by type-specific versions of {@code get()},
 * {@code put()} and {@code remove()} to denote that the map does not contain
 * the specified key. It must be 0/{@code false} by default.
 *
 * @param rv the new default return value.
 * @see #defaultReturnValue()
 */
#else
 /** Sets the default return value (optional operation).
 *
 * This value must be returned by type-specific versions of {@code get()},
 * {@code put()} and {@code remove()} to denote that the map does not contain
 * the specified key. It must be {@code null} by default.
 *
 * <p><strong>Warning</strong>: Changing this to a non-null value can have
 * unforeseen consequences. Especially, it breaks compatibility with the
 * specifications of Java's {@link java.util.Map} interface. It has to be
 * used with great care and thorough study of all method comments is
 * recommended.
 *
 * @param rv the new default return value.
 * @see #defaultReturnValue()
 */
#endif

 @Override
 void defaultReturnValue(VALUE_GENERIC_TYPE rv);

 /** Gets the default return value.
 *
 * @return the current default return value.
 */

 @Override
 VALUE_GENERIC_TYPE defaultReturnValue();

 /** Returns a type-specific set view of the mappings contained in this map.
 *
 * <p>This method is necessary because there is no inheritance along
 * type parameters: it is thus impossible to strengthen {@link Map#entrySet()}
 * so that it returns an {@link it.unimi.dsi.fastutil.objects.ObjectSet}
 * of type-specific entries (the latter makes it possible to
 * access keys and values with type-specific methods).
 *
 * @return a type-specific set view of the mappings contained in this map.
 * @see Map#entrySet()
 */
 ObjectSet<MAP.Entry KEY_VALUE_GENERIC> ENTRYSET();
```

```
#if KEYS_PRIMITIVE || VALUES_PRIMITIVE
 /** Returns a set view of the mappings contained in this map.
  *  <p>Note that this specification strengthens the one given in {@link Map#entrySet()}.
  *
  * @return a set view of the mappings contained in this map.
  * @see Map#entrySet()
  * @deprecated Please use the corresponding type-specific method instead.
  */
 @Deprecated
 @Override
 @SuppressWarnings({ "unchecked", "rawtypes" })
 default ObjectSet<Map.Entry<KEY_GENERIC_CLASS, VALUE_GENERIC_CLASS>> entrySet() {
  return (ObjectSet)ENTRYSET();
 }
#else
 /** Returns a set view of the mappings contained in this map.
  *  <p>Note that this specification strengthens the one given in {@link Map#entrySet()}.
  *
  * @return a set view of the mappings contained in this map.
  * @see Map#entrySet()
  */

 @Override
 @SuppressWarnings({ "unchecked", "rawtypes" })
 default ObjectSet<Map.Entry<KEY_GENERIC_CLASS, VALUE_GENERIC_CLASS>> entrySet() {
  return (ObjectSet)ENTRYSET();
 }
#endif

#if KEYS_PRIMITIVE || VALUES_PRIMITIVE
 /** {@inheritDoc}
  * <p>This default implementation just delegates to the corresponding type-specific&ndash;{@linkplain
it.unimi.dsi.fastutil.Function function} method.
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
#else
 /** {@inheritDoc}
  * <p>This default implementation just delegates to the corresponding function method.
  */
#endif
 @Override
 default VALUE_GENERIC_CLASS put(final KEY_GENERIC_CLASS key, final VALUE_GENERIC_CLASS
value) {
  return FUNCTION.super.put(key, value);
 }

#if KEYS_PRIMITIVE || VALUES_PRIMITIVE
```

```
 /** {@inheritDoc}
  * <p>This default implementation just delegates to the corresponding type-specific&ndash;{@linkplain
it.unimi.dsi.fastutil.Function function} method.
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 default VALUE_GENERIC_CLASS get(final Object key) {
  return FUNCTION.super.get(key);
 }

#endif

#if KEYS_PRIMITIVE || VALUES_PRIMITIVE
 /** {@inheritDoc}
  * <p>This default implementation just delegates to the corresponding type-specific&ndash;{@linkplain
it.unimi.dsi.fastutil.Function function} method.
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
#else
 /** {@inheritDoc}
  * <p>This default implementation just delegates to the corresponding type-specific&ndash;{@linkplain
it.unimi.dsi.fastutil.Function function} method.
  */
#endif
 @Override
 default VALUE_GENERIC_CLASS remove(final Object key) {
  return FUNCTION.super.remove(key);
 }

 /** {@inheritDoc}
  * <p>Note that this specification strengthens the one given in {@link Map#keySet()}.
  * @return a set view of the keys contained in this map.
  * @see Map#keySet()
  */

 @Override
 SET KEY_GENERIC keySet();

 /** {@inheritDoc}
  *  <p>Note that this specification strengthens the one given in {@link Map#values()}.
  * @return a set view of the values contained in this map.
  * @see Map#values()
  */

 @Override
 VALUE_COLLECTION VALUE_GENERIC values();
```

```
#if KEYS_PRIMITIVE

 /** Returns true if this function contains a mapping for the specified key.
  *
  * @param key the key.
  * @return true if this function associates a value to {@code key}.
  * @see Map#containsKey(Object)
  */

 @Override
 boolean containsKey(KEY_TYPE key);

 /** Returns true if this function contains a mapping for the specified key.
  * <p>This default implementation just delegates to the corresponding type-specific&ndash;{@linkplain
it.unimi.dsi.fastutil.Function function} method.
  * @deprecated Please use the corresponding type-specific method instead.
  */

 @Deprecated
 @Override
 default boolean containsKey(final Object key) {
  return FUNCTION.super.containsKey(key);
 }
#else

 /** Returns true if this function contains a mapping for the specified key.
  *
  * @param key the key.
  * @return true if this function associates a value to {@code key}.
  * @see Map#containsKey(Object)
  */

 @Override
 boolean containsKey(Object key);

#endif

#if VALUES_PRIMITIVE

 /** Returns {@code true} if this map maps one or more keys to the specified value.
  * @see Map#containsValue(Object)
  */

 boolean containsValue(VALUE_TYPE value);

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
```

```
 @Deprecated
 @Override
 default boolean containsValue(final Object value) {
  return value == null ? false : containsValue(VALUE_OBJ2TYPE(value));
 }

#endif

 // Defaultable methods

#if KEYS_PRIMITIVE || VALUES_PRIMITIVE

 /** Returns the value to which the specified key is mapped, or the {@code defaultValue} if this
  * map contains no mapping for the key.
  *
  * @param key the key.
  * @param defaultValue the default mapping of the key.
  *
  * @return the value to which the specified key is mapped, or the {@code defaultValue} if this map contains no
mapping for the key.
  *
  * @see java.util.Map#getOrDefault(Object, Object)
  * @since 8.0.0
  */

 default VALUE_GENERIC_TYPE getOrDefault(final KEY_TYPE key, final VALUE_GENERIC_TYPE
defaultValue) {
  final VALUE_GENERIC_TYPE v;
  return ((v = GET_VALUE(key)) != defaultReturnValue() || containsKey(key)) ? v : defaultValue;
 }

 /** If the specified key is not already associated with a value, associates it with the given
  * value and returns the {@linkplain #defaultReturnValue() default return value}, else returns
  * the current value.
  *
  * @param key key with which the specified value is to be associated.
  * @param value value to be associated with the specified key.
  *
  * @return the previous value associated with the specified key, or the {@linkplain #defaultReturnValue() default
return value} if there was no mapping for the key.
  *
  * @see java.util.Map#putIfAbsent(Object, Object)
  * @since 8.0.0
  */

 default VALUE_GENERIC_TYPE putIfAbsent(final KEY_GENERIC_TYPE key, final
VALUE_GENERIC_TYPE value) {
  final VALUE_GENERIC_TYPE v = GET_VALUE(key), drv = defaultReturnValue();
```

```
 if (v != drv || containsKey(key)) return v;
 put(key, value);
 return drv;
}

/** Removes the entry for the specified key only if it is currently mapped to the specified value.
 *
 * @param key key with which the specified value is associated.
 * @param value value expected to be associated with the specified key.
 *
 * @return {@code true} if the value was removed.
 *
 * @see java.util.Map#remove(Object, Object)
 * @since 8.0.0
 */

default boolean remove(final KEY_TYPE key, final VALUE_TYPE value) {
 final VALUE_GENERIC_TYPE curValue = GET_VALUE(key);
 if (!VALUE_EQUALS(curValue, value) || (curValue == defaultReturnValue() && !containsKey(key))) return false;
 REMOVE_VALUE(key);
 return true;
}

/** Replaces the entry for the specified key only if currently mapped to the specified value.
 *
 * @param key key with which the specified value is associated.
 * @param oldValue value expected to be associated with the specified key.
 * @param newValue value to be associated with the specified key.
 *
 * @return {@code true} if the value was replaced.
   *
 * @see java.util.Map#replace(Object, Object, Object)
 * @since 8.0.0
 */

default boolean replace(final KEY_GENERIC_TYPE key, final VALUE_GENERIC_TYPE oldValue, final
VALUE_GENERIC_TYPE newValue) {
 final VALUE_GENERIC_TYPE curValue = GET_VALUE(key);
 if (!VALUE_EQUALS(curValue, oldValue) || (curValue == defaultReturnValue() && !containsKey(key))) return
false;
 put(key, newValue);
 return true;
}

/** Replaces the entry for the specified key only if it is currently mapped to some value.
 *
 * @param key key with which the specified value is associated.
 * @param value value to be associated with the specified key.
```

```
  *
  * @return the previous value associated with the specified key, or the {@linkplain #defaultReturnValue() default return value} if there was no mapping for the key.
  *
  * @see java.util.Map#replace(Object, Object)
  * @since 8.0.0
  */

 default VALUE_GENERIC_TYPE replace(final KEY_GENERIC_TYPE key, final VALUE_GENERIC_TYPE value) { return containsKey(key) ? put(key, value) : defaultReturnValue(); }

#ifdef JDK_PRIMITIVE_FUNCTION

 /** If the specified key is not already associated with a value, attempts to compute its value
  * using the given mapping function and enters it into this map.
  *
  * <p>Note that contrarily to the default {@linkplain java.util.Map#computeIfAbsent(Object, java.util.function.Function) computeIfAbsent()},
  * it is not possible to not add a value for a given key, since the {@code mappingFunction} cannot
  * return {@code null}. If such a behavior is needed, please use the corresponding <em>nullable</em> version.
  *
  * @param key key with which the specified value is to be associated.
  * @param mappingFunction the function to compute a value.
  *
  * @return the current (existing or computed) value associated with the specified key.
  *
  * @see java.util.Map#computeIfAbsent(Object, java.util.function.Function)
  * @since 8.0.0
  */

 default VALUE_GENERIC_TYPE COMPUTE_IF_ABSENT_JDK(final KEY_GENERIC_TYPE key, final JDK_PRIMITIVE_FUNCTION KEY_SUPER_GENERIC VALUE_EXTENDS_GENERIC mappingFunction) {
  java.util.Objects.requireNonNull(mappingFunction);
  final VALUE_GENERIC_TYPE v = GET_VALUE(key);
  if (v != defaultReturnValue() || containsKey(key)) return v;

  VALUE_GENERIC_TYPE newValue = VALUE_NARROWING(mappingFunction.JDK_PRIMITIVE_FUNCTION_APPLY(key));
  put(key, newValue);
  return newValue;
 }

#endif

#if KEYS_PRIMITIVE && VALUES_PRIMITIVE

 /** If the specified key is not already associated with a value, attempts to compute its value
  * using the given mapping function and enters it into this map unless it is {@code null}.
```

```
  *
  * <p>Note that this version of {@linkplain java.util.Map#computeIfAbsent(Object, java.util.function.Function)
computeIfAbsent()}
  * should be used only if you plan to return {@code null} in the mapping function.
  *
  * @param key key with which the specified value is to be associated.
  * @param mappingFunction the function to compute a value.
  *
  * @return the current (existing or computed) value associated with the specified key,
  * or the {@linkplain #defaultReturnValue() default return value} if the computed value is {@code null}.
  *
  * @see java.util.Map#computeIfAbsent(Object, java.util.function.Function)
  * @since 8.0.0
  */

 default VALUE_GENERIC_TYPE COMPUTE_IF_ABSENT_NULLABLE(final KEY_GENERIC_TYPE key,
final JDK_KEY_TO_GENERIC_FUNCTION<? extends VALUE_GENERIC_CLASS> mappingFunction) {
  java.util.Objects.requireNonNull(mappingFunction);
  final VALUE_GENERIC_TYPE v = GET_VALUE(key), drv = defaultReturnValue();
  if (v != drv || containsKey(key)) return v;

  VALUE_GENERIC_CLASS mappedValue = mappingFunction.apply(key);
  if (mappedValue == null) return drv;
  VALUE_GENERIC_TYPE newValue = VALUE_CLASS2TYPE(mappedValue);
  put(key, newValue);
  return newValue;
 }

#endif

 /** If the specified key is not already associated with a value, attempts to compute its value
  * using the given mapping function and enters it into this map, unless the key is not present
  * in the given mapping function.
  *
  * <p>This version of {@linkplain java.util.Map#computeIfAbsent(Object, java.util.function.Function)
computeIfAbsent()}
  * uses a type-specific version of <code>fastutil</code>'s {@link it.unimi.dsi.fastutil.Function Function}.
  * Since {@link it.unimi.dsi.fastutil.Function Function} has a {@link
it.unimi.dsi.fastutil.Function#containsKey(Object) containsKey()}
  * method, it is possible to avoid adding a key by having {@code containsKey()} return {@code false} for that key.
  *
  * @param key key with which the specified value is to be associated.
  * @param mappingFunction the function to compute a value.
  *
  * @return the current (existing or computed) value associated with the specified key.
  *
  * @see java.util.Map#computeIfAbsent(Object, java.util.function.Function)
  * @since 8.0.0
```

```
 */

 default VALUE_GENERIC_TYPE COMPUTE_IF_ABSENT_PARTIAL(final KEY_GENERIC_TYPE key, final
FUNCTION KEY_SUPER_GENERIC VALUE_EXTENDS_GENERIC mappingFunction) {
  java.util.Objects.requireNonNull(mappingFunction);
  final VALUE_GENERIC_TYPE v = GET_VALUE(key), drv = defaultReturnValue();
  if (v != drv || containsKey(key)) return v;

  if (!mappingFunction.containsKey(key)) return drv;
  VALUE_GENERIC_TYPE newValue = mappingFunction.GET_VALUE(key);
  put(key, newValue);
  return newValue;
 }

 /** If the value for the specified key is present, attempts to compute a new mapping given the key and its current
mapped value.
  *
  * @param key key with which the specified value is to be associated.
  * @param remappingFunction the function to compute a value.
  *
  * @return the new value associated with the specified key, or the {@linkplain #defaultReturnValue() default return
value} if none.
  *
  * @see java.util.Map#computeIfPresent(Object, java.util.function.BiFunction)
  * @since 8.0.0
  */

 default VALUE_GENERIC_TYPE COMPUTE_IF_PRESENT(final KEY_GENERIC_TYPE key, final
java.util.function.BiFunction<? super KEY_GENERIC_CLASS, ? super VALUE_GENERIC_CLASS, ? extends
VALUE_GENERIC_CLASS> remappingFunction) {
  java.util.Objects.requireNonNull(remappingFunction);

  final VALUE_GENERIC_TYPE oldValue = GET_VALUE(key), drv = defaultReturnValue();
  if (oldValue == drv && !containsKey(key)) return drv;
  final VALUE_GENERIC_CLASS newValue = remappingFunction.apply(KEY2OBJ(key),
VALUE2OBJ(oldValue));

  if (newValue == null) { REMOVE_VALUE(key); return drv; }

#if VALUES_PRIMITIVE
  VALUE_GENERIC_TYPE newVal = VALUE_CLASS2TYPE(newValue);
  put(key, newVal);
  return newVal;
#else
  put(key, newValue);
  return newValue;
#endif
 }
```

```
 /** Attempts to compute a mapping for the specified key and its current mapped value (or {@code null} if there is no current mapping).
  *
  * <p>If the function returns {@code null}, the mapping is removed (or remains absent if initially absent).
  * If the function itself throws an (unchecked) exception, the exception is rethrown, and the current mapping is left unchanged.
  *
  * @param key key with which the specified value is to be associated.
  * @param remappingFunction the function to compute a value.
  *
  * @return the new value associated with the specified key, or the {@linkplain #defaultReturnValue() default return value} if none.
  *
  * @see java.util.Map#compute(Object, java.util.function.BiFunction)
  * @since 8.0.0
  */

 default VALUE_GENERIC_TYPE COMPUTE(final KEY_GENERIC_TYPE key, final java.util.function.BiFunction<? super KEY_GENERIC_CLASS, ? super VALUE_GENERIC_CLASS, ? extends VALUE_GENERIC_CLASS> remappingFunction) {
  java.util.Objects.requireNonNull(remappingFunction);
  final VALUE_GENERIC_TYPE oldValue = GET_VALUE(key), drv = defaultReturnValue();
  final boolean contained = oldValue != drv || containsKey(key);
  final VALUE_GENERIC_CLASS newValue = remappingFunction.apply(KEY2OBJ(key), contained ? VALUE2OBJ(oldValue) : null);

  if (newValue == null) {
   if (contained) REMOVE_VALUE(key);
   return drv;
  }
#if VALUES_PRIMITIVE
  final VALUE_GENERIC_TYPE newVal = VALUE_CLASS2TYPE(newValue);
  put(key, newVal);
  return newVal;
#else
  put(key, newValue);
  return newValue;
#endif
 }

#if VALUES_PRIMITIVE
 /**
  * If the specified key is not already associated with a value, associates it with the given {@code value}.
  * Otherwise, replaces the associated value with the results of the given remapping function, or removes if the result is {@code null}.
  *
  * @param key key with which the resulting value is to be associated.
```

```
  * @param value the value to be merged with the existing value associated with the key or, if no existing value is associated with the key, to be associated with the key.
  * @param remappingFunction the function to recompute a value if present.
  *
  * @return the new value associated with the specified key, or the {@linkplain #defaultReturnValue() default return value} if no value is associated with the key.
  *
  * @see java.util.Map#merge(Object, Object, java.util.function.BiFunction)
  * @since 8.0.0
  */
#else
 /**
  * If the specified key is not already associated with a value or is associated with null, associates it with the given non-null {@code value}.
  * Otherwise, replaces the associated value with the results of the given remapping function, or removes if the result is {@code null}.
  *
  * @param key key with which the resulting value is to be associated.
  * @param value the non-{@code null} value to be merged with the existing value associated with the key or, if no existing value is associated with the key, to be associated with the key.
  * @param remappingFunction the function to recompute a value if present.
  *
  * @return the new value associated with the specified key, or the {@linkplain #defaultReturnValue() default return value} if no value is associated with the key.
  *
  * @see java.util.Map#merge(Object, Object, java.util.function.BiFunction)
  * @since 8.0.0
  */
#endif

 default VALUE_GENERIC_TYPE MERGE(final KEY_GENERIC_TYPE key, final VALUE_GENERIC_TYPE value, final java.util.function.BiFunction<? super VALUE_GENERIC_CLASS, ? super VALUE_GENERIC_CLASS, ? extends VALUE_GENERIC_CLASS> remappingFunction) {
  java.util.Objects.requireNonNull(remappingFunction);
#if VALUES_REFERENCE
  java.util.Objects.requireNonNull(value);
#endif
  final VALUE_GENERIC_TYPE oldValue = GET_VALUE(key), drv = defaultReturnValue();
  final VALUE_GENERIC_TYPE newValue;
  if (oldValue != drv || containsKey(key)) {
   final VALUE_GENERIC_CLASS mergedValue = remappingFunction.apply(VALUE2OBJ(oldValue), VALUE2OBJ(value));
   if (mergedValue == null) { REMOVE_VALUE(key); return drv; }
   newValue = VALUE_CLASS2TYPE(mergedValue);
  } else {
   newValue = value;
  }
```

```
 put(key, newValue);
 return newValue;
 }

 /** {@inheritDoc}
  * <p>This default implementation just delegates to the corresponding {@link Map} method.
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 default VALUE_GENERIC_CLASS getOrDefault(final Object key, final VALUE_GENERIC_CLASS
defaultValue) {
 return Map.super.getOrDefault(key, defaultValue);
 }

 /** {@inheritDoc}
  * <p>This default implementation just delegates to the corresponding {@link Map} method.
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 default VALUE_GENERIC_CLASS putIfAbsent(final KEY_GENERIC_CLASS key, final
VALUE_GENERIC_CLASS value) {
 return Map.super.putIfAbsent(key, value);
 }

 /** {@inheritDoc}
  * <p>This default implementation just delegates to the corresponding {@link Map} method.
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 default boolean remove(final Object key, final Object value) {
 return Map.super.remove(key, value);
 }

 /** {@inheritDoc}
  * <p>This default implementation just delegates to the corresponding {@link Map} method.
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 default boolean replace(final KEY_GENERIC_CLASS key, final VALUE_GENERIC_CLASS oldValue, final
VALUE_GENERIC_CLASS newValue) {
 return Map.super.replace(key, oldValue, newValue);
 }

 /** {@inheritDoc}
  * <p>This default implementation just delegates to the corresponding {@link Map} method.
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
```

```
 default VALUE_GENERIC_CLASS replace(final KEY_GENERIC_CLASS key, final
VALUE_GENERIC_CLASS value) {
  return Map.super.replace(key, value);
 }

#if KEYS_PRIMITIVE

 /** {@inheritDoc}
  * <p>This default implementation just delegates to the corresponding {@link Map} method.
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 default VALUE_GENERIC_CLASS computeIfAbsent(final KEY_GENERIC_CLASS key, final
java.util.function.Function<? super KEY_GENERIC_CLASS, ? extends VALUE_GENERIC_CLASS>
mappingFunction) {
  return Map.super.computeIfAbsent(key, mappingFunction);
 }

 /** {@inheritDoc}
  * <p>This default implementation just delegates to the corresponding {@link Map} method.
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 default VALUE_GENERIC_CLASS computeIfPresent(final KEY_GENERIC_CLASS key, final
java.util.function.BiFunction<? super KEY_GENERIC_CLASS, ? super VALUE_GENERIC_CLASS, ? extends
VALUE_GENERIC_CLASS> remappingFunction) {
  return Map.super.computeIfPresent(key, remappingFunction);
 }

 /** {@inheritDoc}
  * <p>This default implementation just delegates to the corresponding {@link Map} method.
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 default VALUE_GENERIC_CLASS compute(final KEY_GENERIC_CLASS key, final
java.util.function.BiFunction<? super KEY_GENERIC_CLASS, ? super VALUE_GENERIC_CLASS, ? extends
VALUE_GENERIC_CLASS> remappingFunction) {
  return Map.super.compute(key, remappingFunction);
 }

#endif

 /** {@inheritDoc}
  * <p>This default implementation just delegates to the corresponding {@link Map} method.
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 default VALUE_GENERIC_CLASS merge(final KEY_GENERIC_CLASS key, final VALUE_GENERIC_CLASS
```

```
value, final java.util.function.BiFunction<? super VALUE_GENERIC_CLASS, ? super
VALUE_GENERIC_CLASS, ? extends VALUE_GENERIC_CLASS> remappingFunction) {
 return Map.super.merge(key, value, remappingFunction);
 }

#endif

 /** A type-specific {@link java.util.Map.Entry}; provides some additional methods
  *  that use polymorphism to avoid (un)boxing.
  *
  * @see java.util.Map.Entry
  */

 interface Entry KEY_VALUE_GENERIC extends Map.Entry
<KEY_GENERIC_CLASS,VALUE_GENERIC_CLASS> {

#if KEYS_PRIMITIVE
  /** Returns the key corresponding to this entry.
   * @see java.util.Map.Entry#getKey()
   */
  KEY_TYPE ENTRY_GET_KEY();

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  default KEY_GENERIC_CLASS getKey() {
   return KEY2OBJ(ENTRY_GET_KEY());
  }
#endif

#if VALUES_PRIMITIVE
  /** Returns the value corresponding to this entry.
   * @see java.util.Map.Entry#getValue()
   */
  VALUE_TYPE ENTRY_GET_VALUE();

  /** Replaces the value corresponding to this entry with the specified value (optional operation).
   * @see java.util.Map.Entry#setValue(Object)
   */
  VALUE_TYPE setValue(final VALUE_TYPE value);

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  default VALUE_GENERIC_CLASS getValue() {
   return VALUE2OBJ(ENTRY_GET_VALUE());
```

```
  }

  /** {@inheritDoc}
   * @deprecated Please use the corresponding type-specific method instead. */
  @Deprecated
  @Override
  default VALUE_GENERIC_CLASS setValue(final VALUE_GENERIC_CLASS value) {
   return VALUE2OBJ(setValue(VALUE_CLASS2TYPE(value)));
  }
#endif
 }
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/Map.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/Arrays.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/Function.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/Stack.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/BidirectionalIterator.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/Hash.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/src/it/unimi/dsi/fastutil/HashCommon.java
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-

a775574/src/it/unimi/dsi/fastutil/AbstractStack.java
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2010-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

#if KEYS_REFERENCE
import it.unimi.dsi.fastutil.Stack;
#endif

import java.util.Iterator;
import java.util.Collection;
import java.util.NoSuchElementException;

import it.unimi.dsi.fastutil.BigList;
import it.unimi.dsi.fastutil.BigListIterator;

/**  An abstract class providing basic methods for big lists implementing a type-specific big list interface. */

public abstract class ABSTRACT_BIG_LIST KEY_GENERIC extends ABSTRACT_COLLECTION
KEY_GENERIC implements BIG_LIST KEY_GENERIC, STACK KEY_GENERIC {

 protected ABSTRACT_BIG_LIST() {}

 /** Ensures that the given index is nonnegative and not greater than this big-list size.
  *
  * @param index an index.
  * @throws IndexOutOfBoundsException if the given index is negative or greater than this big-list size.
  */
 protected void ensureIndex(final long index) {
  if (index < 0)  throw new IndexOutOfBoundsException("Index (" + index + ") is negative");
  if (index > size64()) throw new IndexOutOfBoundsException("Index (" + index + ") is greater than list size (" +
```

```
(size64()) + ")");
 }

 /** Ensures that the given index is nonnegative and smaller than this big-list size.
  *
  * @param index an index.
  * @throws IndexOutOfBoundsException if the given index is negative or not smaller than this big-list size.
  */
 protected void ensureRestrictedIndex(final long index) {
  if (index < 0)  throw new IndexOutOfBoundsException("Index (" + index + ") is negative");
  if (index >= size64()) throw new IndexOutOfBoundsException("Index (" + index + ") is greater than or equal to list
size (" + (size64()) + ")");
 }

 /** {@inheritDoc}
  *
  * <p>This implementation always throws an {@link UnsupportedOperationException}.
  */
 @Override
 public void add(final long index, final KEY_GENERIC_TYPE k) {
  throw new UnsupportedOperationException();
 }

 /** {@inheritDoc}
  *
  * <p>This implementation delegates to the type-specific version of {@link BigList#add(long, Object)}.
  */
 @Override
 public boolean add(final KEY_GENERIC_TYPE k) {
  add(size64(), k);
  return true;
 }

 /** {@inheritDoc}
  *
  * <p>This implementation always throws an {@link UnsupportedOperationException}.
  */
 @Override
 public KEY_GENERIC_TYPE REMOVE_KEY(long i) {
  throw new UnsupportedOperationException();
 }

 /** {@inheritDoc}
  *
  * <p>This implementation always throws an {@link UnsupportedOperationException}.
  */
 @Override
 public KEY_GENERIC_TYPE set(final long index, final KEY_GENERIC_TYPE k) {
```

```
 throw new UnsupportedOperationException();
}

/** Adds all of the elements in the specified collection to this list (optional operation). */
@Override
public boolean addAll(long index, final Collection<? extends KEY_GENERIC_CLASS> c) {
 ensureIndex(index);
 final Iterator<? extends KEY_GENERIC_CLASS> i = c.iterator();
 final boolean retVal = i.hasNext();
 while(i.hasNext()) add(index++, i.next());
 return retVal;
}

/** {@inheritDoc}
 *
 * <p>This implementation delegates to the type-specific version of {@link BigList#addAll(long, Collection)}.
 */
@Override
public boolean addAll(final Collection<? extends KEY_GENERIC_CLASS> c) {
 return addAll(size64(), c);
}

/** {@inheritDoc}
 *
 * <p>This implementation delegates to {@link #listIterator()}.
 */
@Override
public KEY_BIG_LIST_ITERATOR KEY_GENERIC iterator() {
 return listIterator();
}

/** {@inheritDoc}
 *
 * <p>This implementation delegates to {@link BigList#listIterator(long) listIterator(0)}.
 */
@Override
public KEY_BIG_LIST_ITERATOR KEY_GENERIC listIterator() {
 return listIterator(0L);
}

/** {@inheritDoc}
 * <p>This implementation is based on the random-access methods. */
@Override
public KEY_BIG_LIST_ITERATOR KEY_GENERIC listIterator(final long index) {
 ensureIndex(index);

 return new KEY_BIG_LIST_ITERATOR KEY_GENERIC() {
   long pos = index, last = -1;
```

```
    @Override
    public boolean hasNext() { return pos < ABSTRACT_BIG_LIST.this.size64(); }

    @Override
    public boolean hasPrevious() { return pos > 0; }

    @Override
    public KEY_GENERIC_TYPE NEXT_KEY() { if (! hasNext()) throw new NoSuchElementException(); return
ABSTRACT_BIG_LIST.this.GET_KEY(last = pos++); }

    @Override
    public KEY_GENERIC_TYPE PREV_KEY() { if (! hasPrevious()) throw new NoSuchElementException(); return
ABSTRACT_BIG_LIST.this.GET_KEY(last = --pos); }

    @Override
    public long nextIndex() { return pos; }

    @Override
    public long previousIndex() { return pos - 1; }

    @Override
    public void add(KEY_GENERIC_TYPE k) {
     ABSTRACT_BIG_LIST.this.add(pos++, k);
     last = -1;
    }

    @Override
    public void set(KEY_GENERIC_TYPE k) {
     if (last == -1) throw new IllegalStateException();
     ABSTRACT_BIG_LIST.this.set(last, k);
    }

    @Override
    public void remove() {
     if (last == -1) throw new IllegalStateException();
     ABSTRACT_BIG_LIST.this.REMOVE_KEY(last);
     /* If the last operation was a next(), we are removing an element *before* us, and we must decrease pos
correspondingly. */
     if (last < pos) pos--;
     last = -1;
    }
   };
  }


  /** Returns true if this list contains the specified element.
   * <p>This implementation delegates to {@code indexOf()}.
```

```
 * @see BigList#contains(Object)
 */
@Override
public boolean contains(final KEY_TYPE k) { return indexOf(k) >= 0; }

@Override
public long indexOf(final KEY_TYPE k) {
 final KEY_BIG_LIST_ITERATOR KEY_GENERIC i = listIterator();
 KEY_GENERIC_TYPE e;
 while(i.hasNext()) {
  e = i.NEXT_KEY();
  if (KEY_EQUALS(k, e)) return i.previousIndex();
 }
 return -1;
}

@Override
public long lastIndexOf(final KEY_TYPE k) {
 KEY_BIG_LIST_ITERATOR KEY_GENERIC i = listIterator(size64());
 KEY_GENERIC_TYPE e;
 while(i.hasPrevious()) {
  e = i.PREV_KEY();
  if (KEY_EQUALS(k, e)) return i.nextIndex();
 }
 return -1;
}

@Override
public void size(final long size) {
 long i = size64();
 if (size > i) while(i++ < size) add(KEY_NULL);
 else while(i-- != size) remove(i);
}

@Override
public BIG_LIST KEY_GENERIC subList(final long from, final long to) {
 ensureIndex(from);
 ensureIndex(to);
 if (from > to) throw new IndexOutOfBoundsException("Start index (" + from + ") is greater than end index (" + to +
")");

 return new SUBLIST KEY_GENERIC_DIAMOND(this, from, to);
}

/** {@inheritDoc}
 *
 * <p>This is a trivial iterator-based implementation. It is expected that
 * implementations will override this method with a more optimized version.
```

```
 */
@Override
public void removeElements(final long from, final long to) {
 ensureIndex(to);
 KEY_BIG_LIST_ITERATOR KEY_GENERIC i = listIterator(from);
 long n = to - from;
 if (n < 0) throw new IllegalArgumentException("Start index (" + from + ") is greater than end index (" + to + ")");
 while(n-- != 0) {
  i.NEXT_KEY();
  i.remove();
 }
}

/** {@inheritDoc}
 *
 * <p>This is a trivial iterator-based implementation. It is expected that
 * implementations will override this method with a more optimized version.
 */
@Override
public void addElements(long index, final KEY_GENERIC_TYPE a[][], long offset, long length) {
 ensureIndex(index);
 BIG_ARRAYS.ensureOffsetLength(a, offset, length);
 while(length-- != 0) add(index++, BIG_ARRAYS.get(a, offset++));
}

/** {@inheritDoc}
 *
 * <p>This implementation delegates to the analogous method for big-array fragments.
 */
@Override
public void addElements(final long index, final KEY_GENERIC_TYPE a[][]) {
 addElements(index, a, 0, BIG_ARRAYS.length(a));
}

/** {@inheritDoc}
 *
 * <p>This is a trivial iterator-based implementation. It is expected that
 * implementations will override this method with a more optimized version.
 */
@Override
public void getElements(final long from, final KEY_TYPE a[][], long offset, long length) {
 KEY_BIG_LIST_ITERATOR KEY_GENERIC i = listIterator(from);
 BIG_ARRAYS.ensureOffsetLength(a, offset, length);
 if (from + length > size64()) throw new IndexOutOfBoundsException("End index (" + (from + length) + ") is
greater than list size (" + size64() + ")");
 while(length-- != 0) BIG_ARRAYS.set(a, offset++, i.NEXT_KEY());
 }
```

```
 /** {@inheritDoc}
  * <p>This implementation delegates to {@link #removeElements(long, long)}.*/
 @Override
 public void clear() {
  removeElements(0, size64());
 }

 /** {@inheritDoc}
  *
  * <p>This implementation delegates to {@link #size64()}.
  * @deprecated Please use {@link #size64()} instead. */
 @Override
 @Deprecated
 public int size() {
  return (int)Math.min(Integer.MAX_VALUE, size64());
 }

#if ! KEY_CLASS_Reference
 private boolean valEquals(final Object a, final Object b) { return a == null ? b == null : a.equals(b); }
#endif

 /** Returns the hash code for this big list, which is identical to {@link java.util.List#hashCode()}.
  *
  * @return the hash code for this big list.
  */
 @Override
 public int hashCode() {
  KEY_ITERATOR KEY_GENERIC i = iterator();
  int h = 1;
  long s = size64();
  while (s-- != 0) {
   KEY_GENERIC_TYPE k = i.NEXT_KEY();
   h = 31 * h + KEY2JAVAHASH(k);
  }
  return h;
 }

 @Override
 public boolean equals(final Object o) {
  if (o == this) return true;
  if (! (o instanceof BigList)) return false;
  final BigList<?> l = (BigList<?>)o;
  long s = size64();
  if (s != l.size64()) return false;

#if KEYS_PRIMITIVE
  if (l instanceof BIG_LIST) {
   final KEY_BIG_LIST_ITERATOR KEY_GENERIC i1 = listIterator(), i2 = ((BIG_LIST
```

```
KEY_GENERIC)l).listIterator();
  while(s-- != 0) if (i1.NEXT_KEY() != i2.NEXT_KEY()) return false;
  return true;
 }
#endif

 final BigListIterator<?> i1 = listIterator(), i2 = l.listIterator();

#if KEY_CLASS_Reference
 while(s-- !=  0) if (i1.next() != i2.next()) return false;
#else
 while(s-- !=  0) if (! valEquals(i1.next(), i2.next())) return false;
#endif
 return true;
}

#if ! KEY_CLASS_Reference
/** Compares this big list to another object. If the
 * argument is a {@link BigList}, this method performs a lexicographical comparison; otherwise,
 * it throws a {@code ClassCastException}.
 *
 * @param l a big list.
 * @return if the argument is a {@link BigList}, a negative integer,
 * zero, or a positive integer as this list is lexicographically less than, equal
 * to, or greater than the argument.
 * @throws ClassCastException if the argument is not a big list.
 */

SUPPRESS_WARNINGS_KEY_UNCHECKED
@Override
public int compareTo(final BigList<? extends KEY_GENERIC_CLASS> l) {
 if (l == this) return 0;

 if (l instanceof BIG_LIST) {

  final KEY_BIG_LIST_ITERATOR KEY_GENERIC i1 = listIterator(), i2 = ((BIG_LIST
KEY_GENERIC)l).listIterator();
  int r;
  KEY_GENERIC_TYPE e1, e2;

  while(i1.hasNext() && i2.hasNext()) {
   e1 = i1.NEXT_KEY();
   e2 = i2.NEXT_KEY();
   if ((r = KEY_CMP(e1, e2)) != 0) return r;
  }
  return i2.hasNext() ? -1 : (i1.hasNext() ? 1 : 0);
 }
```

```
 BigListIterator<? extends KEY_GENERIC_CLASS> i1 = listIterator(), i2 = l.listIterator();
 int r;

 while(i1.hasNext() && i2.hasNext()) {
  if ((r = ((Comparable<? super KEY_GENERIC_CLASS>)i1.next()).compareTo(i2.next())) != 0) return r;
 }
 return i2.hasNext() ? -1 : (i1.hasNext() ? 1 : 0);
}
#endif

@Override
public void push(KEY_GENERIC_TYPE o) {
 add(o);
}

@Override
public KEY_GENERIC_TYPE POP() {
 if (isEmpty()) throw new NoSuchElementException();
 return REMOVE_KEY(size64() - 1);
}

@Override
public KEY_GENERIC_TYPE TOP() {
 if (isEmpty()) throw new NoSuchElementException();
 return GET_KEY(size64() - 1);
}

@Override
public KEY_GENERIC_TYPE PEEK(int i) {
 return GET_KEY(size64() - 1 - i);
}


#if KEYS_PRIMITIVE
 /** Removes a single instance of the specified element from this collection, if it is present (optional operation).
  * <p>This implementation delegates to {@code indexOf()}.
  * @see BigList#remove(Object)
  */
 @Override
 public boolean rem(KEY_TYPE k) {
  long index = indexOf(k);
  if (index == -1) return false;
  REMOVE_KEY(index);
  return true;
 }

 /** {@inheritDoc}
  *
```

```
 * <p>This implementation delegates to the type-specific version of {@link #addAll(long, Collection)}.
 */
@Override
public boolean addAll(final long index, final COLLECTION c) { return addAll(index, (Collection<? extends KEY_CLASS>)c); }

/** {@inheritDoc}
 *
 * <p>This implementation delegates to the type-specific version of {@link #addAll(long, Collection)}.
 */
@Override
public boolean addAll(final long index, final BIG_LIST l) { return addAll(index, (COLLECTION)l); }

/** {@inheritDoc}
 *
 * <p>This implementation delegates to the type-specific version of {@link #addAll(long, Collection)}.
 */
@Override
public boolean addAll(final COLLECTION c) { return addAll(size64(), c); }

/** {@inheritDoc}
 *
 * <p>This implementation delegates to the type-specific list version of {@link #addAll(long, Collection)}.
 */
@Override
public boolean addAll(final BIG_LIST l) { return addAll(size64(), l); }

/** {@inheritDoc}
 *
 * <p>This implementation delegates to the corresponding type-specific method.
 * @deprecated Please use the corresponding type-specific method instead.
 */
@Deprecated
@Override
public void add(final long index, final KEY_CLASS ok) { add(index, ok.KEY_VALUE()); }

/** {@inheritDoc}
 *
 * <p>This implementation delegates to the corresponding type-specific method.
 * @deprecated Please use the corresponding type-specific method instead.
 */
@Deprecated
@Override
public KEY_CLASS set(final long index, final KEY_CLASS ok) { return KEY2OBJ(set(index, ok.KEY_VALUE())); }

/** {@inheritDoc}
 *
```

```
 * <p>This implementation delegates to the corresponding type-specific method.
 * @deprecated Please use the corresponding type-specific method instead.
 */
@Deprecated
@Override
public KEY_CLASS get(final long index) { return KEY2OBJ(GET_KEY(index)); }

/** {@inheritDoc}
 *
 * <p>This implementation delegates to the corresponding type-specific method.
 * @deprecated Please use the corresponding type-specific method instead.
 */
@Deprecated
@Override
public long indexOf(final Object ok) { return indexOf(KEY_OBJ2TYPE(ok)); }

/** {@inheritDoc}
 *
 * <p>This implementation delegates to the corresponding type-specific method.
 * @deprecated Please use the corresponding type-specific method instead.
 */
@Deprecated
@Override
public long lastIndexOf(final Object ok) { return lastIndexOf(KEY_OBJ2TYPE(ok)); }

/** {@inheritDoc}
 *
 * <p>This implementation delegates to the corresponding type-specific method.
 * @deprecated Please use the corresponding type-specific method instead.
 */
@Deprecated
@Override
public KEY_CLASS remove(final long index) { return KEY2OBJ(REMOVE_KEY(index)); }

/** {@inheritDoc}
 *
 * <p>This implementation delegates to the corresponding type-specific method.
 * @deprecated Please use the corresponding type-specific method instead.
 */
@Deprecated
@Override
public void push(KEY_CLASS o) { push(o.KEY_VALUE()); }

/** {@inheritDoc}
 *
 * <p>This implementation delegates to the corresponding type-specific method.
 * @deprecated Please use the corresponding type-specific method instead.
 */
```

```
 @Deprecated
 @Override
 public KEY_CLASS pop() { return KEY_CLASS.valueOf(POP()); }

 /** {@inheritDoc}
  *
  * <p>This implementation delegates to the corresponding type-specific method.
  * @deprecated Please use the corresponding type-specific method instead.
  */
 @Deprecated
 @Override
 public KEY_CLASS top() { return KEY_CLASS.valueOf(TOP()); }

 /** {@inheritDoc}
  *
  * <p>This implementation delegates to the corresponding type-specific method.
  * @deprecated Please use the corresponding type-specific method instead.
  */
 @Deprecated
 @Override
 public KEY_CLASS peek(int i) { return KEY_CLASS.valueOf(PEEK(i)); }

#endif

 @Override
 public String toString() {
  final StringBuilder s = new StringBuilder();
  final KEY_ITERATOR KEY_GENERIC i = iterator();
  long n = size64();
  KEY_GENERIC_TYPE k;
  boolean first = true;

  s.append("[");

  while(n-- != 0) {
   if (first) first = false;
   else s.append(", ");
   k = i.NEXT_KEY();
#if KEYS_REFERENCE
   if (this == k) s.append("(this big list)"); else
#endif
    s.append(String.valueOf(k));
  }

  s.append("]");
  return s.toString();
 }
```

```
 /** A class implementing a sublist view. */
 public static class SUBLIST KEY_GENERIC extends ABSTRACT_BIG_LIST KEY_GENERIC implements
java.io.Serializable {
  private static final long serialVersionUID = -7046029254386353129L;
  /** The list this sublist restricts. */
  protected final BIG_LIST KEY_GENERIC l;
  /** Initial (inclusive) index of this sublist. */
  protected final long from;
  /** Final (exclusive) index of this sublist. */
  protected long to;

  public SUBLIST(final BIG_LIST KEY_GENERIC l, final long from, final long to) {
   this.l = l;
   this.from = from;
   this.to = to;
  }

  private boolean assertRange() {
   assert from <= l.size64();
   assert to <= l.size64();
   assert to >= from;
   return true;
  }

  @Override
  public boolean add(final KEY_GENERIC_TYPE k) {
   l.add(to, k);
   to++;
   assert assertRange();
   return true;
  }

  @Override
  public void add(final long index, final KEY_GENERIC_TYPE k) {
   ensureIndex(index);
   l.add(from + index, k);
   to++;
   assert assertRange();
  }

  @Override
  public boolean addAll(final long index, final Collection<? extends KEY_GENERIC_CLASS> c) {
   ensureIndex(index);
   to += c.size();
   return l.addAll(from + index, c);
  }
```

```
 @Override
 public KEY_GENERIC_TYPE GET_KEY(long index) {
  ensureRestrictedIndex(index);
  return l.GET_KEY(from + index);
 }

 @Override
 public KEY_GENERIC_TYPE REMOVE_KEY(long index) {
  ensureRestrictedIndex(index);
  to--;
  return l.REMOVE_KEY(from + index);
 }

 @Override
 public KEY_GENERIC_TYPE set(long index, KEY_GENERIC_TYPE k) {
  ensureRestrictedIndex(index);
  return l.set(from + index, k);
 }

 @Override
 public long size64() { return to - from; }

 @Override
 public void getElements(final long from, final KEY_TYPE[][] a, final long offset, final long length) {
  ensureIndex(from);
  if (from + length > size64())  throw new IndexOutOfBoundsException("End index (" + from + length + ") is greater
than list size (" + size64() + ")");
  l.getElements(this.from + from, a, offset, length);
 }

 @Override
 public void removeElements(final long from, final long to) {
  ensureIndex(from);
  ensureIndex(to);
  l.removeElements(this.from + from, this.from + to);
  this.to -= (to - from);
  assert assertRange();
 }

 @Override
 public void addElements(final long index, final KEY_GENERIC_TYPE a[][], long offset, long length) {
  ensureIndex(index);
  l.addElements(this.from + index, a, offset, length);
  this.to += length;
  assert assertRange();
 }

 @Override
```

```
  public KEY_BIG_LIST_ITERATOR KEY_GENERIC listIterator(final long index) {
   ensureIndex(index);

   return new KEY_BIG_LIST_ITERATOR KEY_GENERIC() {
    long pos = index, last = -1;

    @Override
    public boolean hasNext() { return pos < size64(); }
    @Override
    public boolean hasPrevious() { return pos > 0; }
    @Override
    public KEY_GENERIC_TYPE NEXT_KEY() { if (! hasNext()) throw new NoSuchElementException(); return
l.GET_KEY(from + (last = pos++)); }
    @Override
    public KEY_GENERIC_TYPE PREV_KEY() { if (! hasPrevious()) throw new NoSuchElementException(); return
l.GET_KEY(from + (last = --pos)); }
    @Override
    public long nextIndex() { return pos; }
    @Override
    public long previousIndex() { return pos - 1; }
    @Override
    public void add(KEY_GENERIC_TYPE k) {
     if (last == -1) throw new IllegalStateException();
     SUBLIST.this.add(pos++, k);
     last = -1;
     assert assertRange();
    }
    @Override
    public void set(KEY_GENERIC_TYPE k) {
     if (last == -1) throw new IllegalStateException();
     SUBLIST.this.set(last, k);
    }
    @Override
    public void remove() {
     if (last == -1) throw new IllegalStateException();
     SUBLIST.this.REMOVE_KEY(last);
     /* If the last operation was a next(), we are removing an element *before* us, and we must decrease pos
correspondingly. */
     if (last < pos) pos--;
     last = -1;
     assert assertRange();
    }
   };
  }

  @Override
  public BIG_LIST KEY_GENERIC subList(final long from, final long to) {
   ensureIndex(from);
```

```
  ensureIndex(to);
  if (from > to) throw new IllegalArgumentException("Start index (" + from + ") is greater than end index (" + to +
")");

  return new SUBLIST KEY_GENERIC_DIAMOND(this, from, to);
  }

#if KEYS_PRIMITIVE
  @Override
  public boolean rem(KEY_TYPE k) {
  long index = indexOf(k);
  if (index == -1) return false;
  to--;
  l.REMOVE_KEY(from + index);
  assert assertRange();
  return true;
  }

  @Override
  public boolean addAll(final long index, final COLLECTION c) {
  ensureIndex(index);
  return super.addAll(index, c);
  }

  @Override
  public boolean addAll(final long index, final BIG_LIST l) {
  ensureIndex(index);
  return super.addAll(index, l);
  }
#endif
 }
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/AbstractBigList.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
```

```
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import it.unimi.dsi.fastutil.BigArrays;
import it.unimi.dsi.fastutil.Hash;
import it.unimi.dsi.fastutil.Size64;
import it.unimi.dsi.fastutil.HashCommon;
import static it.unimi.dsi.fastutil.HashCommon.bigArraySize;
import static it.unimi.dsi.fastutil.HashCommon.maxFill;

import java.util.Collection;
import java.util.Iterator;
import java.util.NoSuchElementException;


/**  A type-specific hash big set with with a fast, small-footprint implementation.
*
* <p>Instances of this class use a hash table to represent a big set: the number
* of elements in the set is limited only by the amount of core memory. The table
* (backed by a {@linkplain it.unimi.dsi.fastutil.BigArrays big array}) is
* filled up to a specified <em>load factor</em>, and then doubled in size to
* accommodate new entries. If the table is emptied below <em>one fourth</em>
* of the load factor, it is halved in size; however, the table is never reduced to a
* size smaller than that at creation time: this approach makes it
* possible to create sets with a large capacity in which insertions and
* deletions do not cause immediately rehashing. Moreover, halving is
* not performed when deleting entries from an iterator, as it would interfere
* with the iteration process.
*
* <p>Note that {@link #clear()} does not modify the hash table size.
* Rather, a family of {@linkplain #trim() trimming
* methods} lets you control the size of the table; this is particularly useful
* if you reuse instances of this class.
*
* <p>The methods of this class are about 30% slower than those of the corresponding non-big set.
*
* @see Hash
* @see HashCommon
*/

public class OPEN_HASH_BIG_SET KEY_GENERIC extends ABSTRACT_SET KEY_GENERIC implements
java.io.Serializable, Cloneable, Hash, Size64 {
```

```
private static final long serialVersionUID = 0L;
private static final boolean ASSERTS = ASSERTS_VALUE;

/** The big array of keys. */
protected transient KEY_GENERIC_TYPE[][] key;

/** The mask for wrapping a position counter. */
protected transient long mask;

/** The mask for wrapping a segment counter. */
protected transient int segmentMask;

/** The mask for wrapping a base counter. */
protected transient int baseMask;

/** Whether this set contains the null key. */
protected transient boolean containsNull;

/** The current table size (always a power of 2). */
protected transient long n;

/** Threshold after which we rehash. It must be the table size times {@link #f}. */
protected transient long maxFill;

/** We never resize below this threshold, which is the construction-time {#n}. */
protected final transient long minN;

/** The acceptable load factor. */
protected final float f;

/** Number of entries in the set. */
protected long size;


/** Initialises the mask values. */
private void initMasks() {
 mask = n - 1;
 /* Note that either we have more than one segment, and in this case all segments
  * are BigArrays.SEGMENT_SIZE long, or we have exactly one segment whose length
  * is a power of two. */
 segmentMask = key[0].length - 1;
 baseMask = key.length - 1;
}

/** Creates a new hash big set.
 *
 * <p>The actual table size will be the least power of two greater than {@code expected}/{@code f}.
```

```
 *
 * @param expected the expected number of elements in the set.
 * @param f the load factor.
 */
SUPPRESS_WARNINGS_KEY_UNCHECKED
public OPEN_HASH_BIG_SET(final long expected, final float f) {
 if (f <= 0 || f > 1) throw new IllegalArgumentException("Load factor must be greater than 0 and smaller than or
equal to 1");
 if (n < 0) throw new IllegalArgumentException("The expected number of elements must be nonnegative");

 this.f = f;

 minN = n = bigArraySize(expected, f);
 maxFill = maxFill(n, f);
 key = KEY_GENERIC_BIG_ARRAY_CAST BIG_ARRAYS.newBigArray(n);
 initMasks();
}


/** Creates a new hash big set with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor.
 *
 * @param expected the expected number of elements in the hash big set.
 */

public OPEN_HASH_BIG_SET(final long expected) {
 this(expected, DEFAULT_LOAD_FACTOR);
}

/** Creates a new hash big set with initial expected {@link Hash#DEFAULT_INITIAL_SIZE} elements
 * and {@link Hash#DEFAULT_LOAD_FACTOR} as load factor.
 */

public OPEN_HASH_BIG_SET() {
 this(DEFAULT_INITIAL_SIZE, DEFAULT_LOAD_FACTOR);
}

/** Creates a new hash big set copying a given collection.
 *
 * @param c a {@link Collection} to be copied into the new hash big set.
 * @param f the load factor.
 */

public OPEN_HASH_BIG_SET(final Collection<? extends KEY_GENERIC_CLASS> c, final float f) {
 this(c.size(), f);
 addAll(c);
}

/** Creates a new hash big set  with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor
```

```
 * copying a given collection.
 *
 * @param c a {@link Collection} to be copied into the new hash big set.
 */

public OPEN_HASH_BIG_SET(final Collection<? extends KEY_GENERIC_CLASS> c) {
 this(c, DEFAULT_LOAD_FACTOR);
}

/** Creates a new hash big set copying a given type-specific collection.
 *
 * @param c a type-specific collection to be copied into the new hash big set.
 * @param f the load factor.
 */

public OPEN_HASH_BIG_SET(final COLLECTION KEY_EXTENDS_GENERIC c, final float f) {
 this(c.size(), f);
 addAll(c);
}

/** Creates a new hash big set  with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor
 * copying a given type-specific collection.
 *
 * @param c a type-specific collection to be copied into the new hash big set.
 */

public OPEN_HASH_BIG_SET(final COLLECTION KEY_EXTENDS_GENERIC c) {
 this(c, DEFAULT_LOAD_FACTOR);
}

/** Creates a new hash big set using elements provided by a type-specific iterator.
 *
 * @param i a type-specific iterator whose elements will fill the new hash big set.
 * @param f the load factor.
 */

public OPEN_HASH_BIG_SET(final STD_KEY_ITERATOR KEY_EXTENDS_GENERIC i, final float f) {
 this(DEFAULT_INITIAL_SIZE, f);
 while(i.hasNext()) add(i.NEXT_KEY());
}

/** Creates a new hash big set with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor using elements
provided by a type-specific iterator.
 *
 * @param i a type-specific iterator whose elements will fill the new hash big set.
 */

public OPEN_HASH_BIG_SET(final STD_KEY_ITERATOR KEY_EXTENDS_GENERIC i) {
```

```
 this(i, DEFAULT_LOAD_FACTOR);
 }


#if KEYS_PRIMITIVE

 /** Creates a new hash big set using elements provided by an iterator.
  *
  * @param i an iterator whose elements will fill the new hash big set.
  * @param f the load factor.
  */

 public OPEN_HASH_BIG_SET(final Iterator<?> i, final float f) {
  this(ITERATORS.AS_KEY_ITERATOR(i), f);
 }
 /** Creates a new hash big set with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor using elements
provided by an iterator.
  *
  * @param i an iterator whose elements will fill the new hash big set.
  */

 public OPEN_HASH_BIG_SET(final Iterator<?> i) {
  this(ITERATORS.AS_KEY_ITERATOR(i));
 }

#endif


 /** Creates a new hash big set and fills it with the elements of a given array.
  *
  * @param a an array whose elements will be used to fill the new hash big set.
  * @param offset the first element to use.
  * @param length the number of elements to use.
  * @param f the load factor.
  */

 public OPEN_HASH_BIG_SET(final KEY_GENERIC_TYPE[] a, final int offset, final int length, final float f) {
  this(length < 0 ? 0 : length, f);
  ARRAYS.ensureOffsetLength(a, offset, length);
  for(int i = 0; i < length; i++) add(a[offset + i]);
 }

 /** Creates a new hash big set with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor and fills it with the
elements of a given array.
  *
  * @param a an array whose elements will be used to fill the new hash big set.
  * @param offset the first element to use.
  * @param length the number of elements to use.
```

```
 */

public OPEN_HASH_BIG_SET(final KEY_GENERIC_TYPE[] a, final int offset, final int length) {
 this(a, offset, length, DEFAULT_LOAD_FACTOR);
}

/** Creates a new hash big set copying the elements of an array.
 *
 * @param a an array to be copied into the new hash big set.
 * @param f the load factor.
 */

public OPEN_HASH_BIG_SET(final KEY_GENERIC_TYPE[] a, final float f) {
 this(a, 0, a.length, f);
}

/** Creates a new hash big set with {@link Hash#DEFAULT_LOAD_FACTOR} as load factor
 * copying the elements of an array.
 *
 * @param a an array to be copied into the new hash big set.
 */

public OPEN_HASH_BIG_SET(final KEY_GENERIC_TYPE[] a) {
 this(a, DEFAULT_LOAD_FACTOR);
}

private long realSize() {
 return containsNull ? size - 1 : size;
}

private void ensureCapacity(final long capacity) {
 final long needed = bigArraySize(capacity, f);
 if (needed > n) rehash(needed);
}

@Override
public boolean addAll(Collection<? extends KEY_GENERIC_CLASS> c) {
 final long size = c instanceof Size64 ? ((Size64)c).size64() : c.size();
 // The resulting collection will be at least c.size() big
 if (f <= .5) ensureCapacity(size); // The resulting collection will be sized for c.size() elements
 else ensureCapacity(size64() + size); // The resulting collection will be sized for size() + c.size() elements
 return super.addAll(c);
}

#if KEYS_PRIMITIVE
 @Override
 public boolean addAll(COLLECTION c) {
  final long size = c instanceof Size64 ? ((Size64)c).size64() : c.size();
```

```
 if (f <= .5) ensureCapacity(size); // The resulting collection will be size for c.size() elements
 else ensureCapacity(size64() + size); // The resulting collection will be sized for size() + c.size() elements
 return super.addAll(c);
 }
#endif

 @Override
 public boolean add(final KEY_GENERIC_TYPE k) {
  int displ, base;

  if (KEY_IS_NULL(k)) {
   if (containsNull) return false;
   containsNull = true;
  }
  else {
   KEY_GENERIC_TYPE curr;
   final KEY_GENERIC_TYPE[][] key = this.key;
   final long h = KEY2LONGHASH(k);

   // The starting point.
   if (! KEY_IS_NULL(curr = key[base = (int)((h & mask) >>> BigArrays.SEGMENT_SHIFT)][displ = (int)(h &
segmentMask)])) {
    if (KEY_EQUALS_NOT_NULL(curr, k)) return false;
    while(! KEY_IS_NULL(curr = key[base = (base + ((displ = (displ + 1) & segmentMask) == 0 ? 1 : 0)) &
baseMask][displ]))
     if (KEY_EQUALS_NOT_NULL(curr, k)) return false;
   }

   key[base][displ] = k;
  }

  if (size++ >= maxFill) rehash(2 * n);
  if (ASSERTS) checkTable();
  return true;
 }

#if KEY_CLASS_Object
 /** Add a random element if not present, get the existing value if already present.
  *
  * This is equivalent to (but faster than) doing a:
  * <pre>
  * K exist = set.get(k);
  * if (exist == null) {
  *   set.add(k);
  *   exist = k;
  * }
  * </pre>
  */
```

```
 public KEY_GENERIC_TYPE addOrGet(final KEY_GENERIC_TYPE k) {
  int displ, base;

  if (KEY_IS_NULL(k)) {
   if (containsNull) return null;
   containsNull = true;
  }
  else {
   KEY_GENERIC_TYPE curr;
   final KEY_GENERIC_TYPE[][] key = this.key;
   final long h = KEY2LONGHASH(k);

   // The starting point.
   if (! KEY_IS_NULL(curr = key[base = (int)((h & mask) >>> BigArrays.SEGMENT_SHIFT)][displ = (int)(h &
segmentMask)])) {
    if (KEY_EQUALS_NOT_NULL(curr, k)) return curr;
    while(! KEY_IS_NULL(curr = key[base = (base + ((displ = (displ + 1) & segmentMask) == 0 ? 1 : 0)) &
baseMask][displ]))
     if (KEY_EQUALS_NOT_NULL(curr, k)) return curr;
   }

   key[base][displ] = k;
  }

  if (size++ >= maxFill) rehash(2 * n);
  if (ASSERTS) checkTable();
  return k;
 }
#endif

 /** Shifts left entries with the specified hash code, starting at the specified position,
  * and empties the resulting free entry.
  *
  * @param pos a starting position.
  */
 protected final void shiftKeys(long pos) {
  // Shift entries with the same hash.
  long last, slot;
  final KEY_GENERIC_TYPE[][] key = this.key;

  for(;;) {
   pos = ((last = pos) + 1) & mask;

   for(;;) {
    if (KEY_IS_NULL(BIG_ARRAYS.get(key, pos))) {
     BIG_ARRAYS.set(key, last, KEY_NULL);
     return;
    }
```

```
   slot = KEY2LONGHASH(BIG_ARRAYS.get(key, pos)) & mask;
   if (last <= pos ? last >= slot || slot > pos : last >= slot && slot > pos) break;
   pos = (pos + 1) & mask;
  }

  BIG_ARRAYS.set(key, last, BIG_ARRAYS.get(key, pos));
 }
}

private boolean removeEntry(final int base, final int displ) {
 size--;
 shiftKeys(base * (long)BigArrays.SEGMENT_SIZE + displ);
 if (n > minN && size < maxFill / 4 && n > DEFAULT_INITIAL_SIZE) rehash(n / 2);
 return true;
}

private boolean removeNullEntry() {
 containsNull = false;
 size--;
 if (n > minN && size < maxFill / 4 && n > DEFAULT_INITIAL_SIZE) rehash(n / 2);
 return true;
}

@Override
public boolean remove(final KEY_TYPE k) {
 if (KEY_IS_NULL(k)) {
  if (containsNull) return removeNullEntry();
  return false;
 }

 KEY_GENERIC_TYPE curr;
 final KEY_GENERIC_TYPE[][] key = this.key;
 final long h = KEY2LONGHASH(k);
 int displ, base;

 // The starting point.
 if (KEY_IS_NULL(curr = key[base = (int)((h & mask) >>> BigArrays.SEGMENT_SHIFT)][displ = (int)(h &
segmentMask)])) return false;
 if (KEY_EQUALS_NOT_NULL(curr, k)) return removeEntry(base, displ);
 while(true) {
  if (KEY_IS_NULL(curr = key[base = (base + ((displ = (displ + 1) & segmentMask) == 0 ? 1 : 0)) &
baseMask][displ])) return false;
  if (KEY_EQUALS_NOT_NULL(curr, k)) return removeEntry(base, displ);
 }
}

@Override
public boolean contains(final KEY_TYPE k) {
```

```
  if (KEY_IS_NULL(k)) return containsNull;

  KEY_GENERIC_TYPE curr;
  final KEY_GENERIC_TYPE[][] key = this.key;
  final long h = KEY2LONGHASH(k);
  int displ, base;

  // The starting point.
  if (KEY_IS_NULL(curr = key[base = (int)((h & mask) >>> BigArrays.SEGMENT_SHIFT)][displ = (int)(h & segmentMask)])) return false;
  if (KEY_EQUALS_NOT_NULL(curr, k)) return true;
  while(true) {
   if (KEY_IS_NULL(curr = key[base = (base + ((displ = (displ + 1) & segmentMask) == 0 ? 1 : 0)) & baseMask][displ])) return false;
   if (KEY_EQUALS_NOT_NULL(curr, k)) return true;
  }
 }

#if KEY_CLASS_Object
 /** Returns the element of this set that is equal to the given key, or {@code null}.
  * @return the element of this set that is equal to the given key, or {@code null}.
  */
 public K get(final KEY_TYPE k) {
  if (k == null) return null; // This is correct independently of the value of containsNull

  KEY_GENERIC_TYPE curr;
  final KEY_GENERIC_TYPE[][] key = this.key;
  final long h = KEY2LONGHASH(k);
  int displ, base;

  // The starting point.
  if (KEY_IS_NULL(curr = key[base = (int)((h & mask) >>> BigArrays.SEGMENT_SHIFT)][displ = (int)(h & segmentMask)])) return null;
  if (KEY_EQUALS_NOT_NULL(curr, k)) return curr;
  while(true) {
   if (KEY_IS_NULL(curr = key[base = (base + ((displ = (displ + 1) & segmentMask) == 0 ? 1 : 0)) & baseMask][displ])) return null;
   if (KEY_EQUALS_NOT_NULL(curr, k)) return curr;
  }
 }
#endif

 /* Removes all elements from this set.
  *
  */

 /** {@inheritDoc}
  *
```

```
 * <p>To increase object reuse, this method does not change the table size.
 * If you want to reduce the table size, you must use {@link #trim(long)}.
 */
@Override
public void clear() {
 if (size == 0) return;
 size = 0;
 containsNull = false;
 BIG_ARRAYS.fill(key, KEY_NULL);
}



/** An iterator over a hash big set. */

private class SetIterator implements KEY_ITERATOR KEY_GENERIC {
 /** The base of the last entry returned, if positive or zero; initially, the number of components
  of the key array. If negative, the last element returned was
  that of index {@code - base - 1} from the {@link #wrapped} list. */
 int base = key.length;
 /** The displacement of the last entry returned; initially, zero. */
 int displ;
 /** The index of the last entry that has been returned (or {@link Long#MIN_VALUE} if {@link #base} is
negative).
  It is -1 if either we did not return an entry yet, or the last returned entry has been removed. */
 long last = -1;
 /** A downward counter measuring how many entries must still be returned. */
 long c = size;
 /** A boolean telling us whether we should return the null key. */
 boolean mustReturnNull = OPEN_HASH_BIG_SET.this.containsNull;
 /** A lazily allocated list containing elements that have wrapped around the table because of removals. */
 ARRAY_LIST KEY_GENERIC wrapped;

 @Override
 public boolean hasNext() { return c != 0; }

 @Override
 public KEY_GENERIC_TYPE NEXT_KEY() {
  if (! hasNext()) throw new NoSuchElementException();
  c--;

  if (mustReturnNull) {
   mustReturnNull = false;
   last = n;
   return KEY_NULL;
  }

  final KEY_GENERIC_TYPE[][] key = OPEN_HASH_BIG_SET.this.key;
```

```
 for(;;) {
  if (displ == 0 && base <= 0) {
   // We are just enumerating elements from the wrapped list.
   last = Long.MIN_VALUE;
   return wrapped.GET_KEY(- (--base) - 1);
  }

  if (displ-- == 0) displ = key[--base].length - 1;

  final KEY_GENERIC_TYPE k = key[base][displ];
  if (! KEY_IS_NULL(k)) {
   last = base * (long)BigArrays.SEGMENT_SIZE + displ;
   return k;
  }
 }
}

/** Shifts left entries with the specified hash code, starting at the specified position,
 * and empties the resulting free entry.
 *
 * @param pos a starting position.
 */
private final void shiftKeys(long pos) {
 // Shift entries with the same hash.
 long last, slot;
 KEY_GENERIC_TYPE curr;
 final KEY_GENERIC_TYPE[][] key = OPEN_HASH_BIG_SET.this.key;

 for(;;) {
  pos = ((last = pos) + 1) & mask;

  for(;;) {
   if(KEY_IS_NULL(curr = BIG_ARRAYS.get(key, pos))) {
    BIG_ARRAYS.set(key, last, KEY_NULL);
    return;
   }
   slot = KEY2LONGHASH(curr) & mask;
   if (last <= pos ? last >= slot || slot > pos : last >= slot && slot > pos) break;
   pos = (pos + 1) & mask;
  }

  if (pos < last) { // Wrapped entry.
   if (wrapped == null) wrapped = new ARRAY_LIST KEY_GENERIC_DIAMOND();
   wrapped.add(BIG_ARRAYS.get(key, pos));
  }

  BIG_ARRAYS.set(key, last, curr);
```

```
  }
  }

  @Override
  public void remove() {
   if (last == -1) throw new IllegalStateException();
   if (last == n) OPEN_HASH_BIG_SET.this.containsNull = false;
   else if (base >= 0) shiftKeys(last);
   else {
    // We're removing wrapped entries.
#if KEYS_REFERENCE
    OPEN_HASH_BIG_SET.this.remove(wrapped.set(- base - 1, null));
#else
    OPEN_HASH_BIG_SET.this.remove(wrapped.GET_KEY(- base - 1));
#endif
    last = -1; // Note that we must not decrement size
    return;
   }

   size--;
   last = -1; // You can no longer remove this entry.
   if (ASSERTS) checkTable();
  }
 }

 @Override
 public KEY_ITERATOR KEY_GENERIC iterator() {
  return new SetIterator();
 }


 /** Rehashes this set, making the table as small as possible.
  *
  * <p>This method rehashes the table to the smallest size satisfying the
  * load factor. It can be used when the set will not be changed anymore, so
  * to optimize access speed and size.
  *
  * <p>If the table size is already the minimum possible, this method
  * does nothing.
  *
  * @return true if there was enough memory to trim the set.
  * @see #trim(long)
  */

 public boolean trim() {
  final long l = bigArraySize(size, f);
  if (l >= n || size > maxFill(l, f)) return true;
  try {
```

```
 rehash(l);
 }
 catch(OutOfMemoryError cantDoIt) { return false; }
 return true;
}

/** Rehashes this set if the table is too large.
 *
 * <p>Let <var>N</var> be the smallest table size that can hold
 * <code>max(n,{@link #size64()})</code> entries, still satisfying the load factor. If the current
 * table size is smaller than or equal to <var>N</var>, this method does
 * nothing. Otherwise, it rehashes this set in a table of size
 * <var>N</var>.
 *
 * <p>This method is useful when reusing sets.  {@linkplain #clear() Clearing a
 * set} leaves the table size untouched. If you are reusing a set
 * many times, you can call this method with a typical
 * size to avoid keeping around a very large table just
 * because of a few large transient sets.
 *
 * @param n the threshold for the trimming.
 * @return true if there was enough memory to trim the set.
 * @see #trim()
 */

public boolean trim(final long n) {
 final long l = bigArraySize(n, f);
 if (this.n <= l) return true;
 try {
  rehash(l);
 }
 catch(OutOfMemoryError cantDoIt) { return false; }
 return true;
}

/** Resizes the set.
 *
 * <p>This method implements the basic rehashing strategy, and may be
 * overriden by subclasses implementing different rehashing strategies (e.g.,
 * disk-based rehashing). However, you should not override this method
 * unless you understand the internal workings of this class.
 *
 * @param newN the new size
 */

SUPPRESS_WARNINGS_KEY_UNCHECKED
protected void rehash(final long newN) {
 final KEY_GENERIC_TYPE key[][] = this.key;
```

```
 final KEY_GENERIC_TYPE newKey[][] = KEY_GENERIC_BIG_ARRAY_CAST
BIG_ARRAYS.newBigArray(newN);
 final long mask = newN - 1; // Note that this is used by the hashing macro
 final int newSegmentMask = newKey[0].length - 1;
 final int newBaseMask = newKey.length - 1;

 int base = 0, displ = 0, b, d;
 long h;
 KEY_GENERIC_TYPE k;

 for(long i = realSize(); i-- != 0;) {

  while(KEY_IS_NULL(key[base][displ])) base = (base + ((displ = (displ + 1) & segmentMask) == 0 ? 1 : 0));

  k = key[base][displ];
  h = KEY2LONGHASH(k);

  // The starting point.
  if (! KEY_IS_NULL(newKey[b = (int)((h & mask) >>> BigArrays.SEGMENT_SHIFT)][d = (int)(h &
newSegmentMask)]))
   while(! KEY_IS_NULL(newKey[b = (b + ((d = (d + 1) & newSegmentMask) == 0 ? 1 : 0)) & newBaseMask][d]));

  newKey[b][d] = k;

  base = (base + ((displ = (displ + 1) & segmentMask) == 0 ? 1 : 0));
  }

 this.n = newN;
 this.key = newKey;
 initMasks();
 maxFill = maxFill(n, f);
 }

 @Deprecated
 @Override
 public int size() {
  return (int)Math.min(Integer.MAX_VALUE, size);
 }

 @Override
 public long size64() {
  return size;
 }

 @Override
 public boolean isEmpty() {
  return size == 0;
 }
```

```
/** Returns a deep copy of this big set.
 *
 * <p>This method performs a deep copy of this big hash set; the data stored in the
 * set, however, is not cloned. Note that this makes a difference only for object keys.
 *
 *  @return a deep copy of this big set.
 */
@Override
SUPPRESS_WARNINGS_KEY_UNCHECKED
public OPEN_HASH_BIG_SET KEY_GENERIC clone() {
 OPEN_HASH_BIG_SET KEY_GENERIC c;
 try {
  c = (OPEN_HASH_BIG_SET KEY_GENERIC)super.clone();
 }
 catch(CloneNotSupportedException cantHappen) {
  throw new InternalError();
 }
 c.key = BIG_ARRAYS.copy(key);
 c.containsNull = containsNull;
 return c;
}

/** Returns a hash code for this set.
 *
 * This method overrides the generic method provided by the superclass.
 * Since {@code equals()} is not overriden, it is important
 * that the value returned by this method is the same value as
 * the one returned by the overriden method.
 *
 * @return a hash code for this set.
 */

@Override
public int hashCode() {
 final KEY_GENERIC_TYPE key[][] = this.key;
 int h = 0, base = 0, displ = 0;

 for(long j = realSize(); j-- != 0;) {
  while(KEY_IS_NULL(key[base][displ])) base = (base + ((displ = (displ + 1) & segmentMask) == 0 ? 1 : 0));
#if KEYS_REFERENCE
  if (this != key[base][displ])
#endif
   h += KEY2JAVAHASH_NOT_NULL(key[base][displ]);
  base = (base + ((displ = (displ + 1) & segmentMask) == 0 ? 1 : 0));
 }
```

```
 return h;
 }


 private void writeObject(java.io.ObjectOutputStream s) throws java.io.IOException {
  final KEY_ITERATOR KEY_GENERIC i = iterator();
  s.defaultWriteObject();
  for(long j = size; j-- != 0;) s.WRITE_KEY(i.NEXT_KEY());
 }


 SUPPRESS_WARNINGS_KEY_UNCHECKED
 private void readObject(java.io.ObjectInputStream s) throws java.io.IOException, ClassNotFoundException {
  s.defaultReadObject();

  n = bigArraySize(size, f);
  maxFill = maxFill(n, f);

  final KEY_GENERIC_TYPE[][] key = this.key = KEY_GENERIC_BIG_ARRAY_CAST
BIG_ARRAYS.newBigArray(n);

  initMasks();

  long h;
  KEY_GENERIC_TYPE k;
  int base, displ;

  for(long i = size; i-- != 0;) {
   k = KEY_GENERIC_CAST s.READ_KEY();

   if (KEY_IS_NULL(k)) containsNull = true;
   else {
    h = KEY2LONGHASH(k);
    if (! KEY_IS_NULL(key[base = (int)((h & mask) >>> BigArrays.SEGMENT_SHIFT)][displ = (int)(h &
segmentMask)]))
     while(! KEY_IS_NULL(key[base = (base + ((displ = (displ + 1) & segmentMask) == 0 ? 1 : 0)) &
baseMask][displ]));
    key[base][displ] = k;
   }
  }

  if (ASSERTS) checkTable();
 }


#ifdef ASSERTS_CODE
 private void checkTable() {
  assert (n & -n) == n : "Table length is not a power of two: " + n;
```

```
  assert n == BIG_ARRAYS.length(key);
  long n = this.n;
  while(n-- != 0)
   if (! KEY_IS_NULL(BIG_ARRAYS.get(key, n)) && ! contains(BIG_ARRAYS.get(key, n)))
    throw new AssertionError("Hash table has key " + BIG_ARRAYS.get(key, n) + " marked as occupied, but the key
does not belong to the table");

#if KEYS_PRIMITIVE
  java.util.HashSet<KEY_GENERIC_CLASS> s = new java.util.HashSet<KEY_GENERIC_CLASS> ();
#else
  java.util.HashSet<Object> s = new java.util.HashSet<Object>();
#endif

  for(long i = size(); i-- != 0;)
   if (! KEY_IS_NULL(BIG_ARRAYS.get(key, i)) && ! s.add(BIG_ARRAYS.get(key, i))) throw new
AssertionError("Key " + BIG_ARRAYS.get(key, i) + " appears twice");

 }
#else
 private void checkTable() {}
#endif

#ifdef TEST

 private static long seed = System.currentTimeMillis();
 private static java.util.Random r = new java.util.Random(seed);

 private static KEY_TYPE genKey() {
#if KEY_CLASS_Byte || KEY_CLASS_Short || KEY_CLASS_Character
  return (KEY_TYPE)(r.nextInt());
#elif KEYS_PRIMITIVE
  return r.NEXT_KEY();
#elif KEY_CLASS_Object
  return Integer.toBinaryString(r.nextInt());
#else
  return new java.io.Serializable() {};
#endif
 }


 private static final class ArrayComparator implements java.util.Comparator {
  public int compare(Object a, Object b) {
   byte[] aa = (byte[])a;
   byte[] bb = (byte[])b;
   int length = Math.min(aa.length, bb.length);
   for(int i = 0; i < length; i++) {
    if (aa[i] < bb[i]) return -1;
    if (aa[i] > bb[i]) return 1;
```

```
 }
 return aa.length == bb.length ? 0 : (aa.length < bb.length ? -1 : 1);
 }
}

private static final class MockSet extends java.util.TreeSet {
 private java.util.List list = new java.util.ArrayList();

 public MockSet(java.util.Comparator c) { super(c); }

 public boolean add(Object k) {
  if (! contains(k)) list.add(k);
  return super.add(k);
 }

 public boolean addAll(Collection c) {
  java.util.Iterator i = c.iterator();
  boolean result = false;
  while(i.hasNext()) result |= add(i.next());
  return result;
 }

 public boolean removeAll(Collection c) {
  java.util.Iterator i = c.iterator();
  boolean result = false;
  while(i.hasNext()) result |= remove(i.next());
  return result;
 }

 public boolean remove(Object k) {
  if (contains(k)) {
   int i = list.size();
   while(i-- != 0) if (comparator().compare(list.get(i), k) == 0) {
    list.remove(i);
    break;
   }
  }
  return super.remove(k);
 }

 private void justRemove(Object k) { super.remove(k); }

 public java.util.Iterator iterator() {
  return new java.util.Iterator() {
    final java.util.Iterator iterator = list.iterator();
    Object curr;
    public Object next() { return curr = iterator.next(); }
    public boolean hasNext() { return iterator.hasNext(); }
```

```
   public void remove() {
    justRemove(curr);
    iterator.remove();
   }
  };
 }
}

private static java.text.NumberFormat format = new java.text.DecimalFormat("#,###.00");
private static java.text.FieldPosition fp = new java.text.FieldPosition(0);

private static String format(double d) {
 StringBuffer s = new StringBuffer();
 return format.format(d, s, fp).toString();
}

private static void speedTest(int n, float f, boolean comp) {
 int i, j;
 OPEN_HASH_BIG_SET m;
 java.util.HashSet t;

 KEY_TYPE k[] = new KEY_TYPE[n];
 KEY_TYPE nk[] = new KEY_TYPE[n];
 long ms;

 for(i = 0; i < n; i++) {
  k[i] = genKey();
  nk[i] = genKey();
 }

 double totAdd = 0, totYes = 0, totNo = 0, totIter = 0, totRemYes = 0, totRemNo = 0, d;

 if (comp) { for(j = 0; j < 20; j++) {

  t = new java.util.HashSet(16);

  /* We add pairs to t. */
  ms = System.currentTimeMillis();
  for(i = 0; i < n;  i++) t.add(KEY2OBJ(k[i]));
  d = 1.0 * n / (System.currentTimeMillis() - ms);
  if (j > 2) totAdd += d;
  System.out.print("Add: " + format(d) +" K/s ");

  /* We check for pairs in t. */
  ms = System.currentTimeMillis();
  for(i = 0; i < n;  i++) t.contains(KEY2OBJ(k[i]));
  d = 1.0 * n / (System.currentTimeMillis() - ms);
  if (j > 2) totYes += d;
```

```
 System.out.print("Yes: " + format(d) +" K/s ");

 /* We check for pairs not in t. */
 ms = System.currentTimeMillis();
 for(i = 0; i < n;  i++) t.contains(KEY2OBJ(nk[i]));
 d = 1.0 * n / (System.currentTimeMillis() - ms);
 if (j > 2) totNo += d;
 System.out.print("No: " + format(d) +" K/s ");

 /* We iterate on t. */
 ms = System.currentTimeMillis();
 for(java.util.Iterator it = t.iterator(); it.hasNext(); it.next());
 d = 1.0 * n / (System.currentTimeMillis() - ms);
 if (j > 2) totIter += d;
 System.out.print("Iter: " + format(d) +" K/s ");

 /* We delete pairs not in t. */
 ms = System.currentTimeMillis();
 for(i = 0; i < n;  i++) t.remove(KEY2OBJ(nk[i]));
 d = 1.0 * n / (System.currentTimeMillis() - ms);
 if (j > 2) totRemNo += d;
 System.out.print("RemNo: " + format(d) +" K/s ");

 /* We delete pairs in t. */
 ms = System.currentTimeMillis();
 for(i = 0; i < n;  i++) t.remove(KEY2OBJ(k[i]));
 d = 1.0 * n / (System.currentTimeMillis() - ms);
 if (j > 2) totRemYes += d;
 System.out.print("RemYes: " + format(d) +" K/s ");

 System.out.println();
 }

 System.out.println();
 System.out.println("java.util Add: " + format(totAdd/(j-3)) + " K/s Yes: " + format(totYes/(j-3)) + " K/s No: " +
format(totNo/(j-3)) + " K/s Iter: " + format(totIter/(j-3)) + " K/s RemNo: " + format(totRemNo/(j-3)) + " K/s
RemYes: " + format(totRemYes/(j-3)) + "K/s");

 System.out.println();

 totAdd = totYes = totNo = totIter = totRemYes = totRemNo = 0;
 }

 for(j = 0; j < 20; j++) {

  m = new OPEN_HASH_BIG_SET(16, f);

  /* We add pairs to m. */
```

```
 ms = System.currentTimeMillis();
 for(i = 0; i < n;  i++) m.add(k[i]);
 d = 1.0 * n / (System.currentTimeMillis() - ms);
 if (j > 2) totAdd += d;
 System.out.print("Add: " + format(d) +" K/s ");

 /* We check for pairs in m. */
 ms = System.currentTimeMillis();
 for(i = 0; i < n;  i++) m.contains(k[i]);
 d = 1.0 * n / (System.currentTimeMillis() - ms);
 if (j > 2) totYes += d;
 System.out.print("Yes: " + format(d) +" K/s ");

 /* We check for pairs not in m. */
 ms = System.currentTimeMillis();
 for(i = 0; i < n;  i++) m.contains(nk[i]);
 d = 1.0 * n / (System.currentTimeMillis() - ms);
 if (j > 2) totNo += d;
 System.out.print("No: " + format(d) +" K/s ");

 /* We iterate on m. */
 ms = System.currentTimeMillis();
 for(KEY_ITERATOR it = (KEY_ITERATOR)m.iterator(); it.hasNext(); it.NEXT_KEY());
 d = 1.0 * n / (System.currentTimeMillis() - ms);
 if (j > 2) totIter += d;
 System.out.print("Iter: " + format(d) +" K/s ");

 /* We delete pairs not in m. */
 ms = System.currentTimeMillis();
 for(i = 0; i < n;  i++) m.remove(nk[i]);
 d = 1.0 * n / (System.currentTimeMillis() - ms);
 if (j > 2) totRemNo += d;
 System.out.print("RemNo: " + format(d) +" K/s ");

 /* We delete pairs in m. */
 ms = System.currentTimeMillis();
 for(i = 0; i < n;  i++) m.remove(k[i]);
 d = 1.0 * n / (System.currentTimeMillis() - ms);
 if (j > 2) totRemYes += d;
 System.out.print("RemYes: " + format(d) +" K/s ");

 System.out.println();
}


 System.out.println();
 System.out.println("fastutil  Add: " + format(totAdd/(j-3)) + " K/s Yes: " + format(totYes/(j-3)) + " K/s No: " +
format(totNo/(j-3)) + " K/s Iter: " + format(totIter/(j-3)) + " K/s RemNo: " + format(totRemNo/(j-3)) + " K/s
```

```
RemYes: " + format(totRemYes/(j-3)) + " K/s");

 System.out.println();
 }


 private static void fatal(String msg) {
  System.out.println(msg);
  System.exit(1);
 }

 private static void ensure(boolean cond, String msg) {
  if (cond) return;
  fatal(msg);
 }


 private static void printProbes(OPEN_HASH_BIG_SET m) {
  long totProbes = 0;
  double totSquareProbes = 0;
  int maxProbes = 0;
  final double f = (double)m.size / m.n;
  for(int i = 0, c = 0; i < m.n; i++) {
   if (! KEY_IS_NULL(BIG_ARRAYS.get(m.key, i))) c++;
   else {
    if (c != 0) {
     final long p = (c + 1) * (c + 2) / 2;
     totProbes += p;
     totSquareProbes += (double)p * p;
    }
    maxProbes = Math.max(c, maxProbes);
    c = 0;
    totProbes++;
    totSquareProbes++;
   }
  }

  final double expected = (double)totProbes / m.n;
  System.err.println("Expected probes: " + (
   3 * Math.sqrt(3) * (f / ((1 - f) * (1 - f))) + 4 / (9 * f) - 1
  ) + "; actual: " + expected + "; stddev: " + Math.sqrt(totSquareProbes / m.n - expected * expected)  + "; max probes:
" + maxProbes);
 }
 private static void runTest(int n, float f) {
  int c;
  OPEN_HASH_BIG_SET m = new OPEN_HASH_BIG_SET(Hash.DEFAULT_INITIAL_SIZE, f);
  java.util.Set t = new java.util.HashSet();
```

```
  /* First of all, we fill t with random data. */

  for(int i=0; i<f * n;  i++) t.add(KEY2OBJ(genKey()));

  /* Now we add to m the same data */

  m.addAll(t);

  if (!m.equals(t)) System.out.println("Error (" + seed + "): !m.equals(t) after insertion");
  if (!t.equals(m)) System.out.println("Error (" + seed + "): !t.equals(m) after insertion");
  printProbes(m);

  /* Now we check that m actually holds that data. */

  for(java.util.Iterator i=t.iterator(); i.hasNext();) {
   Object e = i.next();
   if (!m.contains(e)) {
    System.out.println("Error (" + seed + "): m and t differ on a key ("+e+") after insertion (iterating on t)");
    System.exit(1);
   }
  }

  /* Now we check that m actually holds that data, but iterating on m. */

  c = 0;
  for(java.util.Iterator i=m.iterator(); i.hasNext();) {
   Object e = i.next();
   c++;
   if (!t.contains(e)) {
    System.out.println("Error (" + seed + "): m and t differ on a key ("+e+") after insertion (iterating on m)");
    System.exit(1);
   }
  }

  if (c != t.size()) {
   System.out.println("Error (" + seed + "): m has only " + c + " keys instead of " + t.size() + " after insertion (iterating
on m)");
   System.exit(1);
  }
  /* Now we check that inquiries about random data give the same answer in m and t. For
     m we use the polymorphic method. */

  for(int i=0; i<n;  i++) {
   KEY_TYPE T = genKey();
   if (m.contains(T) != t.contains(KEY2OBJ(T))) {
    System.out.println("Error (" + seed + "): divergence in keys between t and m (polymorphic method)");
    System.exit(1);
   }
```

```
}

/* Again, we check that inquiries about random data give the same answer in m and t, but
   for m we use the standard method. */

for(int i=0; i<n;  i++) {
 KEY_TYPE T = genKey();
 if (m.contains(KEY2OBJ(T)) != t.contains(KEY2OBJ(T))) {
  System.out.println("Error (" + seed + "): divergence between t and m (standard method)");
  System.exit(1);
 }
}


/* Now we put and remove random data in m and t, checking that the result is the same. */

for(int i=0; i<20*n;  i++) {
 KEY_TYPE T = genKey();
 if (m.add(KEY2OBJ(T)) != t.add(KEY2OBJ(T))) {
  System.out.println("Error (" + seed + "): divergence in add() between t and m");
  System.exit(1);
 }
 T = genKey();
 if (m.remove(KEY2OBJ(T)) != t.remove(KEY2OBJ(T))) {
  System.out.println("Error (" + seed + "): divergence in remove() between t and m");
  System.exit(1);
 }
}

if (!m.equals(t)) System.out.println("Error (" + seed + "): !m.equals(t) after removal");
if (!t.equals(m)) System.out.println("Error (" + seed + "): !t.equals(m) after removal");

/* Now we check that m actually holds that data. */

for(java.util.Iterator i=t.iterator(); i.hasNext();) {
 Object e = i.next();
 if (!m.contains(e)) {
  System.out.println("Error (" + seed + "): m and t differ on a key ("+e+") after removal (iterating on t)");
  System.exit(1);
 }
}

/* Now we check that m actually holds that data, but iterating on m. */

for(java.util.Iterator i=m.iterator(); i.hasNext();) {
 Object e = i.next();
 if (!t.contains(e)) {
  System.out.println("Error (" + seed + "): m and t differ on a key ("+e+") after removal (iterating on m)");
```

```
 System.exit(1);
 }
 }

 printProbes(m);

 /* Now we make m into an array, make it again a set and check it is OK. */
 KEY_TYPE a[] = m.TO_KEY_ARRAY();

 if (!new OPEN_HASH_BIG_SET(a).equals(m))
  System.out.println("Error (" + seed + "): toArray() output (or array-based constructor) is not OK");

 /* Now we check cloning. */

 ensure(m.equals(((OPEN_HASH_BIG_SET)m).clone()), "Error (" + seed + "): m does not equal m.clone()");
 ensure(((OPEN_HASH_BIG_SET)m).clone().equals(m), "Error (" + seed + "): m.clone() does not equal m");

 int h = m.hashCode();

 /* Now we save and read m. */

 try {
  java.io.File ff = new java.io.File("it.unimi.dsi.fastutil.test");
  java.io.OutputStream os = new java.io.FileOutputStream(ff);
  java.io.ObjectOutputStream oos = new java.io.ObjectOutputStream(os);

  oos.writeObject(m);
  oos.close();

  java.io.InputStream is = new java.io.FileInputStream(ff);
  java.io.ObjectInputStream ois = new java.io.ObjectInputStream(is);

  m = (OPEN_HASH_BIG_SET)ois.readObject();
  ois.close();
  ff.delete();
 }
 catch(Exception e) {
  e.printStackTrace();
  System.exit(1);
 }

#if !KEY_CLASS_Reference
 if (m.hashCode() != h) System.out.println("Error (" + seed + "): hashCode() changed after save/read");

 printProbes(m);

 /* Now we check that m actually holds that data, but iterating on m. */
```

```
  for(java.util.Iterator i=m.iterator(); i.hasNext();) {
   Object e = i.next();
   if (!t.contains(e)) {
    System.out.println("Error (" + seed + "): m and t differ on a key ("+e+") after save/read");
    System.exit(1);
   }
  }
#else
  m.clear();
  m.addAll(t);
#endif

  /* Now we put and remove random data in m and t, checking that the result is the same. */

  for(int i=0; i<20*n;  i++) {
   KEY_TYPE T = genKey();
   if (m.add(KEY2OBJ(T)) != t.add(KEY2OBJ(T))) {
    System.out.println("Error (" + seed + "): divergence in add() between t and m after save/read");
    System.exit(1);
   }
   T = genKey();
   if (m.remove(KEY2OBJ(T)) != t.remove(KEY2OBJ(T))) {
    System.out.println("Error (" + seed + "): divergence in remove() between t and m after save/read");
    System.exit(1);
   }
  }

  if (!m.equals(t)) System.out.println("Error (" + seed + "): !m.equals(t) after post-save/read removal");
  if (!t.equals(m)) System.out.println("Error (" + seed + "): !t.equals(m) after post-save/read removal");


  /* Now we take out of m everything, and check that it is empty. */

  for(java.util.Iterator i=m.iterator(); i.hasNext();) { i.next(); i.remove();}

  if (!m.isEmpty())  {
   System.out.println("Error (" + seed + "): m is not empty (as it should be)");
   System.exit(1);
  }

#if KEY_CLASS_Integer || KEY_CLASS_Long
  m = new OPEN_HASH_BIG_SET(n, f);
  t.clear();
  int x;

  /* Now we torture-test the hash table. This part is implemented only for integers and longs. */

  int p = m.key.length - 1;
```

```
  for(int i=0; i<p; i++) {
   for (int j=0; j<20; j++) {
    m.add(i+(r.nextInt() % 10)*p);
    m.remove(i+(r.nextInt() % 10)*p);
   }

   for (int j=-10; j<10; j++) m.remove(i+j*p);
  }

  t.addAll(m);

  /* Now all table entries are REMOVED. */

  int k = 0;
  for(int i=0; i<(p*f)/10; i++) {
   for (int j=0; j<10; j++) {
    k++;
    x = i+(r.nextInt() % 10)*p;
    if (m.add(x) != t.add(KEY2OBJ(x)))
     System.out.println("Error (" + seed + "): m and t differ on a key during torture-test insertion.");
   }
  }

  if (!m.equals(t)) System.out.println("Error (" + seed + "): !m.equals(t) after torture-test insertion");
  if (!t.equals(m)) System.out.println("Error (" + seed + "): !t.equals(m) after torture-test insertion");

  for(int i=0; i<(p*f)/10; i++) {
   for (int j=0; j<10; j++) {
    x = i+(r.nextInt() % 10)*p;
    if (m.remove(x) != t.remove(KEY2OBJ(x)))
     System.out.println("Error (" + seed + "): m and t differ on a key during torture-test removal.");
   }
  }

  if (!m.equals(t)) System.out.println("Error (" + seed + "): !m.equals(t) after torture-test removal");
  if (!t.equals(m)) System.out.println("Error (" + seed + "): !t.equals(m) after torture-test removal");

  if (!m.equals(m.clone())) System.out.println("Error (" + seed + "): !m.equals(m.clone()) after torture-test removal");
  if (!((OPEN_HASH_BIG_SET)m.clone()).equals(m)) System.out.println("Error (" + seed + "): !m.clone().equals(m)
after torture-test removal");

  m.trim();

  if (!m.equals(t)) System.out.println("Error (" + seed + "): !m.equals(t) after trim()");
  if (!t.equals(m)) System.out.println("Error (" + seed + "): !t.equals(m) after trim()");

#endif
```

```
 System.out.println("Test OK");
 return;
 }


 public static void main(String args[]) {
  float f = Hash.DEFAULT_LOAD_FACTOR;
  int n  = Integer.parseInt(args[1]);
  if (args.length>2) f = Float.parseFloat(args[2]);
  if (args.length > 3) r = new java.util.Random(seed = Long.parseLong(args[3]));

  try {
   if ("speedTest".equals(args[0]) || "speedComp".equals(args[0])) speedTest(n, f, "speedComp".equals(args[0]));
   else if ("test".equals(args[0])) runTest(n, f);
  } catch(Throwable e) {
   e.printStackTrace(System.err);
   System.err.println("seed: " + seed);
  }
 }

#endif

}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/OpenHashBigSet.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;
```

```
import java.util.Iterator;
import java.util.NoSuchElementException;

/** A class providing static methods and objects that do useful things with type-specific iterators.
*
* @see Iterator
*/

public final class BIG_LIST_ITERATORS {

 private BIG_LIST_ITERATORS() {}

 /** A class returning no elements and a type-specific big list iterator interface.
  *
  * <p>This class may be useful to implement your own in case you subclass
  * a type-specific iterator.
  */

 public static class EmptyBigListIterator KEY_GENERIC implements KEY_BIG_LIST_ITERATOR
KEY_GENERIC, java.io.Serializable, Cloneable {

  private static final long serialVersionUID = -7046029254386353129L;

  protected EmptyBigListIterator() {}

  @Override
  public boolean hasNext() { return false; }

  @Override
  public boolean hasPrevious() { return false; }

  @Override
  public KEY_GENERIC_TYPE NEXT_KEY() { throw new NoSuchElementException(); }

  @Override
  public KEY_GENERIC_TYPE PREV_KEY() { throw new NoSuchElementException(); }

  @Override
  public long nextIndex() { return 0; }

  @Override
  public long previousIndex() { return -1; }

  @Override
  public long skip(long n) { return 0; };

  @Override
```

```
 public long back(long n) { return 0; };

 @Override
 public Object clone() { return EMPTY_BIG_LIST_ITERATOR; }

 private Object readResolve() { return EMPTY_BIG_LIST_ITERATOR; }
}

/** An empty iterator (immutable). It is serializable and cloneable.
 *
 * <p>The class of this objects represent an abstract empty iterator
 * that can iterate as a type-specific (list) iterator.
 */

SUPPRESS_WARNINGS_KEY_RAWTYPES
public static final EmptyBigListIterator EMPTY_BIG_LIST_ITERATOR = new EmptyBigListIterator();


/** An iterator returning a single element. */

private static class SingletonBigListIterator KEY_GENERIC implements KEY_BIG_LIST_ITERATOR
KEY_GENERIC {
 private final KEY_GENERIC_TYPE element;
 private int curr;

 public SingletonBigListIterator(final KEY_GENERIC_TYPE element) {
  this.element = element;
 }

 @Override
 public boolean hasNext() { return curr == 0; }

 @Override
 public boolean hasPrevious() { return curr == 1; }

 @Override
 public KEY_GENERIC_TYPE NEXT_KEY() {
  if (! hasNext()) throw new NoSuchElementException();
  curr = 1;
  return element;
 }

 @Override
 public KEY_GENERIC_TYPE PREV_KEY() {
  if (! hasPrevious()) throw new NoSuchElementException();
  curr = 0;
  return element;
 }
```

```
 @Override
 public long nextIndex() {
  return curr;
 }

 @Override
 public long previousIndex() {
  return curr - 1;
 }
}


/** Returns an iterator that iterates just over the given element.
 *
 * @param element the only element to be returned by a type-specific list iterator.
 * @return  an iterator that iterates just over {@code element}.
 */
 public static KEY_GENERIC KEY_BIG_LIST_ITERATOR KEY_GENERIC singleton(final
KEY_GENERIC_TYPE element) {
 return new SingletonBigListIterator KEY_GENERIC_DIAMOND(element);
 }


 /** An unmodifiable wrapper class for big list iterators. */

 public static class UnmodifiableBigListIterator KEY_GENERIC implements KEY_BIG_LIST_ITERATOR
KEY_GENERIC {
 protected final KEY_BIG_LIST_ITERATOR KEY_GENERIC i;

 public UnmodifiableBigListIterator(final KEY_BIG_LIST_ITERATOR KEY_GENERIC i) {
  this.i = i;
 }

 @Override
 public boolean hasNext() { return i.hasNext(); }

 @Override
 public boolean hasPrevious() { return i.hasPrevious(); }

 @Override
 public KEY_GENERIC_TYPE NEXT_KEY() { return i.NEXT_KEY(); }

 @Override
 public KEY_GENERIC_TYPE PREV_KEY() { return i.PREV_KEY(); }

 @Override
 public long nextIndex() { return i.nextIndex(); }
```

```
 @Override
 public long previousIndex() { return i.previousIndex(); }
 }

 /** Returns an unmodifiable list iterator backed by the specified list iterator.
  *
  * @param i the list iterator to be wrapped in an unmodifiable list iterator.
  * @return an unmodifiable view of the specified list iterator.
  */
 public static KEY_GENERIC KEY_BIG_LIST_ITERATOR KEY_GENERIC unmodifiable(final
KEY_BIG_LIST_ITERATOR KEY_GENERIC i) { return new UnmodifiableBigListIterator
KEY_GENERIC_DIAMOND(i); }


 /** A class exposing a list iterator as a big-list iterator.. */

 public static class BigListIteratorListIterator KEY_GENERIC implements KEY_BIG_LIST_ITERATOR
KEY_GENERIC {
  protected final KEY_LIST_ITERATOR KEY_GENERIC i;

  protected BigListIteratorListIterator(final KEY_LIST_ITERATOR KEY_GENERIC i) {
   this.i = i;
  }

  private int intDisplacement(long n) {
   if (n < Integer.MIN_VALUE || n > Integer.MAX_VALUE) throw new IndexOutOfBoundsException("This big
iterator is restricted to 32-bit displacements");
   return (int)n;
  }

  @Override
  public void set(KEY_GENERIC_TYPE ok) { i.set(ok); }

  @Override
  public void add(KEY_GENERIC_TYPE ok) { i.add(ok); }

  @Override
  public int back(int n) { return i.back(n); }

  @Override
  public long back(long n) { return i.back(intDisplacement(n)); }

  @Override
  public void remove() { i.remove(); }

  @Override
  public int skip(int n) { return i.skip(n); }
```

```
 @Override
 public long skip(long n) { return i.skip(intDisplacement(n)); }

 @Override
 public boolean hasNext() { return i.hasNext(); }

 @Override
 public boolean hasPrevious() { return i.hasPrevious(); }

 @Override
 public KEY_GENERIC_TYPE NEXT_KEY() { return i.NEXT_KEY(); }

 @Override
 public KEY_GENERIC_TYPE PREV_KEY() { return i.PREV_KEY(); }

 @Override
 public long nextIndex() { return i.nextIndex(); }

 @Override
 public long previousIndex() { return i.previousIndex(); }
}

 /** Returns a big-list iterator backed by the specified list iterator.
  *
  * @param i the list iterator to adapted to the big-list-iterator interface.
  * @return a big-list iterator backed by the specified list iterator.
  */
 public static KEY_GENERIC KEY_BIG_LIST_ITERATOR KEY_GENERIC asBigListIterator(final
KEY_LIST_ITERATOR KEY_GENERIC i) { return new BigListIteratorListIterator
KEY_GENERIC_DIAMOND(i); }
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/BigListIterators.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
```

```
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

/**  An abstract class providing basic methods for priority queues implementing a type-specific interface.
* @deprecated As of fastutil 8 this class is no longer necessary, as its previous abstract
* methods are now default methods of the type-specific interface.
*/
@Deprecated
public abstract class ABSTRACT_PRIORITY_QUEUE KEY_GENERIC extends
it.unimi.dsi.fastutil.AbstractPriorityQueue<KEY_GENERIC_CLASS> implements java.io.Serializable,
PRIORITY_QUEUE KEY_GENERIC  {
 private static final long serialVersionUID = 1L;
}

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-
a775574/drv/AbstractPriorityQueue.drv
No license file was found, but licenses were detected in source scan.

/*
* Copyright (C) 2003-2017 Paolo Boldi and Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

#if KEY_CLASS_Object
import java.util.Comparator;
#endif
```

```
import java.util.Arrays;

/** A class providing static methods and objects that do useful things with indirect heaps.
*
* <p>An indirect heap is an extension of a semi-indirect heap using also an
* <em>inversion array</em> of the same length as the reference array,
* satisfying the relation {@code heap[inv[i]]==i} when
* {@code inv[i]&gt;=0}, and {@code inv[heap[i]]==i} for all elements in the heap.
*/

public final class INDIRECT_HEAPS {

 private INDIRECT_HEAPS() {}

 /** Moves the given element down into the indirect heap until it reaches the lowest possible position.
  *
  * @param refArray the reference array.
  * @param heap the indirect heap (starting at 0).
  * @param inv the inversion array.
  * @param size the number of elements in the heap.
  * @param i the index in the heap of the element to be moved down.
  * @param c a type-specific comparator, or {@code null} for the natural order.
  * @return the new position in the heap of the element of heap index {@code i}.
  */

 SUPPRESS_WARNINGS_KEY_UNCHECKED
 public static KEY_GENERIC int downHeap(final KEY_GENERIC_TYPE[] refArray, final int[] heap, final int[]
inv, final int size, int i, final KEY_COMPARATOR KEY_GENERIC c) {
  assert i < size;

  final int e = heap[i];
  final KEY_GENERIC_TYPE E = refArray[e];
  int child;

  if (c == null)
   while ((child = (i << 1) + 1) < size) {
    int t = heap[child];
    final int right = child + 1;
    if (right < size && KEY_LESS(refArray[heap[right]], refArray[t])) t = heap[child = right];
    if (KEY_LESSEQ(E, refArray[t])) break;
    heap[i] = t;
    inv[heap[i]] = i;
    i = child;
   }
  else
   while ((child = (i << 1) + 1) < size) {
    int t = heap[child];
    final int right = child + 1;
```

```
  if (right < size && c.compare(refArray[heap[right]], refArray[t]) < 0) t = heap[child = right];
  if (c.compare(E, refArray[t]) <= 0) break;
  heap[i] = t;
  inv[heap[i]] = i;
  i = child;
  }

 heap[i] = e;
 inv[e] = i;
 return i;
}

/** Moves the given element up in the indirect heap until it reaches the highest possible position.
 *
 * Note that in principle after this call the heap property may be violated.
 *
 * @param refArray the reference array.
 * @param heap the indirect heap (starting at 0).
 * @param inv the inversion array.
 * @param size the number of elements in the heap.
 * @param i the index in the heap of the element to be moved up.
 * @param c a type-specific comparator, or {@code null} for the natural order.
 * @return the new position in the heap of the element of heap index {@code i}.
 */

SUPPRESS_WARNINGS_KEY_UNCHECKED
public static KEY_GENERIC int upHeap(final KEY_GENERIC_TYPE[] refArray, final int[] heap, final int[] inv,
final int size, int i, final KEY_COMPARATOR KEY_GENERIC c) {
 assert i < size;

 final int e = heap[i];
 final KEY_GENERIC_TYPE E = refArray[e];

 if (c == null)
  while (i != 0) {
   final int parent = (i - 1) >>> 1;
   final int t = heap[parent];
   if (KEY_LESSEQ(refArray[t], E)) break;
   heap[i] = t;
   inv[heap[i]] = i;
   i = parent;
  }
 else
  while (i != 0) {
   final int parent = (i - 1) >>> 1;
   final int t = heap[parent];
   if (c.compare(refArray[t], E) <= 0) break;
   heap[i] = t;
```

```
  inv[heap[i]] = i;
  i = parent;
  }

 heap[i] = e;
 inv[e] = i;

 return i;
}

/** Creates an indirect heap in the given array.
 *
 * @param refArray the reference array.
 * @param offset the first element of the reference array to be put in the heap.
 * @param length the number of elements to be put in the heap.
 * @param heap the array where the heap is to be created.
 * @param inv the inversion array.
 * @param c a type-specific comparator, or {@code null} for the natural order.
 */

public static KEY_GENERIC void makeHeap(final KEY_GENERIC_TYPE[] refArray, final int offset, final int
length, final int[] heap, final int[] inv, final KEY_COMPARATOR KEY_GENERIC c) {
 ARRAYS.ensureOffsetLength(refArray, offset, length);
 if (heap.length < length) throw new IllegalArgumentException("The heap length (" + heap.length + ") is smaller
than the number of elements (" + length + ")");
 if (inv.length < refArray.length) throw new IllegalArgumentException("The inversion array length (" + heap.length
+ ") is smaller than the length of the reference array (" + refArray.length + ")");

 Arrays.fill(inv, 0, refArray.length, -1);

 int i = length;
 while(i-- != 0) inv[heap[i] = offset + i] = i;

 i = length >>> 1;
 while(i-- != 0) downHeap(refArray, heap, inv, length, i, c);
}


/** Creates an indirect heap from a given index array.
 *
 * @param refArray the reference array.
 * @param heap an array containing indices into {@code refArray}.
 * @param inv the inversion array.
 * @param size the number of elements in the heap.
 * @param c a type-specific comparator, or {@code null} for the natural order.
 */

public static KEY_GENERIC void makeHeap(final KEY_GENERIC_TYPE[] refArray, final int[] heap, final int[]
```

```
inv, final int size, final KEY_COMPARATOR KEY_GENERIC c) {
  int i = size >>> 1;
  while(i-- != 0) downHeap(refArray, heap, inv, size, i, c);
 }
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/IndirectHeaps.drv
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import java.util.SortedSet;
import java.util.Collection;

/** A type-specific {@link SortedSet}; provides some additional methods that use polymorphism to avoid (un)boxing.
*
* <p>Additionally, this interface strengthens {@link #iterator()},
* {@link #comparator()} (for primitive types), {@link SortedSet#subSet(Object,Object)},
* {@link SortedSet#headSet(Object)} and {@link SortedSet#tailSet(Object)}.
*
* @see SortedSet
*/


public interface SORTED_SET KEY_GENERIC extends SET KEY_GENERIC,
SortedSet<KEY_GENERIC_CLASS>, KEY_BIDI_ITERABLE KEY_GENERIC {

 /** Returns a type-specific {@link it.unimi.dsi.fastutil.BidirectionalIterator} on the elements in
```

```
 * this set, starting from a given element of the domain (optional operation).
 *
 * <p>This method returns a type-specific bidirectional iterator with given
 * starting point. The starting point is any element comparable to the
 * elements of this set (even if it does not actually belong to the
 * set). The next element of the returned iterator is the least element of
 * the set that is greater than the starting point (if there are no
 * elements greater than the starting point, {@link
 * it.unimi.dsi.fastutil.BidirectionalIterator#hasNext() hasNext()} will return
 * {@code false}). The previous element of the returned iterator is
 * the greatest element of the set that is smaller than or equal to the
 * starting point (if there are no elements smaller than or equal to the
 * starting point, {@link it.unimi.dsi.fastutil.BidirectionalIterator#hasPrevious()
 * hasPrevious()} will return {@code false}).
 *
 * <p>Note that passing the last element of the set as starting point and
 * calling {@link it.unimi.dsi.fastutil.BidirectionalIterator#previous() previous()} you can traverse the
 * entire set in reverse order.
 *
 * @param fromElement an element to start from.
 * @return a bidirectional iterator on the element in this set, starting at the given element.
 * @throws UnsupportedOperationException if this set does not support iterators with a starting point.
 */

KEY_BIDI_ITERATOR KEY_GENERIC iterator(KEY_GENERIC_TYPE fromElement);

/** Returns a type-specific {@link it.unimi.dsi.fastutil.BidirectionalIterator} on the elements in
 * this set.
 *
 * <p>Note that this specification strengthens the one given in the corresponding type-specific
 * {@link Collection}.
 *
 * <p>This method returns a parameterised bidirectional iterator. The iterator
 * can be moreover safely cast to a type-specific iterator.
 *
 * @return a bidirectional iterator on the element in this set.
 */

@Override
KEY_BIDI_ITERATOR KEY_GENERIC iterator();

/** Returns a view of the portion of this sorted set whose elements range from {@code fromElement}, inclusive, to
{@code toElement}, exclusive.
 * <p>Note that this specification strengthens the one given in {@link SortedSet#subSet(Object,Object)}.
 * @see SortedSet#subSet(Object,Object)
 */

#if KEYS_REFERENCE
```

```
 @Override
#endif
 SORTED_SET KEY_GENERIC subSet(KEY_GENERIC_TYPE fromElement, KEY_GENERIC_TYPE
toElement) ;

 /** Returns a view of the portion of this sorted set whose elements are strictly less than {@code toElement}.
  * <p>Note that this specification strengthens the one given in {@link SortedSet#headSet(Object)}.
  * @see SortedSet#headSet(Object)
  */

#if KEYS_REFERENCE
 @Override
#endif
 SORTED_SET KEY_GENERIC headSet(KEY_GENERIC_TYPE toElement);

 /** Returns a view of the portion of this sorted set whose elements are greater than or equal to {@code
fromElement}.
  * <p>Note that this specification strengthens the one given in {@link SortedSet#headSet(Object)}.
  * @see SortedSet#tailSet(Object)
  */

#if KEYS_REFERENCE
 @Override
#endif
 SORTED_SET KEY_GENERIC tailSet(KEY_GENERIC_TYPE fromElement);

#if KEYS_PRIMITIVE
 /** {@inheritDoc}
  * <p>Note that this specification strengthens the one given in {@link SortedSet#comparator()}.
  */
 @Override
 KEY_COMPARATOR comparator();

 /** Returns the first (lowest) element currently in this set.
  * @see SortedSet#first()
  */
 KEY_TYPE FIRST();

 /** Returns the last (highest) element currently in this set.
  * @see SortedSet#last()
  */
 KEY_TYPE LAST();

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead.
  */
 @Deprecated
 @Override
```

```
 default SORTED_SET KEY_GENERIC subSet(final KEY_GENERIC_CLASS from, final
KEY_GENERIC_CLASS to) {
  return subSet(from.KEY_VALUE(), to.KEY_VALUE());
 }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead.
  */
 @Deprecated
 @Override
 default SORTED_SET KEY_GENERIC headSet(final KEY_GENERIC_CLASS to) {
  return headSet(to.KEY_VALUE());
 }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead.
  */
 @Deprecated
 @Override
 default SORTED_SET KEY_GENERIC tailSet(final KEY_GENERIC_CLASS from) {
  return tailSet(from.KEY_VALUE());
 }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead.
  */
 @Deprecated
 @Override
 default KEY_GENERIC_CLASS first() {
  return KEY2OBJ(FIRST());
 }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead.
  */
 @Deprecated
 @Override
 default KEY_CLASS last() {
  return KEY2OBJ(LAST());
 }

#endif

}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/SortedSet.drv

No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (C) 2002-2017 Sebastiano Vigna
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

import it.unimi.dsi.fastutil.Function;

/** A type-specific {@link Function}; provides some additional methods that use polymorphism to avoid (un)boxing.
*
* <p>Type-specific versions of {@code get()}, {@code put()} and
* {@code remove()} cannot rely on {@code null} to denote absence of
* a key. Rather, they return a {@linkplain #defaultReturnValue() default
* return value}, which is set to 0/false at creation, but can be changed using
* the {@code defaultReturnValue()} method.
*
* <p>For uniformity reasons, even functions returning objects implement the default
* return value (of course, in this case the default return value is
* initialized to {@code null}).
*
* <p>The default implementation of optional operations just throw an {@link
* UnsupportedOperationException}, except for the type-specific {@code
* containsKey()}, which return true. Generic versions of accessors delegate to
* the corresponding type-specific counterparts following the interface rules.
*
* <p><strong>Warning:</strong> to fall in line as much as possible with the
* {@linkplain java.util.Map standard map interface}, it is required that
* standard versions of {@code get()}, {@code put()} and
* {@code remove()} for maps with primitive-type keys or values <em>return
* {@code null} to denote missing keys </em> rather than wrap the default
* return value in an object. In case both keys and values are reference
* types, the default return value must be returned instead, thus violating
```

```
* the {@linkplain java.util.Map standard map interface} when the default
* return value is not {@code null}.
*
* @see Function
*/

@FunctionalInterface
#ifdef JDK_PRIMITIVE_FUNCTION
public interface FUNCTION KEY_VALUE_GENERIC extends Function<KEY_GENERIC_CLASS,
VALUE_GENERIC_CLASS>, JDK_PRIMITIVE_FUNCTION KEY_GENERIC VALUE_GENERIC {
#else
public interface FUNCTION KEY_VALUE_GENERIC extends Function<KEY_GENERIC_CLASS,
VALUE_GENERIC_CLASS> {
#endif

#ifdef JDK_PRIMITIVE_FUNCTION
#if KEY_WIDENED
 /** {@inheritDoc}
  *
  * <p>In this default implementation, the key gets narrowed down to the
  * actual key type, throwing an exception if the given key can't be
  * represented in the restricted domain. This is done for interoperability
  * with the Java 8 function environment. Its use is discouraged, as
  * unexpected errors can occur. Instead, the corresponding classes should be
  * used (e.g., {@link it.unimi.dsi.fastutil.ints.Int2IntFunction} instead of
  * {@link it.unimi.dsi.fastutil.shorts.Short2IntFunction}).
  *
  * @throws IllegalArgumentException If the given operand is not an element of the key domain.
  * @since 8.0.0
  * @deprecated Please use primitive types which don't have to be widened as keys.
  */
 @Deprecated
#else
 /** {@inheritDoc}
  * @since 8.0.0
  */
#endif
 @Override
 default VALUE_GENERIC_TYPE_WIDENED
JDK_PRIMITIVE_FUNCTION_APPLY(KEY_GENERIC_TYPE_WIDENED operand) { return
GET_VALUE(KEY_NARROWING(operand)); }
#endif

 /** Adds a pair to the map (optional operation).
  *
  * @param key the key.
  * @param value the value.
  * @return the old value, or the {@linkplain #defaultReturnValue() default return value} if no value was present for
```

```
the given key.
  * @see Function#put(Object,Object)
  */

 default VALUE_GENERIC_TYPE put(final KEY_GENERIC_TYPE key, final VALUE_GENERIC_TYPE value)
{
  throw new UnsupportedOperationException();
 }

 /** Returns the value to which the given key is mapped.
  *
  * @param key the key.
  * @return the corresponding value, or the {@linkplain #defaultReturnValue() default return value} if no value was
present for the given key.
  * @see Function#get(Object)
  */

 VALUE_GENERIC_TYPE GET_VALUE(KEY_TYPE key);

 /** Removes the mapping with the given key (optional operation).
  * @param key the key.
  * @return the old value, or the {@linkplain #defaultReturnValue() default return value} if no value was present for
the given key.
  * @see Function#remove(Object)
  */

 default VALUE_GENERIC_TYPE REMOVE_VALUE(final KEY_TYPE key) {
  throw new UnsupportedOperationException();
 }

#if KEYS_PRIMITIVE || VALUES_PRIMITIVE

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 default VALUE_GENERIC_CLASS put(final KEY_GENERIC_CLASS key, final VALUE_GENERIC_CLASS
value) {
  final KEY_GENERIC_TYPE k = KEY_CLASS2TYPE(key);
  final boolean containsKey = containsKey(k);
  final VALUE_GENERIC_TYPE v = put(k, VALUE_CLASS2TYPE(value));
  return containsKey ? VALUE2OBJ(v) : null;
 }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
```

```
 default VALUE_GENERIC_CLASS get(final Object key) {
#if KEYS_PRIMITIVE
  if (key == null) return null;
#endif
  final KEY_TYPE k = KEY_OBJ2TYPE(key);
  final VALUE_GENERIC_TYPE v = GET_VALUE(k);
  return (v != defaultReturnValue() || containsKey(k)) ? VALUE2OBJ(v) : null;
 }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */
 @Deprecated
 @Override
 default VALUE_GENERIC_CLASS remove(final Object key) {
#if KEYS_PRIMITIVE
  if (key == null) return null;
#endif
  final KEY_TYPE k = KEY_OBJ2TYPE(key);
  return containsKey(k) ? VALUE2OBJ(REMOVE_VALUE(k)) : null;
 }

#if KEYS_PRIMITIVE

 /** Returns true if this function contains a mapping for the specified key.
  *
  * <p>Note that for some kind of functions (e.g., hashes) this method will
  * always return true. In particular, this default implementation always
  * returns true.
  *
  * @param key the key.
  * @return true if this function associates a value to {@code key}.
  * @see Function#containsKey(Object)
  */
 default boolean containsKey(KEY_TYPE key) {
  return true;
 }

 /** {@inheritDoc}
  * @deprecated Please use the corresponding type-specific method instead. */

 @Deprecated
 @Override
 default boolean containsKey(final Object key) {
  return key == null ? false : containsKey(KEY_OBJ2TYPE(key));
 }

#endif
```

```
#endif

 /** Sets the default return value (optional operation).
  *
  * This value must be returned by type-specific versions of
  * {@code get()}, {@code put()} and {@code remove()} to
  * denote that the map does not contain the specified key. It must be
  * 0/{@code false}/{@code null} by default.
  *
  * @param rv the new default return value.
  * @see #defaultReturnValue()
  */

 default void defaultReturnValue(VALUE_GENERIC_TYPE rv) {
  throw new UnsupportedOperationException();
 }


 /** Gets the default return value.
  *
  * <p>This default implementation just return the default null value
  * of the type ({@code null} for objects, 0 for scalars, false for Booleans).
  *
  * @return the current default return value.
  */

 default VALUE_GENERIC_TYPE defaultReturnValue() {
  return VALUE_NULL;
 }
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/Function.drv
No license file was found, but licenses were detected in source scan.

```
* See the License for the specific language governing permissions and
* limitations under the License.
*/


package PACKAGE;

/**  An abstract class facilitating the creation of type-specific {@linkplain java.util.Comparator comparators}.
*
* @see java.util.Comparator
* @deprecated As of fastutil 8 this class is no longer necessary, as its only previous abstract
* method is now a default method of the type-specific interface.
*/

@Deprecated
public abstract class KEY_ABSTRACT_COMPARATOR KEY_GENERIC implements KEY_COMPARATOR
KEY_GENERIC, java.io.Serializable {
 private static final long serialVersionUID = 0L;
 protected KEY_ABSTRACT_COMPARATOR() {}
}
```

Found in path(s):
* /opt/cola/permits/1473460152_1668478175.1613455/0/vigna-fastutil-8-2-3-0-ga775574-1-tar-gz/vigna-fastutil-a775574/drv/AbstractComparator.drv

1.105 jackson-dataformat-smile 2.14.0

1.105.1 Available under license :

No license file was found, but licenses were detected in source scan.

```
#  Copyright 2012 FasterXML.com
#
#  Licensed under the Apache License, Version 2.0 (the "License");
#  you may not use this file except in compliance with the License.
#  You may obtain a copy of the License at
#
#  http://www.apache.org/licenses/LICENSE-2.0
#
#  Unless required by applicable law or agreed to in writing, software
#  distributed under the License is distributed on an "AS IS" BASIS,
#  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
#  See the License for the specific language governing permissions and
#  limitations under the License.

com.fasterxml.jackson.dataformat.smile.SmileFactory
```

Found in path(s):

* /opt/cola/permits/1473561857_1668493070.3545423/0/jackson-dataformat-smile-2-14-0-sources-jar/META-INF/services/com.fasterxml.jackson.core.JsonFactory

1.106 aop-alliance 1.0

1.106.1 Available under license :

all the source code provided by AOP Alliance is Public Domain.

1.107 metrics---dropwizard v4.0.5

1.107.1 Available under license :

No license file was found, but licenses were detected in source scan.

Manifest-Version: 1.0
Bnd-LastModified: 1545937975057
Build-Jdk: 1.8.0_191
Built-By: artem
Bundle-Description: An Apache HttpClient wrapper providing Metrics ins
trumentation of connection pools, request durations and rates,
and other useful information.
Bundle-License: http://www.apache.org/licenses/LICENSE-2.0.html
Bundle-ManifestVersion: 2
Bundle-Name: Metrics Integration for Apache HttpClient
Bundle-SymbolicName: io.dropwizard.metrics.httpclient
Bundle-Version: 4.0.5
Created-By: Apache Maven Bundle Plugin
Export-Package: com.codahale.metrics.httpclient;uses:="com.codahale.me
trics,org.apache.http,org.apache.http.config,org.apache.http.conn,org
.apache.http.conn.routing,org.apache.http.conn.socket,org.apache.http
.impl.client,org.apache.http.impl.conn,org.apache.http.protocol";vers
ion="4.0.5"
Implementation-Title: Metrics Integration for Apache HttpClient
Implementation-URL: http://metrics.dropwizard.io/metrics-httpclient
Implementation-Vendor-Id: io.dropwizard.metrics
Implementation-Version: 4.0.5
Import-Package: com.codahale.metrics;version="[4.0,5)",org.apache.http
,org.apache.http.client,org.apache.http.client.methods,org.apache.htt
p.client.utils,org.apache.http.config,org.apache.http.conn,org.apache
.http.conn.routing,org.apache.http.conn.socket,org.apache.http.conn.s
sl,org.apache.http.impl.client,org.apache.http.impl.conn,org.apache.h
ttp.pool,org.apache.http.protocol
Require-Capability: osgi.ee;filter:="(&(osgi.ee=JavaSE)(version=1.8))"
Tool: Bnd-3.3.0.201609221906

Found in path(s):
* /opt/cola/permits/1274705442_1648835835.08/0/metrics-httpclient-4-0-5-jar/META-INF/MANIFEST.MF

1.108 idna 3.3

1.108.1 Available under license :

BSD 3-Clause License

Copyright (c) 2013-2021, Kim Davies
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its
  contributors may be used to endorse or promote products derived from
  this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

1.109 java-architecture-for-xml-binding 2.3.3

1.109.1 Available under license :

(See license.txt for the actual license terms)

Copyright 2001-@@YEAR@@ Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto,
California, 94303, U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to
technology embodied in this product. In particular, and without
limitation, these intellectual property rights may include one or more
of the U.S. patents listed at http://www.sun.com/patents and one or

more additional patents or pending patent applications in the U.S. and other countries. This product is distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third party software, including font technology, is copyrighted and licensed from Sun suppliers. Sun, the Sun logo, and Sun Microsystems are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. This product includes software developed by the Apache Software Foundation (http://www.apache.org/). Federal Acquisitions: Commercial Software - Government Users Subject to Standard License Terms and Conditions.

Copyright 2001--@@YEAR@@ Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, Californie 94303 tats-Unis. Tous droits rservs. Distribue par des licences qui en restreignent l'utilisation. Sun Microsystems, Inc. a les droits de proprit intellectuels relatants la technologie incorpore dans ce produit. En particulier, et sans la limitation, ces droits de proprit intellectuels peuvent inclure un ou plus des brevets amricains numrs http://www.sun.com/patents et un ou les brevets plus supplmentaires ou les applications de brevet en attente dans les Etats Unis et les autres pays. Ce produit ou document est protg par un copyright et distribu avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la dcompilation. Aucune partie de ce produit ou document ne peut tre reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation pralable et crite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel dtenu par des tiers, et qui comprend la technologie relative aux polices de caractres, est protg par un copyright et licenci par des fournisseurs de Sun. Sun, le logo Sun, Sun Microsystems et sont des marques de fabrique ou des marques d?pos?es de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Ce produit inclut le logiciel dvelopp par la base de Apache Software Foundation (http://www.apache.org/). L'accord du gouvernement des tats Unis est requis avant l'exportation du produit.

```
/*
* The Apache Software License, Version 1.1
*
*
* Copyright (c) 1999-2004 The Apache Software Foundation.  All rights
* reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
*
```

* 1. Redistributions of source code must retain the above copyright
*   notice, this list of conditions and the following disclaimer.
*
* 2. Redistributions in binary form must reproduce the above copyright
*   notice, this list of conditions and the following disclaimer in
*   the documentation and/or other materials provided with the
*   distribution.
*
* 3. The end-user documentation included with the redistribution,
*   if any, must include the following acknowledgment:
*   "This product includes software developed by the
*   Apache Software Foundation (http://www.apache.org/)."
*   Alternately, this acknowledgment may appear in the software itself,
*   if and wherever such third-party acknowledgments normally appear.
*
* 4. The names "Xerces" and "Apache Software Foundation" must
*   not be used to endorse or promote products derived from this
*   software without prior written permission. For written
*   permission, please contact apache@apache.org.
*
* 5. Products derived from this software may not be called "Apache",
*   nor may "Apache" appear in their name, without prior written
*   permission of the Apache Software Foundation.
*
* THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED
* WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
* OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
* DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR
* ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
* LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
* USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
* ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
* OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
* OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
* ==
*
* This software consists of voluntary contributions made by many
* individuals on behalf of the Apache Software Foundation and was
* originally based on software copyright (c) 1999, International
* Business Machines, Inc., http://www.apache.org. For more
* information on the Apache Software Foundation, please see
* <http://www.apache.org/>.
*/
THIS LICENSE IS INTENDED TO BE USED FOR DEBUGGING THE INSTALLER.

Amendment I

Congress shall make no law respecting an establishment of religion, or prohibiting the free exercise thereof; or abridging the freedom of speech, or of the press; or the right of the people peaceably to assemble, and to petition the government for a redress of grievances.

Amendment II

A well regulated militia, being necessary to the security of a free state, the right of the people to keep and bear arms, shall not be infringed.

Amendment III

No soldier shall, in time of peace be quartered in any house, without the consent of the owner, nor in time of war, but in a manner to be prescribed by law.

Amendment IV

The right of the people to be secure in their persons, houses, papers, and effects, against unreasonable searches and seizures, shall not be violated, and no warrants shall issue, but upon probable cause, supported by oath or affirmation, and particularly describing the place to be searched, and the persons or things to be seized.

Amendment V

No person shall be held to answer for a capital, or otherwise infamous crime, unless on a presentment or indictment of a grand jury, except in cases arising in the land or naval forces, or in the militia, when in actual service in time of war or public danger; nor shall any person be subject for the same offense to be twice put in jeopardy of life or limb; nor shall be compelled in any criminal case to be a witness against himself, nor be deprived of life, liberty, or property, without due process of law; nor shall private property be taken for public use, without just compensation.

Amendment VI

In all criminal prosecutions, the accused shall enjoy the right to a speedy and public trial, by an impartial jury of the state and district wherein the crime shall have been committed, which district shall have been previously ascertained by law, and to be informed of the nature and cause of the accusation; to be confronted with the witnesses against him; to have compulsory process for obtaining witnesses in his favor, and to have the assistance of counsel for his defense.

Amendment VII

In suits at common law, where the value in controversy shall exceed twenty dollars, the right of trial by jury shall be preserved, and no fact tried by a jury, shall be otherwise reexamined in any court of the United States, than according to the rules of the common law.

Amendment VIII

Excessive bail shall not be required, nor excessive fines imposed, nor cruel and unusual punishments inflicted.

Amendment IX

The enumeration in the Constitution, of certain rights, shall not be construed to deny or disparage others retained by the people.

Amendment X

The powers not delegated to the United States by the Constitution, nor prohibited by it to the states, are reserved to the states respectively, or to the people.
(See license.txt for the actual license terms)

---

protg par un copyright et licenci par des fournisseurs de Sun. Sun,
le logo Sun, Sun Microsystems et sont des marques de fabrique ou des
marques d?pos?es de Sun Microsystems, Inc. aux Etats-Unis et dans
d'autres pays. Ce produit inclut le logiciel dvelopp par la base de
Apache Software Foundation (http://www.apache.org/). L'accord du
gouvernement des tats Unis est requis avant l'exportation du produit.

```
/* ====================================================================
* The Apache Software License, Version 1.1
*
* Copyright (c) 2001-2003 The Apache Software Foundation.  All rights
* reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
*
* 1. Redistributions of source code must retain the above copyright
*    notice, this list of conditions and the following disclaimer.
*
* 2. Redistributions in binary form must reproduce the above copyright
*    notice, this list of conditions and the following disclaimer in
*    the documentation and/or other materials provided with the
*    distribution.
*
* 3. The end-user documentation included with the redistribution,
*    if any, must include the following acknowledgment:
*       "This product includes software developed by the
*        Apache Software Foundation (http://www.apache.org/)."
*    Alternately, this acknowledgment may appear in the software itself,
*    if and wherever such third-party acknowledgments normally appear.
*
* 4. The names "Apache" and "Apache Software Foundation" must
*    not be used to endorse or promote products derived from this
*    software without prior written permission. For written
*    permission, please contact apache@apache.org.
*
* 5. Products derived from this software may not be called "Apache",
*    nor may "Apache" appear in their name, without prior written
*    permission of the Apache Software Foundation.
*
* THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED
* WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
* OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
* DISCLAIMED.  IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR
* ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
* LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
* USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
```

```
* ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
* OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
* OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
* ====================================================================
*
* This software consists of voluntary contributions made by many
* individuals on behalf of the Apache Software Foundation.  For more
* information on the Apache Software Foundation, please see
* <http://www.apache.org/>.
*/
/* ====================================================================
* The Apache Software License, Version 1.1
*
* Copyright (c) 2000 The Apache Software Foundation.  All rights
* reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
*
* 1. Redistributions of source code must retain the above copyright
*    notice, this list of conditions and the following disclaimer.
*
* 2. Redistributions in binary form must reproduce the above copyright
*    notice, this list of conditions and the following disclaimer in
*    the documentation and/or other materials provided with the
*    distribution.
*
* 3. The end-user documentation included with the redistribution,
*    if any, must include the following acknowledgment:
*       "This product includes software developed by the
*        Apache Software Foundation (http://www.apache.org/)."
*    Alternately, this acknowledgment may appear in the software itself,
*    if and wherever such third-party acknowledgments normally appear.
*
* 4. The names "Apache" and "Apache Software Foundation" must
*    not be used to endorse or promote products derived from this
*    software without prior written permission. For written
*    permission, please contact apache@apache.org.
*
* 5. Products derived from this software may not be called "Apache",
*    nor may "Apache" appear in their name, without prior written
*    permission of the Apache Software Foundation.
*
* THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED
* WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
* OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
```

* DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR
* ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
* LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
* USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
* ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
* OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
* OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
* ==
*
* This software consists of voluntary contributions made by many
* individuals on behalf of the Apache Software Foundation. For more
* information on the Apache Software Foundation, please see
* <http://www.apache.org/>.
*
* Portions of this software are based upon public domain software
* originally written at the National Center for Supercomputing Applications,
* University of Illinois, Urbana-Champaign.
*/

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made,

use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions

for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability

incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following
boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL) Version 1.0

1. Definitions.

1.1. "Contributor" means each individual or entity that
creates or contributes to the creation of Modifications.

1.2. "Contributor Version" means the combination of the
Original Software, prior Modifications used by a
Contributor (if any), and the Modifications made by that
particular Contributor.

1.3. "Covered Software" means (a) the Original Software, or
(b) Modifications, or (c) the combination of files
containing Original Software with files containing
Modifications, in each case including portions thereof.

1.4. "Executable" means the Covered Software in any form

other than Source Code.

1.5. "Initial Developer" means the individual or entity that first makes Original Software available under this License.

1.6. "Larger Work" means a work which combines Covered Software or portions thereof with code not governed by the terms of this License.

1.7. "License" means this document.

1.8. "Licensable" means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently acquired, any and all of the rights conveyed herein.

1.9. "Modifications" means the Source Code and Executable form of any of the following:

A. Any file that results from an addition to, deletion from or modification of the contents of a file containing Original Software or previous Modifications;

B. Any new file that contains any part of the Original Software or previous Modification; or

C. Any new file that is contributed or otherwise made available under the terms of this License.

1.10. "Original Software" means the Source Code and Executable form of computer software code that is originally released under this License.

1.11. "Patent Claims" means any patent claim(s), now owned or hereafter acquired, including without limitation, method, process, and apparatus claims, in any patent Licensable by grantor.

1.12. "Source Code" means (a) the common form of computer software code in which modifications are made and (b) associated documentation included in or with such code.

1.13. "You" (or "Your") means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License. For legal entities, "You" includes any entity which controls, is controlled by, or is

under common control with You. For purposes of this definition, "control" means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. License Grants.

2.1. The Initial Developer Grant.

Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, the Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license:

(a) under intellectual property rights (other than patent or trademark) Licensable by Initial Developer, to use, reproduce, modify, display, perform, sublicense and distribute the Original Software (or portions thereof), with or without Modifications, and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using or selling of Original Software, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Software (or portions thereof).

(c) The licenses granted in Sections 2.1(a) and (b) are effective on the date Initial Developer first distributes or otherwise makes the Original Software available to a third party under the terms of this License.

(d) Notwithstanding Section 2.1(b) above, no patent license is granted: (1) for code that You delete from the Original Software, or (2) for infringements caused by: (i) the modification of the Original Software, or (ii) the combination of the Original Software with other software or devices.

2.2. Contributor Grant.

Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:

(a) under intellectual property rights (other than patent or trademark) Licensable by Contributor to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof), either on an unmodified basis, with other Modifications, as Covered Software and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: (1) Modifications made by that Contributor (or portions thereof); and (2) the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination).

(c) The licenses granted in Sections 2.2(a) and 2.2(b) are effective on the date Contributor first distributes or otherwise makes the Modifications available to a third party.

(d) Notwithstanding Section 2.2(b) above, no patent license is granted: (1) for any code that Contributor has deleted from the Contributor Version; (2) for infringements caused by: (i) third party modifications of Contributor Version, or (ii) the combination of Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or (3) under Patent Claims infringed by Covered Software in the absence of Modifications made by that Contributor.

3. Distribution Obligations.

3.1. Availability of Source Code.

Any Covered Software that You distribute or otherwise make available in Executable form must also be made available in Source Code form and that Source Code form must be distributed only under the terms of this License. You must include a copy of this License with every copy of the Source Code form of the Covered Software You distribute or otherwise make available. You must inform recipients of any

such Covered Software in Executable form as to how they can obtain such Covered Software in Source Code form in a reasonable manner on or through a medium customarily used for software exchange.

3.2. Modifications.

The Modifications that You create or to which You contribute are governed by the terms of this License. You represent that You believe Your Modifications are Your original creation(s) and/or You have sufficient rights to grant the rights conveyed by this License.

3.3. Required Notices.

You must include a notice in each of Your Modifications that identifies You as the Contributor of the Modification. You may not remove or alter any copyright, patent or trademark notices contained within the Covered Software, or any notices of licensing or any descriptive text giving attribution to any Contributor or the Initial Developer.

3.4. Application of Additional Terms.

You may not offer or impose any terms on any Covered Software in Source Code form that alters or restricts the applicable version of this License or the recipients rights hereunder. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However, you may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear that any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

3.5. Distribution of Executable Versions.

You may distribute the Executable form of the Covered Software under the terms of this License or under the terms of a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable form does not attempt to limit

or alter the recipients rights in the Source Code form from the rights set forth in this License. If You distribute the Covered Software in Executable form under a different license, You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.

3.6. Larger Works.

You may create a Larger Work by combining Covered Software with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Software.

4. Versions of the License.

4.1. New Versions.

Sun Microsystems, Inc. is the initial license steward and may publish revised and/or new versions of this License from time to time. Each version will be given a distinguishing version number. Except as provided in Section 4.3, no one other than the license steward has the right to modify this License.

4.2. Effect of New Versions.

You may always continue to use, distribute or otherwise make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. If the Initial Developer includes a notice in the Original Software prohibiting it from being distributed or otherwise made available under any subsequent version of the License, You must distribute and make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. Otherwise, You may also choose to use, distribute or otherwise make the Covered Software available under the terms of any subsequent version of the License published by the license steward.

4.3. Modified Versions.

When You are an Initial Developer and You want to create a new license for Your Original Software, You may create and use a modified version of this License if You: (a) rename the license and remove any references to the name of the license steward (except to note that the license differs from this License); and (b) otherwise make it clear that the license contains terms which differ from this License.

5. DISCLAIMER OF WARRANTY.

COVERED SOFTWARE IS PROVIDED UNDER THIS LICENSE ON AN "AS IS" BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED SOFTWARE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED SOFTWARE IS WITH YOU. SHOULD ANY COVERED SOFTWARE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED SOFTWARE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

6. TERMINATION.

6.1. This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.

6.2. If You assert a patent infringement claim (excluding declaratory judgment actions) against Initial Developer or a Contributor (the Initial Developer or Contributor against whom You assert such claim is referred to as "Participant") alleging that the Participant Software (meaning the Contributor Version where the Participant is a Contributor or the Original Software where the Participant is the Initial Developer) directly or indirectly infringes any patent, then any and all rights granted directly or indirectly to You by such Participant, the Initial Developer (if the Initial Developer is not the Participant) and all Contributors under Sections 2.1 and/or 2.2 of this License shall, upon 60 days notice from Participant terminate prospectively and automatically at the expiration of such 60 day notice period, unless if within such 60 day

period You withdraw Your claim with respect to the Participant Software against such Participant either unilaterally or pursuant to a written agreement with Participant.

6.3. In the event of termination under Sections 6.1 or 6.2 above, all end user licenses that have been validly granted by You or any distributor hereunder prior to termination (excluding licenses granted to You by any distributor) shall survive termination.

7. LIMITATION OF LIABILITY.

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL YOU, THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED SOFTWARE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO ANY PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOST PROFITS, LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTYS NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

8. U.S. GOVERNMENT END USERS.

The Covered Software is a "commercial item," as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of "commercial computer software" (as that term is defined at 48 C.F.R. 252.227-7014(a)(1)) and "commercial computer software documentation" as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Software with only those rights set forth herein. This U.S. Government Rights clause is in lieu of, and supersedes, any other FAR, DFAR, or other clause or provision that addresses Government rights in computer software under this License.

9. MISCELLANEOUS.

This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be

unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall be governed by the law of the jurisdiction specified in a notice contained within the Original Software (except to the extent applicable law, if any, provides otherwise), excluding such jurisdictions conflict-of-law provisions. Any litigation relating to this License shall be subject to the jurisdiction of the courts located in the jurisdiction and venue specified in a notice contained within the Original Software, with the losing party responsible for costs, including, without limitation, court costs and reasonable attorneys fees and expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License. You agree that You alone are responsible for compliance with the United States export administration regulations (and the export control laws and regulation of any other countries) when You use, distribute or otherwise make available any Covered Software.

10. RESPONSIBILITY FOR CLAIMS.

As between Initial Developer and the Contributors, each party is responsible for claims and damages arising, directly or indirectly, out of its utilization of rights under this License and You agree to work with Initial Developer and Contributors to distribute such responsibility on an equitable basis. Nothing herein is intended or shall be deemed to constitute any admission of liability.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

* Neither the name of Sun Microsystems, Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL
THE COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
This license came from: http://www.megginson.com/SAX/copying.html
However please note future versions of SAX may be covered
under http://saxproject.org/?selected=pd

This page is now out of date -- see the new SAX site at
http://www.saxproject.org/ for more up-to-date
releases and other information. Please change your bookmarks.

SAX2 is Free!

I hereby abandon any property rights to SAX 2.0 (the Simple API for
XML), and release all of the SAX 2.0 source code, compiled code, and
documentation contained in this distribution into the Public Domain.
SAX comes with NO WARRANTY or guarantee of fitness for any
purpose.

David Megginson, david@megginson.com
2000-05-05
This license came from:
http://www.w3.org/Consortium/Legal/copyright-software-19980720

W3C SOFTWARE NOTICE AND LICENSE
Copyright 1994-2001 World
Wide Web Consortium, World
Wide Web Consortium, (<a href=
"http://www.lcs.mit.edu/">Massachusetts Institute of
Technology, Institut National de
Recherche en Informatique et en Automatique, <a href=
"http://www.keio.ac.jp/">Keio University). All Rights Reserved.
http://www.w3.org/Consortium/Legal/

This W3C work (including software, documents, or other related

items) is being provided by the copyright holders under the
following license. By obtaining, using and/or copying this work,
you (the licensee) agree that you have read, understood, and will
comply with the following terms and conditions:
Permission to use, copy, modify, and distribute this software
and its documentation, with or without modification, for any
purpose and without fee or royalty is hereby granted, provided that
you include the following on ALL copies of the software and
documentation or portions thereof, including modifications, that
you make:

The full text of this NOTICE in a location viewable to users of
the redistributed or derivative work.

Any pre-existing intellectual property disclaimers, notices, or
terms and conditions. If none exist, a short notice of the
following form (hypertext is preferred, text is permitted) should
be used within the body of any redistributed or derivative code:
"Copyright [$date-of-software] World Wide Web Consortium, (Massachusetts Institute of
Technology, Institut National de
Recherche en Informatique et en Automatique, Keio University). All Rights Reserved.
http://www.w3.org/Consortium/Legal/"

Notice of any changes or modifications to the W3C files,
including the date changes were made. (We recommend you provide
URIs to the location from which the code is derived.)

THIS SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS," AND
COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF
MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE
USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE ANY THIRD
PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.
COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT,
SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE
SOFTWARE OR DOCUMENTATION.

The name and trademarks of copyright holders may NOT be used in
advertising or publicity pertaining to the software without
specific, written prior permission. Title to copyright in this
software and any associated documentation will at all times remain
with copyright holders.

This formulation of W3C's notice and license became active on
August 14 1998 so as to improve compatibility with GPL. This
version ensures that W3C software licensing terms are no more
restrictive than GPL and consequently W3C software may be
distributed in GPL packages. See the older formulation for the

policy prior to this date. Please see our Copyright FAQ for common
questions about using materials from
our site, including specific terms and conditions for packages like
libwww, Amaya, and Jigsaw.
Other questions about this notice can be
directed to site-policy@w3.org.

webmaster
Sun Microsystems, Inc.
Binary Code License Agreement

READ THE TERMS OF THIS AGREEMENT AND ANY PROVIDED SUPPLEMENTAL LICENSE TERMS (COLLECTIVELY "AGREEMENT") CAREFULLY BEFORE OPENING THE SOFTWARE MEDIA PACKAGE. BY OPENING THE SOFTWARE MEDIA PACKAGE, YOU AGREE TO THE TERMS OF THIS AGREEMENT. IF YOU ARE ACCESSING THE SOFTWARE ELECTRONICALLY, INDICATE YOUR ACCEPTANCE OF THESE TERMS BY SELECTING THE "ACCEPT" BUTTON AT THE END OF THIS AGREEMENT. IF YOU DO NOT AGREE TO ALL THESE TERMS, PROMPTLY RETURN THE UNUSED SOFTWARE TO YOUR PLACE OF PURCHASE FOR A REFUND OR, IF THE SOFTWARE IS ACCESSED ELECTRONICALLY, SELECT THE "DECLINE" BUTTON AT THE END OF THIS AGREEMENT.

1. LICENSE TO USE. Sun grants you a non-exclusive and
non-transferable license for the internal use only of the
accompanying software and documentation and any error
corrections provided by Sun (collectively "Software"), by
the number of users and the class of computer hardware for
which the corresponding fee has been paid.

2. RESTRICTIONS. Software is confidential and copyrighted.
Title to Software and all associated intellectual property
rights is retained by Sun and/or its licensors. Except as
specifically authorized in any Supplemental License Terms,
you may not make copies of Software, other than a single
copy of Software for archival purposes. Unless enforcement
is prohibited by applicable law, you may not modify,
decompile, or reverse engineer Software. You acknowledge
that Software is not designed, licensed or intended for use
in the design, construction, operation or maintenance of any
nuclear facility. Sun disclaims any express or implied
warranty of fitness for such uses. No right, title or
interest in or to any trademark, service mark, logo or trade
name of Sun or its licensors is granted under this
Agreement.

3. LIMITED WARRANTY. Sun warrants to you that for a period
of ninety (90) days from the date of purchase, as evidenced

by a copy of the receipt, the media on which Software is furnished (if any) will be free of defects in materials and workmanship under normal use. Except for the foregoing, Software is provided "AS IS". Your exclusive remedy and Sun's entire liability under this limited warranty will be at Sun's option to replace Software media or refund the fee paid for Software.

4. DISCLAIMER OF WARRANTY. UNLESS SPECIFIED IN THIS AGREEMENT, ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT THESE DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

5. LIMITATION OF LIABILITY. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL SUN OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR SPECIAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF OR RELATED TO THE USE OF OR INABILITY TO USE SOFTWARE, EVEN IF SUN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. In no event will Sun's liability to you, whether in contract, tort (including negligence), or otherwise, exceed the amount paid by you for Software under this Agreement. The foregoing limitations will apply even if the above stated warranty fails of its essential purpose.

6. Termination. This Agreement is effective until terminated. You may terminate this Agreement at any time by destroying all copies of Software. This Agreement will terminate immediately without notice from Sun if you fail to comply with any provision of this Agreement. Upon Termination, you must destroy all copies of Software.

7. Export Regulations. All Software and technical data delivered under this Agreement are subject to US export control laws and may be subject to export or import regulations in other countries. You agree to comply strictly with all such laws and regulations and acknowledge that you have the responsibility to obtain such licenses to export, re-export, or import as may be required after delivery to you.

8. U.S. Government Restricted Rights. If Software is being acquired by or on behalf of the U.S. Government or by a U.S. Government prime contractor or subcontractor (at any

tier), then the Government's rights in Software and accompanying documentation will be only as set forth in this Agreement; this is in accordance with 48 CFR 227.7201 through 227.7202-4 (for Department of Defense (DOD) acquisitions) and with 48 CFR 2.101 and 12.212 (for non-DOD acquisitions).

9. Governing Law. Any action related to this Agreement will be governed by California law and controlling U.S. federal law. No choice of law rules of any jurisdiction will apply.

10. Severability. If any provision of this Agreement is held to be unenforceable, this Agreement will remain in effect with the provision omitted, unless omission would frustrate the intent of the parties, in which case this Agreement will immediately terminate.

11. Integration. This Agreement is the entire agreement between you and Sun relating to its subject matter. It supersedes all prior or contemporaneous oral or written communications, proposals, representations and warranties and prevails over any conflicting or additional terms of any quote, order, acknowledgment, or other communication between the parties relating to its subject matter during the term of this Agreement. No modification of this Agreement will be binding, unless in writing and signed by an authorized representative of each party.

SUN XML INSTANCE GENERATOR, VERSION 1.0 SUPPLEMENTAL LICENSE TERMS

These supplemental license terms ("Supplemental Terms") add to or modify the terms of the Binary Code License Agreement (collectively, the "Agreement"). Capitalized terms not defined in these Supplemental Terms shall have the same meanings ascribed to them in the Agreement. These Supplemental Terms shall supersede any inconsistent or conflicting terms in the Agreement, or in any license contained within the Software.

1. Software Internal Use and Development License Grant. Subject to the terms and conditions of this Agreement, including, but not limited to Section 4 (Java(TM) Technology Restrictions) of these Supplemental Terms, Sun grants to you, a non-exclusive, non-transferable, royalty-free and limited license to reproduce, modify, and create derivative

works of the Software for the sole purpose of adding value and improving the Software for the development of applications ("Programs").

2. License to Distribute Software. Subject to the terms and conditions of this Agreement, including, but not limited to Section 4 (Java (TM) Technology Restrictions) of these Supplemental Terms, Sun grants you a non-exclusive, non-transferable, limited license to reproduce and distribute the Software modified by you as permitted in Section 1 of these Supplemental Terms ("Modified Software") in source or binary code form, provided that (i) you distribute the Modified Software only bundled as part of, and for the sole purpose of running, your Programs, (ii) the Modified Software adds value and improveS the function of the Software, (iv) you do not remove or alter any proprietary legends or notices contained in the Software, (v) you only distribute the Modified Software subject to a license agreement that protects Sun's interests consistent with the terms contained in this Agreement, and (vi) you agree to defend and indemnify Sun and its licensors from and against any damages, costs, liabilities, settlement amounts and/or expenses (including attorneys' fees) incurred in connection with any claim, lawsuit or action by any third party that arises or results from the use or distribution of any and all Programs and/or Modified Software.

3. Experimental Software. You acknowledge that the Software is experimental and may contain errors, defects, or deficiencies which cannot or will not be corrected by Sun. You shall have the sole responsibility to protect adequately and backup your data and/or equipment used in connection with the Software. You shall not claim against Sun for lost data, re-run time, inaccurate output, work delays or lost profits resulting from your use of the Licensed Software.

4. Java Technology Restrictions. You may not modify the Java Platform Interface ("JPI", identified as classes contained within the "java" package or any subpackages of the "java" package), by creating additional classes within the JPI or otherwise causing the addition to or modification of the classes in the JPI. In the event that you create an additional class and associated API(s) which (i) extends the functionality of the Java platform, and (ii) is exposed to third party software developers for the purpose of developing additional software which invokes such additional API, you must promptly publish broadly an accurate specification for such API for free use by all developers.

You may not create, or authorize your licensees to create, additional classes, interfaces, or subpackages that are in any way identified as "java", "javax", "sun" or similar convention as specified by Sun in any naming convention designation.

5. Trademarks and Logos. You acknowledge and agree as between you and Sun that Sun owns the SUN, SOLARIS, JAVA, JINI, FORTE, and iPLANET trademarks and all SUN, SOLARIS, JAVA, JINI, FORTE, and iPLANET-related trademarks, service marks, logos and other brand designations ("Sun Marks"), and you agree to comply with the Sun Trademark and Logo Usage Requirements currently located at http://www.sun.com/policies/trademarks. Any use you make of the Sun Marks inures to Sun's benefit.

6. Termination for Infringement. Either party may terminate this Agreement immediately should any Software become, or in either party's opinion be likely to become, the subject of a claim of infringement of any intellectual property right.

For inquiries please contact:
Sun Microsystems, Inc.
901 San Antonio Road, Palo Alto, California 94303
(LFI#100313/Form ID#011801)

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS

BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,
OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER
IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF
THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
THE POSSIBILITY OF SUCH DAMAGE.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

-Redistribution of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

-Redistribution in binary form must reproduce the above copyright notice,
this list of conditions and the following disclaimer in the documentation and/or
other materials provided with the distribution.

Neither the name of Sun Microsystems, Inc. or the names of contributors may
be used to endorse or promote products derived from this software without
specific prior written permission.

This software is provided "AS IS," without a warranty of any kind. ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SUN MIDROSYSTEMS, INC. ("SUN") AND ITS LICENSORS SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL SUN OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE USE OF OR INABILITY TO USE THIS SOFTWARE, EVEN IF SUN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You acknowledge that this software is not designed, licensed or intended for use in the design, construction, operation or maintenance of any nuclear facility.

Sun Microsystems, Inc.
Binary Code License Agreement

READ THE TERMS OF THIS AGREEMENT AND ANY PROVIDED SUPPLEMENTAL LICENSE TERMS (COLLECTIVELY "AGREEMENT") CAREFULLY BEFORE OPENING THE SOFTWARE MEDIA PACKAGE. BY OPENING THE SOFTWARE MEDIA PACKAGE, YOU AGREE TO THE TERMS OF THIS AGREEMENT. IF YOU ARE ACCESSING THE SOFTWARE ELECTRONICALLY, INDICATE YOUR ACCEPTANCE OF THESE TERMS BY SELECTING THE "ACCEPT" BUTTON AT THE END OF THIS AGREEMENT. IF YOU DO NOT AGREE TO ALL THESE TERMS, PROMPTLY RETURN

THE UNUSED SOFTWARE TO YOUR PLACE OF PURCHASE FOR A REFUND OR, IF THE SOFTWARE IS ACCESSED ELECTRONICALLY, SELECT THE "DECLINE" BUTTON AT THE END OF THIS AGREEMENT.

1. LICENSE TO USE. Sun grants you a non-exclusive and non-transferable license for the internal use only of the accompanying software and documentation and any error corrections provided by Sun (collectively "Software"), by the number of users and the class of computer hardware for which the corresponding fee has been paid.

2. RESTRICTIONS. Software is confidential and copyrighted. Title to Software and all associated intellectual property rights is retained by Sun and/or its licensors. Except as specifically authorized in any Supplemental License Terms, you may not make copies of Software, other than a single copy of Software for archival purposes. Unless enforcement is prohibited by applicable law, you may not modify, decompile, or reverse engineer Software. You acknowledge that Software is not designed, licensed or intended for use in the design, construction, operation or maintenance of any nuclear facility. Sun disclaims any express or implied warranty of fitness for such uses. No right, title or interest in or to any trademark, service mark, logo or trade name of Sun or its licensors is granted under this Agreement.

3. LIMITED WARRANTY. Sun warrants to you that for a period of ninety (90) days from the date of purchase, as evidenced by a copy of the receipt, the media on which Software is furnished (if any) will be free of defects in materials and workmanship under normal use. Except for the foregoing, Software is provided "AS IS". Your exclusive remedy and Sun's entire liability under this limited warranty will be at Sun's option to replace Software media or refund the fee paid for Software.

4. DISCLAIMER OF WARRANTY. UNLESS SPECIFIED IN THIS AGREEMENT, ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT THESE DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

5. LIMITATION OF LIABILITY. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL SUN OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR SPECIAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF OR RELATED TO THE USE OF OR INABILITY TO USE SOFTWARE, EVEN IF SUN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. In no event will Sun's liability to you, whether in contract, tort (including negligence), or otherwise, exceed the amount paid by you for Software under this Agreement. The foregoing limitations will apply even if the above stated warranty fails of its essential purpose.

6. Termination. This Agreement is effective until terminated. You may terminate this Agreement at any time by destroying all copies of Software. This Agreement will terminate immediately without notice from Sun if you fail to comply with any provision of this Agreement. Upon Termination, you must destroy all copies of Software.

7. Export Regulations. All Software and technical data delivered under this Agreement are subject to US export control laws and may be subject to export or import regulations in other countries. You agree to comply strictly with all such laws and regulations and acknowledge that you have the responsibility to obtain such licenses to export, re-export, or import as may be required after delivery to you.

8. U.S. Government Restricted Rights. If Software is being acquired by or on behalf of the U.S. Government or by a U.S. Government prime contractor or subcontractor (at any tier), then the Government's rights in Software and accompanying documentation will be only as set forth in this Agreement; this is in accordance with 48 CFR 227.7201 through 227.7202-4 (for Department of Defense (DOD) acquisitions) and with 48 CFR 2.101 and 12.212 (for non-DOD acquisitions).

9. Governing Law. Any action related to this Agreement will be governed by California law and controlling U.S. federal law. No choice of law rules of any jurisdiction will apply.

10. Severability. If any provision of this Agreement is held to be unenforceable, this Agreement will remain in effect with the provision omitted, unless omission would frustrate the intent of the parties, in which case this Agreement will immediately terminate.

11. Integration. This Agreement is the entire agreement between you and Sun relating to its subject matter. It supersedes all prior or contemporaneous oral or written communications, proposals, representations and warranties and prevails over any conflicting or additional terms of any quote, order, acknowledgment, or other communication between the parties relating to its subject matter during the term of this Agreement. No modification of this Agreement will be binding, unless in writing and signed by an authorized representative of each party.

JAVA OPTIONAL PACKAGE

JAVABEANS(TM) ACTIVATION FRAMEWORK, VERSION 1.0.2
SUPPLEMENTAL LICENSE TERMS

These supplemental license terms ("Supplemental Terms") add to or modify the terms of the Binary Code License Agreement (collectively, the "Agreement"). Capitalized terms not defined in these Supplemental Terms shall have the same meanings ascribed to them in the Agreement. These Supplemental Terms

shall supersede any inconsistent or conflicting terms in the Agreement, or in any license contained within the Software.

1. Software Internal Use and Development License Grant. Subject to the terms and conditions of this Agreement, including, but not limited to Section 3 (Java(TM) Technology Restrictions) of these Supplemental Terms, Sun grants you a non-exclusive, non-transferable, limited license to reproduce internally and use internally the binary form of the Software, complete and unmodified, for the sole purpose of designing, developing and testing your Java applets and applications ("Programs").

2. License to Distribute Software. In addition to the license granted in Section 1 (Software Internal Use and Development License Grant) of these Supplemental Terms, subject to the terms and conditions of this Agreement, including but not limited to, Section 3 (Java Technology Restrictions) of these Supplemental Terms, Sun grants you a non-exclusive, non-transferable, limited license to reproduce and distribute the Software in binary code form only, provided that you (i) distribute the Software complete and unmodified and only bundled as part of your Programs, (ii) do not distribute additional software intended to replace any component(s) of the Software, (iii) do not remove or alter any proprietary legends or notices contained in the Software, (iv) only distribute the Software subject to a license agreement that protects Sun's interests consistent with the terms contained in this Agreement, and (v) agree to defend and indemnify Sun and its licensors from and against any damages, costs, liabilities, settlement amounts and/or expenses (including attorneys' fees) incurred in connection with any claim, lawsuit or action by any third party that arises or results from the use or distribution of any and all Programs and/or Software.

3. Java Technology Restrictions. You may not modify the Java Platform Interface ("JPI", identified as classes contained within the "java" package or any subpackages of the "java" package), by creating additional classes within the JPI or otherwise causing the addition to or modification of the classes in the JPI. In the event that you create an additional class and associated API(s) which (i) extends the functionality of the Java platform, and (ii) is exposed to third party software developers for the purpose of developing additional software which invokes such additional API, you must promptly publish broadly an accurate specification for such API for free use by all developers. You may not create, or authorize your licensees to create additional classes, interfaces, or subpackages that are in any way identified as "java", "javax", "sun" or similar convention as specified by Sun in any naming convention designation.

4. No Support. Sun is under no obligation to support the Software or to provide you with updates or error corrections. You acknowledge that the Software may have defects or deficiencies which cannot or will not be corrected by Sun.

5. Trademarks and Logos. You acknowledge and agree as between you and Sun that Sun owns the SUN, SOLARIS, JAVA, JINI, FORTE, and iPLANET trademarks and all SUN, SOLARIS, JAVA, JINI, FORTE, and iPLANET-related trademarks, service marks, logos and other brand designations ("Sun Marks"), and you agree to comply with the Sun Trademark and Logo Usage Requirements currently located at http://www.sun.com/policies/trademarks. Any use you make of the Sun Marks inures to Sun's benefit.

6. Source Code. Software may contain source code that is provided solely for reference purposes pursuant to the terms of this Agreement. Source code may not be redistributed unless expressly provided for in this Agreement.

7. Termination for Infringement. Either party may terminate this Agreement immediately should any Software become, or in either party's opinion be likely to become, the subject of a claim of infringement of any intellectual property right.

For inquiries please contact: Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, California 94303
(LFI#115020/Form ID#011801)

```
/*
* The Apache Software License, Version 1.1
*
*
* Copyright (c) 1999-2002 The Apache Software Foundation.  All rights
* reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
*
* 1. Redistributions of source code must retain the above copyright
*    notice, this list of conditions and the following disclaimer.
*
* 2. Redistributions in binary form must reproduce the above copyright
*    notice, this list of conditions and the following disclaimer in
*    the documentation and/or other materials provided with the
*    distribution.
*
* 3. The end-user documentation included with the redistribution,
*    if any, must include the following acknowledgment:
*       "This product includes software developed by the
*        Apache Software Foundation (http://www.apache.org/)."
*    Alternately, this acknowledgment may appear in the software itself,
*    if and wherever such third-party acknowledgments normally appear.
*
* 4. The names "Xerces" and "Apache Software Foundation" must
*    not be used to endorse or promote products derived from this
```

* software without prior written permission. For written
* permission, please contact apache@apache.org.
*
* 5. Products derived from this software may not be called "Apache",
* nor may "Apache" appear in their name, without prior written
* permission of the Apache Software Foundation.
*
* THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED
* WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
* OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
* DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR
* ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
* LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
* USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
* ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
* OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
* OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
* ==
*
* This software consists of voluntary contributions made by many
* individuals on behalf of the Apache Software Foundation and was
* originally based on software copyright (c) 1999, International
* Business Machines, Inc., http://www.ibm.com. For more
* information on the Apache Software Foundation, please see
* <http://www.apache.org/>.
*/
Copyright (c) 2004 Kohsuke Kawaguchi

Permission is hereby granted, free of charge, to any person
obtaining a copy of this software and associated documentation
files (the "Software"), to deal in the Software without
restriction, including without limitation the rights to use,
copy, modify, merge, publish, distribute, sublicense, and/or
sell copies of the Software, and to permit persons to whom
the Software is furnished to do so, subject to the following
conditions:

The above copyright notice and this permission notice shall
be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY
KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS
OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR

OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL) Version 1.0

1. Definitions.

1.1. Contributor. means each individual or entity that creates or contributes to the creation of Modifications.

1.2. Contributor Version. means the combination of the Original Software, prior Modifications used by a Contributor (if any), and the Modifications made by that particular Contributor.

1.3. Covered Software. means (a) the Original Software, or (b) Modifications, or (c) the combination of files containing Original Software with files containing Modifications, in each case including portions thereof.

1.4. Executable. means the Covered Software in any form other than Source Code.

1.5. Initial Developer. means the individual or entity that first makes Original Software available under this License.

1.6. Larger Work. means a work which combines Covered Software or portions thereof with code not governed by the terms of this License.

1.7. License. means this document.

1.8. Licensable. means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently acquired, any and all of the rights conveyed herein.

1.9. Modifications. means the Source Code and Executable form of any of the following:

A. Any file that results from an addition to, deletion from or modification of the contents of a file containing Original Software or previous Modifications;

B. Any new file that contains any part of the Original Software or previous Modification; or

C. Any new file that is contributed or otherwise made available under the terms of this License.

1.10. Original Software. means the Source Code and Executable form of computer software code that is originally released under this License.

1.11. Patent Claims. means any patent claim(s), now owned or hereafter acquired, including without limitation, method, process, and apparatus claims, in any patent Licensable by grantor.

1.12. Source Code. means (a) the common form of computer software code in which modifications are made and (b) associated documentation included in or with such code.

1.13. You. (or .Your.) means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License. For legal entities, .You. includes any entity which controls, is controlled by, or is under common control with You. For purposes of this definition, .control. means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. License Grants.

2.1. The Initial Developer Grant.

Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, the Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license:

(a) under intellectual property rights (other than patent or trademark) Licensable by Initial Developer, to use, reproduce, modify, display, perform, sublicense and distribute the Original Software (or portions thereof), with or without Modifications, and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using or selling of Original Software, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Software (or portions thereof).

(c) The licenses granted in Sections 2.1(a) and (b) are effective on the date Initial Developer first distributes or otherwise makes the Original Software available to a third party under the terms of this License.

(d) Notwithstanding Section 2.1(b) above, no patent license is granted: (1) for code that You delete from the Original Software, or (2) for infringements caused by: (i) the modification of the Original Software, or (ii) the combination of the Original Software with other software or devices.

2.2. Contributor Grant.

Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:

(a) under intellectual property rights (other than patent or trademark) Licensable by Contributor to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof), either on an unmodified basis, with other Modifications, as Covered Software and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: (1) Modifications made by that Contributor (or portions thereof); and (2) the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination).

(c) The licenses granted in Sections 2.2(a) and 2.2(b) are effective on the date Contributor first distributes or otherwise makes the Modifications available to a third party.

(d) Notwithstanding Section 2.2(b) above, no patent license is granted: (1) for any code that Contributor has deleted from the Contributor Version; (2) for infringements caused by: (i) third party modifications of Contributor Version, or (ii) the combination of Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or (3) under Patent Claims infringed by Covered Software in the absence of Modifications made by that Contributor.

3. Distribution Obligations.

3.1. Availability of Source Code.

Any Covered Software that You distribute or otherwise make available in Executable form must also be made available in Source Code form and that Source Code form must be distributed only under the terms of this License. You must include a copy of this License with every copy of the Source Code form of the Covered Software You distribute or otherwise make available. You must inform recipients of any such Covered Software in Executable form as to how they can obtain such Covered Software in Source Code form in a reasonable manner on or through a medium customarily used for software exchange.

3.2. Modifications.

The Modifications that You create or to which You contribute are governed by the terms of this License. You represent that You believe Your Modifications are Your original creation(s) and/or You have sufficient rights to grant the rights conveyed by this License.

3.3. Required Notices.

You must include a notice in each of Your Modifications that identifies You as the Contributor of the Modification. You may not remove or alter any copyright, patent or trademark notices contained within the Covered Software, or any notices of licensing or any descriptive text giving attribution to any Contributor or the Initial Developer.

3.4. Application of Additional Terms.

You may not offer or impose any terms on any Covered Software in Source Code form that alters or restricts the applicable version of this License or the recipients. rights hereunder. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However, you may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear that any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

3.5. Distribution of Executable Versions.

You may distribute the Executable form of the Covered Software under the terms of this License or under the terms of a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable form does not attempt to limit or alter the recipient.s rights in the Source Code form from the rights set forth in this License. If You distribute the Covered Software in Executable form under a different license, You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.

3.6. Larger Works.

You may create a Larger Work by combining Covered Software with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the

requirements of this License are fulfilled for the Covered Software.

4. Versions of the License.

4.1. New Versions.

Sun Microsystems, Inc. is the initial license steward and may publish revised and/or new versions of this License from time to time. Each version will be given a distinguishing version number. Except as provided in Section 4.3, no one other than the license steward has the right to modify this License.

4.2. Effect of New Versions.

You may always continue to use, distribute or otherwise make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. If the Initial Developer includes a notice in the Original Software prohibiting it from being distributed or otherwise made available under any subsequent version of the License, You must distribute and make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. Otherwise, You may also choose to use, distribute or otherwise make the Covered Software available under the terms of any subsequent version of the License published by the license steward.

4.3. Modified Versions.

When You are an Initial Developer and You want to create a new license for Your Original Software, You may create and use a modified version of this License if You: (a) rename the license and remove any references to the name of the license steward (except to note that the license differs from this License); and (b) otherwise make it clear that the license contains terms which differ from this License.

5. DISCLAIMER OF WARRANTY.

COVERED SOFTWARE IS PROVIDED UNDER THIS LICENSE ON AN .AS IS. BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED SOFTWARE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED SOFTWARE IS WITH YOU. SHOULD ANY COVERED SOFTWARE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED SOFTWARE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

6. TERMINATION.

6.1. This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.

6.2. If You assert a patent infringement claim (excluding declaratory judgment actions) against Initial Developer or a Contributor (the Initial Developer or Contributor against whom You assert such claim is referred to as .Participant.) alleging that the Participant Software (meaning the Contributor Version where the Participant is a Contributor or the Original Software where the Participant is the Initial Developer) directly or indirectly infringes any patent, then any and all rights granted directly or indirectly to You by such Participant, the Initial Developer (if the Initial Developer is not the Participant) and all Contributors under Sections 2.1 and/or 2.2 of this License shall, upon 60 days notice from Participant terminate prospectively and automatically at the expiration of such 60 day notice period, unless if within such 60 day period You withdraw Your claim with respect to the Participant Software against such Participant either unilaterally or pursuant to a written agreement with Participant.

6.3. In the event of termination under Sections 6.1 or 6.2 above, all end user licenses that have been validly granted by You or any distributor hereunder prior to termination (excluding licenses granted to You by any distributor) shall survive termination.

7. LIMITATION OF LIABILITY.

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL YOU, THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED SOFTWARE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO ANY PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOST PROFITS, LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTY.S NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

8. U.S. GOVERNMENT END USERS.

The Covered Software is a .commercial item,. as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of .commercial computer

software. (as that term is defined at 48 C.F.R. 252.227-7014(a)(1)) and .commercial computer software documentation. as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Software with only those rights set forth herein. This U.S. Government Rights clause is in lieu of, and supersedes, any other FAR, DFAR, or other clause or provision that addresses Government rights in computer software under this License.

9. MISCELLANEOUS.

This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall be governed by the law of the jurisdiction specified in a notice contained within the Original Software (except to the extent applicable law, if any, provides otherwise), excluding such jurisdiction.s conflict-of-law provisions. Any litigation relating to this License shall be subject to the jurisdiction of the courts located in the jurisdiction and venue specified in a notice contained within the Original Software, with the losing party responsible for costs, including, without limitation, court costs and reasonable attorneys. fees and expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License. You agree that You alone are responsible for compliance with the United States export administration regulations (and the export control laws and regulation of any other countries) when You use, distribute or otherwise make available any Covered Software.

10. RESPONSIBILITY FOR CLAIMS.

As between Initial Developer and the Contributors, each party is responsible for claims and damages arising, directly or indirectly, out of its utilization of rights under this License and You agree to work with Initial Developer and Contributors to distribute such responsibility on an equitable basis. Nothing herein is intended or shall be deemed to constitute any admission of liability.

NOTICE PURSUANT TO SECTION 9 OF THE COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL)

The code released under the CDDL shall be governed by the laws of the State of California (excluding conflict-of-law provisions). Any litigation relating to this License shall be subject to the jurisdiction of the Federal Courts of the Northern District of California and the state courts of the State of California, with venue lying in Santa Clara

County, California.

The GNU General Public License (GPL) Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so

that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices

stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable

copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions

on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

One line to give the program's name and a brief idea of what it does.

Copyright (C)

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'. This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

"CLASSPATH" EXCEPTION TO THE GPL VERSION 2

Certain source files distributed by Sun Microsystems, Inc. are subject to the following clarification and special exception to the GPL Version 2, but only where Sun has expressly included in the particular source file's header the words

"Sun designates this particular file as subject to the "Classpath" exception as provided by Sun in the License file that accompanied this code."

Linking this library statically or dynamically with other modules is making a combined work based on this library. Thus, the terms and conditions of the GNU General Public License Version 2 cover the whole combination.

As a special exception, the copyright holders of this library give you permission to link this library with independent modules to produce an executable, regardless of the license terms of these independent modules, and to copy and distribute the resulting executable under terms of your choice, provided that you also meet, for each linked independent module, the terms and conditions of the license of that module.? An independent module is a module which is not derived from or based on this library.? If you modify this library, you may extend this exception to your version of the library, but you are not obligated to do so.? If you do not wish to do so, delete this exception statement from your version.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- Redistribution in binary form must reproduct the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of Sun Microsystems, Inc. or the names of

contributors may be used to endorse or promote products derived from
this software without specific prior written permission.

This software is provided "AS IS," without a warranty of any kind. ALL
EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES,
INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A
PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SUN AND
ITS LICENSORS SHALL NOT BE LIABLE FOR ANY DAMAGES OR LIABILITIES
SUFFERED BY LICENSEE AS A RESULT OF OR RELATING TO USE, MODIFICATION
OR DISTRIBUTION OF THE SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL
SUN OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA,
OR FOR DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR
PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF
LIABILITY, ARISING OUT OF THE USE OF OR INABILITY TO USE SOFTWARE,
EVEN IF SUN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You acknowledge that Software is not designed,licensed or intended for
use in the design, construction, operation or maintenance of any
nuclear facility.
Copyright (c) 2003, Kohsuke Kawaguchi
All rights reserved.

Redistribution and use in source and binary forms,
with or without modification, are permitted provided
that the following conditions are met:

* Redistributions of source code must retain
the above copyright notice, this list of
conditions and the following disclaimer.
* Redistributions in binary form must reproduce
the above copyright notice, this list of
conditions and the following disclaimer in
the documentation and/or other materials
provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS
OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS
BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,
OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER
IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING

NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF
THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
THE POSSIBILITY OF SUCH DAMAGE.
Copyright 2001-2005 (C) MetaStuff, Ltd. All Rights Reserved.

Redistribution and use of this software and associated documentation
("Software"), with or without modification, are permitted provided
that the following conditions are met:

1. Redistributions of source code must retain copyright
  statements and notices. Redistributions must also contain a
  copy of this document.

2. Redistributions in binary form must reproduce the
  above copyright notice, this list of conditions and the
  following disclaimer in the documentation and/or other
  materials provided with the distribution.

3. The name "DOM4J" must not be used to endorse or promote
  products derived from this Software without prior written
  permission of MetaStuff, Ltd. For written permission,
  please contact dom4j-info@metastuff.com.

4. Products derived from this Software may not be called "DOM4J"
  nor may "DOM4J" appear in their names without prior written
  permission of MetaStuff, Ltd. DOM4J is a registered
  trademark of MetaStuff, Ltd.

5. Due credit should be given to the DOM4J Project -
  http://www.dom4j.org

THIS SOFTWARE IS PROVIDED BY METASTUFF, LTD. AND CONTRIBUTORS
``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT
NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL
METASTUFF, LTD. OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
OF THE POSSIBILITY OF SUCH DAMAGE.
Copyright 2001-@@YEAR@@ Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto,
California, 94303, U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to
technology embodied in this product. In particular, and without

limitation, these intellectual property rights may include one or more of the U.S. patents listed at http://www.sun.com/patents and one or more additional patents or pending patent applications in the U.S. and other countries. This product is distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third party software, including font technology, is copyrighted and licensed from Sun suppliers. Sun, the Sun logo, and Sun Microsystems are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. This product includes software developed by the Apache Software Foundation (http://www.apache.org/). Federal Acquisitions: Commercial Software - Government Users Subject to Standard License Terms and Conditions.

Copyright 2001-@@YEAR@@ Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, Californie 94303 tats-Unis. Tous droits rservs. Distribue par des licences qui en restreignent l'utilisation. Sun Microsystems, Inc. a les droits de proprit intellectuels relatants la technologie incorpore dans ce produit. En particulier, et sans la limitation, ces droits de proprit intellectuels peuvent inclure un ou plus des brevets amricains numrs http://www.sun.com/patents et un ou les brevets plus supplmentaires ou les applications de brevet en attente dans les Etats Unis et les autres pays. Ce produit ou document est protg par un copyright et distribu avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la dcompilation. Aucune partie de ce produit ou document ne peut tre reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation pralable et crite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel dtenu par des tiers, et qui comprend la technologie relative aux polices de caractres, est protg par un copyright et licenci par des fournisseurs de Sun. Sun, le logo Sun, Sun Microsystems et sont des marques de fabrique ou des marques d?pos?es de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Ce produit inclut le logiciel dvelopp par la base de Apache Software Foundation (http://www.apache.org/). L'accord du gouvernement des tats Unis est requis avant l'exportation du produit.
COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL) Version 1.0

1. Definitions.

1.1. Contributor. means each individual or entity that creates or contributes to the creation of Modifications.

1.2. Contributor Version. means the combination of the Original Software, prior Modifications used by a Contributor (if any), and the Modifications made by that particular Contributor.

1.3. Covered Software. means (a) the Original Software, or (b) Modifications, or (c) the combination of files containing Original Software with files containing Modifications, in each case including portions thereof.

1.4. Executable. means the Covered Software in any form other than Source Code.

1.5. Initial Developer. means the individual or entity that first makes Original Software available under this License.

1.6. Larger Work. means a work which combines Covered Software or portions thereof with code not governed by the terms of this License.

1.7. License. means this document.

1.8. Licensable. means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently acquired, any and all of the rights conveyed herein.

1.9. Modifications. means the Source Code and Executable form of any of the following:

A. Any file that results from an addition to, deletion from or modification of the contents of a file containing Original Software or previous Modifications;

B. Any new file that contains any part of the Original Software or previous Modification; or

C. Any new file that is contributed or otherwise made available under the terms of this License.

1.10. Original Software. means the Source Code and Executable form of computer software code that is originally released under this License.

1.11. Patent Claims. means any patent claim(s), now owned or hereafter acquired, including without limitation, method, process, and apparatus claims, in any patent Licensable by grantor.

1.12. Source Code. means (a) the common form of computer software code in which modifications are made and (b) associated documentation included in or with such code.

1.13. You. (or .Your.) means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License. For legal entities, .You. includes any entity which controls, is controlled by, or is under common control with You. For purposes of this definition, .control. means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. License Grants.

2.1. The Initial Developer Grant.

Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, the Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license:

(a) under intellectual property rights (other than patent or trademark) Licensable by Initial Developer, to use,

reproduce, modify, display, perform, sublicense and distribute the Original Software (or portions thereof), with or without Modifications, and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using or selling of Original Software, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Software (or portions thereof).

(c) The licenses granted in Sections 2.1(a) and (b) are effective on the date Initial Developer first distributes or otherwise makes the Original Software available to a third party under the terms of this License.

(d) Notwithstanding Section 2.1(b) above, no patent license is granted: (1) for code that You delete from the Original Software, or (2) for infringements caused by: (i) the modification of the Original Software, or (ii) the combination of the Original Software with other software or devices.

2.2. Contributor Grant.

Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:

(a) under intellectual property rights (other than patent or trademark) Licensable by Contributor to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof), either on an unmodified basis, with other Modifications, as Covered Software and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: (1) Modifications made by that Contributor (or portions thereof); and (2) the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination).

(c) The licenses granted in Sections 2.2(a) and 2.2(b) are effective on the date Contributor first distributes or otherwise makes the Modifications available to a third party.

(d) Notwithstanding Section 2.2(b) above, no patent license is granted: (1) for any code that Contributor has deleted from the Contributor Version; (2) for infringements caused by: (i) third party modifications of Contributor Version, or (ii) the combination of Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or (3) under Patent Claims infringed by Covered Software in the absence of Modifications made by that Contributor.

3. Distribution Obligations.

3.1. Availability of Source Code.

Any Covered Software that You distribute or otherwise make available in Executable form must also be made available in Source Code form and that Source Code form must be distributed only under the terms of this License. You must include a copy of this License with every copy of the Source Code form of the Covered Software You distribute or otherwise make available. You must inform recipients of any such Covered Software in Executable form as to how they can obtain such Covered Software in Source Code form in a reasonable manner on or through a medium customarily used for software exchange.

3.2. Modifications.

The Modifications that You create or to which You contribute are governed by the terms of this License. You represent that You believe Your Modifications are Your original creation(s) and/or You have sufficient rights to grant the rights conveyed by this License.

3.3. Required Notices.

You must include a notice in each of Your Modifications that identifies You as the Contributor of the Modification. You may not remove or alter any copyright, patent or trademark notices contained within the Covered Software, or any notices of licensing or any descriptive text giving attribution to any Contributor or the Initial Developer.

3.4. Application of Additional Terms.

You may not offer or impose any terms on any Covered Software in Source Code form that alters or restricts the applicable version of this License or the recipients. rights hereunder. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However, you may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear that any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

3.5. Distribution of Executable Versions.

You may distribute the Executable form of the Covered Software under the terms of this License or under the terms of a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable form does not attempt to limit or alter the recipient.s rights in the Source Code form from the rights set forth in this License. If You distribute the Covered Software in Executable form under a different license, You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.

3.6. Larger Works.

You may create a Larger Work by combining Covered Software with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Software.

4. Versions of the License.

4.1. New Versions.

Sun Microsystems, Inc. is the initial license steward and may publish revised and/or new versions of this License from time to time. Each version will be given a distinguishing version number. Except as provided in Section 4.3, no one other than the license steward has the right to modify this License.

4.2. Effect of New Versions.

You may always continue to use, distribute or otherwise make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. If the Initial Developer includes a notice in the Original Software prohibiting it from being distributed or otherwise made available under any subsequent version of the License, You must distribute and make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. Otherwise, You may

also choose to use, distribute or otherwise make the Covered Software available under the terms of any subsequent version of the License published by the license steward.

4.3. Modified Versions.

When You are an Initial Developer and You want to create a new license for Your Original Software, You may create and use a modified version of this License if You: (a) rename the license and remove any references to the name of the license steward (except to note that the license differs from this License); and (b) otherwise make it clear that the license contains terms which differ from this License.

5. DISCLAIMER OF WARRANTY.

COVERED SOFTWARE IS PROVIDED UNDER THIS LICENSE ON AN .AS IS. BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED SOFTWARE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED SOFTWARE IS WITH YOU. SHOULD ANY COVERED SOFTWARE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED SOFTWARE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

6. TERMINATION.

6.1. This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.

6.2. If You assert a patent infringement claim (excluding declaratory judgment actions) against Initial Developer or a Contributor (the Initial Developer or Contributor against whom You assert such claim is referred to as .Participant.) alleging that the Participant Software (meaning the Contributor Version where the Participant is a Contributor or the Original Software where the Participant is the Initial Developer) directly or indirectly infringes any patent, then any and all rights granted directly or indirectly to You by such Participant, the Initial Developer (if the Initial Developer is not the Participant) and all Contributors under Sections 2.1 and/or 2.2 of this License shall, upon 60 days notice from Participant terminate prospectively and automatically at the expiration of such 60 day notice period, unless if within such 60 day period You withdraw Your claim with respect to the Participant Software against such Participant either unilaterally or pursuant to a written agreement with Participant.

6.3. In the event of termination under Sections 6.1 or 6.2 above, all end user licenses that have been validly granted by You or any distributor hereunder prior to termination (excluding licenses granted to You by any distributor) shall survive termination.

7. LIMITATION OF LIABILITY.

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL YOU, THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED SOFTWARE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO ANY PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES

FOR LOST PROFITS, LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTY.S NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

8. U.S. GOVERNMENT END USERS.

The Covered Software is a .commercial item,. as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of .commercial computer software. (as that term is defined at 48 C.F.R. ? 252.227-7014(a)(1)) and .commercial computer software documentation. as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Software with only those rights set forth herein. This U.S. Government Rights clause is in lieu of, and supersedes, any other FAR, DFAR, or other clause or provision that addresses Government rights in computer software under this License.

9. MISCELLANEOUS.

This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall be governed by the law of the jurisdiction specified in a notice contained within the Original Software (except to the extent applicable law, if any, provides otherwise), excluding such jurisdiction.s conflict-of-law provisions. Any litigation relating to this License shall be subject to the jurisdiction of the courts located in the jurisdiction and venue specified in a notice contained within the Original Software, with the losing party responsible for costs, including, without limitation, court costs and reasonable attorneys. fees and expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License. You agree that You alone are responsible for compliance with the United States export administration regulations (and the export control laws and regulation of any other countries) when You use, distribute or otherwise make available any Covered Software.

10. RESPONSIBILITY FOR CLAIMS.

As between Initial Developer and the Contributors, each party is responsible for claims and damages arising, directly or indirectly, out of its utilization of rights under this License and You agree to work with Initial Developer and Contributors to distribute such responsibility on an equitable basis. Nothing herein is intended or shall be deemed to constitute any admission of liability.

NOTICE PURSUANT TO SECTION 9 OF THE COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL)

The code released under the CDDL shall be governed by the laws of the State of California (excluding conflict-of-law provisions). Any litigation relating to this License shall be subject to the jurisdiction of the Federal Courts of the Northern District of California and the state courts of the State of California, with venue lying in Santa Clara County, California.

The GNU General Public License (GPL) Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited

to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE

PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

One line to give the program's name and a brief idea of what it does.

Copyright (C)

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'. This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License.

Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

"CLASSPATH" EXCEPTION TO THE GPL VERSION 2

Certain source files distributed by Sun Microsystems, Inc. are subject to the following clarification and special exception to the GPL Version 2, but only where Sun has expressly included in the particular source file's header the words

"Sun designates this particular file as subject to the "Classpath" exception as provided by Sun in the License file that accompanied this code."

Linking this library statically or dynamically with other modules is making a combined work based on this library. Thus, the terms and conditions of the GNU General Public License Version 2 cover the whole combination.

As a special exception, the copyright holders of this library give you permission to link this library with independent modules to produce an executable, regardless of the license terms of these independent modules, and to copy and distribute the resulting executable under terms of your choice, provided that you also meet, for each linked independent module, the terms and conditions of the license of that module.? An independent module is a module which is not derived from or based on this library.? If you modify this library, you may extend this exception to your version of the library, but you are not obligated to do so.? If you do not wish to do so, delete this exception statement from your version.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- Redistribution in binary form must reproduct the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of Sun Microsystems, Inc. or the names of contributors may be used to endorse or promote products derived from this software without specific prior written permission.

This software is provided "AS IS," without a warranty of any kind. ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SUN AND ITS LICENSORS SHALL NOT BE LIABLE FOR ANY DAMAGES OR LIABILITIES SUFFERED BY LICENSEE AS A RESULT OF OR RELATING TO USE, MODIFICATION OR DISTRIBUTION OF THE SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL SUN OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE USE OF OR INABILITY TO USE SOFTWARE, EVEN IF SUN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You acknowledge that Software is not designed,licensed or intended for use in the design, construction, operation or maintenance of any nuclear facility.

```
 ========================================================================
 ==  NOTICE file corresponding to the section 4 d of                   ==
 ==  the Apache License, Version 2.0,                                  ==
 ==  in this case for the Apache Ant distribution.                     ==
 ========================================================================

 This product includes software developed by
 The Apache Software Foundation (http://www.apache.org/).

 This product includes also software developed by :
  - the W3C consortium (http://www.w3c.org) ,
  - the SAX project (http://www.saxproject.org)

 Please read the different LICENSE files present in the root directory of
 this distribution.
/*
* The Apache Software License, Version 1.1
*
* Copyright (c) 2001-2003 Ant-Contrib project.  All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
*
* 1. Redistributions of source code must retain the above copyright
*    notice, this list of conditions and the following disclaimer.
*
```

* 2. Redistributions in binary form must reproduce the above copyright
*   notice, this list of conditions and the following disclaimer in
*   the documentation and/or other materials provided with the
*   distribution.
*
* 3. The end-user documentation included with the redistribution, if
*   any, must include the following acknowlegement:
*   "This product includes software developed by the
*   Ant-Contrib project (http://sourceforge.net/projects/ant-contrib)."
*   Alternately, this acknowlegement may appear in the software itself,
*   if and wherever such third-party acknowlegements normally appear.
*
* 4. The name Ant-Contrib must not be used to endorse or promote products
*   derived from this software without prior written permission. For
*   written permission, please contact
*   ant-contrib-developers@lists.sourceforge.net.
*
* 5. Products derived from this software may not be called "Ant-Contrib"
*   nor may "Ant-Contrib" appear in their names without prior written
*   permission of the Ant-Contrib project.
*
* THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED
* WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
* OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
* DISCLAIMED. IN NO EVENT SHALL THE ANT-CONTRIB PROJECT OR ITS
* CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
* LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
* USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
* ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
* OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
* OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
* ==
*/
/*
* $Id: license.txt,v 1.2 2006/04/01 06:01:50 jeffsuttor Exp $
* %W% %E%
*/

COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL) Version 1.0 1.

Definitions.

1.1. Contributor means each individual or entity that creates or
   contributes to the creation of Modifications.

1.2. Contributor Version means the combination of the Original

Software, prior Modifications used by a Contributor (if any), and the Modifications made by that particular Contributor.

1.3. Covered Software means (a) the Original Software, or (b) Modifications, or (c) the combination of files containing Original Software with files containing Modifications, in each case including portions thereof.

1.4. Executable means the Covered Software in any form other than Source Code.

1.5. Initial Developer means the individual or entity that first makes Original Software available under this License.

1.6. Larger Work means a work which combines Covered Software or portions thereof with code not governed by the terms of this License.

1.7. License means this document.

1.8. Licensable means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently acquired, any and all of the rights conveyed herein.

1.9. Modifications means the Source Code and Executable form of any of the following: A. Any file that results from an addition to, deletion from or modification of the contents of a file containing Original Software or previous Modifications; B. Any new file that contains any part of the Original Software or previous Modification; or C. Any new file that is contributed or otherwise made available under the terms of this License.

1.10. Original Software means the Source Code and Executable form of computer software code that is originally released under this License.

1.11. Patent Claims means any patent claim(s), now owned or hereafter acquired, including without limitation, method, process, and apparatus claims, in any patent Licensable by grantor.

1.12. Source Code means (a) the common form of computer software code in which modifications are made and (b) associated documentation included in or with such code.

1.13. You (or Your) means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License. For legal entities, You includes any entity which controls, is controlled by, or is under common control with

You. For purposes of this definition, control means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. License Grants.

2.1. The Initial Developer Grant. Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, the Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license:

(a) under intellectual property rights (other than patent or trademark) Licensable by Initial Developer, to use, reproduce, modify, display, perform, sublicense and distribute the Original Software (or portions thereof), with or without Modifications, and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using or selling of Original Software, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Software (or portions thereof);

(c) The licenses granted in Sections 2.1(a) and (b) are effective on the date Initial Developer first distributes or otherwise makes the Original Software available to a third party under the terms of this License;

(d) Notwithstanding Section 2.1(b) above, no patent license is granted: (1) for code that You delete from the Original Software, or (2) for infringements caused by: (i) the modification of the Original Software, or (ii) the combination of the Original Software with other software or devices.

2.2. Contributor Grant. Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:

(a) under intellectual property rights (other than patent or trademark) Licensable by Contributor to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof), either on an unmodified basis, with other Modifications, as Covered Software and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using, or selling of

Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: (1) Modifications made by that Contributor (or portions thereof); and (2) the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination).

(c) The licenses granted in Sections 2.2(a) and 2.2(b) are effective on the date Contributor first distributes or otherwise makes the Modifications available to a third party.

(d) Notwithstanding Section 2.2(b) above, no patent license is granted: (1) for any code that Contributor has deleted from the Contributor Version; (2) for infringements caused by: (i) third party modifications of Contributor Version, or (ii) the combination of Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or (3) under Patent Claims infringed by Covered Software in the absence of Modifications made by that Contributor.

3. Distribution Obligations.

3.1. Availability of Source Code. Any Covered Software that You distribute or otherwise make available in Executable form must also be made available in Source Code form and that Source Code form must be distributed only under the terms of this License. You must include a copy of this License with every copy of the Source Code form of the Covered Software You distribute or otherwise make available. You must inform recipients of any such Covered Software in Executable form as to how they can obtain such Covered Software in Source Code form in a reasonable manner on or through a medium customarily used for software exchange.

3.2. Modifications. The Modifications that You create or to which You contribute are governed by the terms of this License. You represent that You believe Your Modifications are Your original creation(s) and/or You have sufficient rights to grant the rights conveyed by this License.

3.3. Required Notices. You must include a notice in each of Your Modifications that identifies You as the Contributor of the Modification. You may not remove or alter any copyright, patent or trademark notices contained within the Covered Software, or any notices of licensing or any descriptive text giving attribution to any Contributor or the Initial Developer.

3.4. Application of Additional Terms. You may not offer or impose any

terms on any Covered Software in Source Code form that alters or restricts the applicable version of this License or the recipients rights hereunder. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However, you may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear that any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

3.5. Distribution of Executable Versions. You may distribute the Executable form of the Covered Software under the terms of this License or under the terms of a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable form does not attempt to limit or alter the recipients rights in the Source Code form from the rights set forth in this License. If You distribute the Covered Software in Executable form under a different license, You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.

3.6. Larger Works. You may create a Larger Work by combining Covered Software with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Software.

4. Versions of the License.

4.1. New Versions. Sun Microsystems, Inc. is the initial license steward and may publish revised and/or new versions of this License from time to time. Each version will be given a distinguishing version number. Except as provided in Section 4.3, no one other than the license steward has the right to modify this License.

4.2. Effect of New Versions. You may always continue to use, distribute or otherwise make the Covered Software available under the terms of the version of the License under which You

originally received the Covered Software. If the Initial Developer includes a notice in the Original Software prohibiting it from being distributed or otherwise made available under any subsequent version of the License, You must distribute and make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. Otherwise, You may also choose to use, distribute or otherwise make the Covered Software available under the terms of any subsequent version of the License published by the license steward.

4.3. Modified Versions. When You are an Initial Developer and You want to create a new license for Your Original Software, You may create and use a modified version of this License if You: (a) rename the license and remove any references to the name of the license steward (except to note that the license differs from this License); and (b) otherwise make it clear that the license contains terms which differ from this License.

5. DISCLAIMER OF WARRANTY. COVERED SOFTWARE IS PROVIDED UNDER THIS LICENSE ON AN AS IS BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED SOFTWARE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED SOFTWARE IS WITH YOU. SHOULD ANY COVERED SOFTWARE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED SOFTWARE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

6. TERMINATION.

6.1. This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.

6.2. If You assert a patent infringement claim (excluding declaratory judgment actions) against Initial Developer or a Contributor (the Initial Developer or Contributor against whom You assert such claim is referred to as Participant) alleging that the Participant Software (meaning the Contributor Version where the Participant is a Contributor or the Original Software where the Participant is the Initial Developer) directly or indirectly infringes any patent, then any and all rights granted directly or

indirectly to You by such Participant, the Initial Developer (if the Initial Developer is not the Participant) and all Contributors under Sections 2.1 and/or 2.2 of this License shall, upon 60 days notice from Participant terminate prospectively and automatically at the expiration of such 60 day notice period, unless if within such 60 day period You withdraw Your claim with respect to the Participant Software against such Participant either unilaterally or pursuant to a written agreement with Participant.

6.3. In the event of termination under Sections 6.1 or 6.2 above, all end user licenses that have been validly granted by You or any distributor hereunder prior to termination (excluding licenses granted to You by any distributor) shall survive termination.

7. LIMITATION OF LIABILITY. UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL YOU, THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED SOFTWARE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO ANY PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOST PROFITS, LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTYS NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

8. U.S. GOVERNMENT END USERS. The Covered Software is a commercial item, as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of commercial computer software (as that term is defined at 48 C.F.R. 252.227-7014(a)(1)) and commercial computer software documentation as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Software with only those rights set forth herein. This U.S. Government Rights clause is in lieu of, and supersedes, any other FAR, DFAR, or other clause or provision that addresses Government rights in computer software under this License.

9. MISCELLANEOUS. This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall

be governed by the law of the jurisdiction specified in a notice contained within the Original Software (except to the extent applicable law, if any, provides otherwise), excluding such jurisdictions conflict-of-law provisions. Any litigation relating to this License shall be subject to the jurisdiction of the courts located in the jurisdiction and venue specified in a notice contained within the Original Software, with the losing party responsible for costs, including, without limitation, court costs and reasonable attorneys fees and expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License. You agree that You alone are responsible for compliance with the United States export administration regulations (and the export control laws and regulation of any other countries) when You use, distribute or otherwise make available any Covered Software.

10. RESPONSIBILITY FOR CLAIMS. As between Initial Developer and the Contributors, each party is responsible for claims and damages arising, directly or indirectly, out of its utilization of rights under this License and You agree to work with Initial Developer and Contributors to distribute such responsibility on an equitable basis. Nothing herein is intended or shall be deemed to constitute any admission of liability.

NOTICE PURSUANT TO SECTION 9 OF THE COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL) The code released under the CDDL shall be governed by the laws of the State of California (excluding conflict-of-law provisions). Any litigation relating to this License shall be subject to the jurisdiction of the Federal Courts of the Northern District of California and the state courts of the State of California, with venue lying in Santa Clara County, California.

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object
form, that is based on (or derived from) the Work and for which the
editorial revisions, annotations, elaborations, or other modifications
represent, as a whole, an original work of authorship. For the purposes
of this License, Derivative Works shall not include works that remain
separable from, or merely link (or bind by name) to the interfaces of,
the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity

on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one

of the following places: within a NOTICE text file distributed
as part of the Derivative Works; within the Source form or
documentation, if provided along with the Derivative Works; or,
within a display generated by the Derivative Works, if and
wherever such third-party notices normally appear. The contents
of the NOTICE file are for informational purposes only and
do not modify the License. You may add Your own attribution
notices within Derivative Works that You distribute, alongside
or as an addendum to the NOTICE text from the Work, provided
that such additional attribution notices cannot be construed
as modifying the License.

You may add Your own copyright statement to Your modifications and
may provide additional or different license terms and conditions
for use, reproduction, or distribution of Your modifications, or
for any such Derivative Works as a whole, provided Your use,
reproduction, and distribution of the Work otherwise complies with
the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
any Contribution intentionally submitted for inclusion in the Work
by You to the Licensor shall be under the terms and conditions of
this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify
the terms of any separate license agreement you may have executed
with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
names, trademarks, service marks, or product names of the Licensor,
except as required for reasonable and customary use in describing the
origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
agreed to in writing, Licensor provides the Work (and each
Contributor provides its Contributions) on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied, including, without limitation, any warranties or conditions
of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
PARTICULAR PURPOSE. You are solely responsible for determining the
appropriateness of using or redistributing the Work and assume any
risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
whether in tort (including negligence), contract, or otherwise,
unless required by applicable law (such as deliberate and grossly
negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special,
incidental, or consequential damages of any character arising as a

result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

Copyright (c) 2002-@@YEAR@@ Sun Microsystems, Inc. All Rights Reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

- Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

- Redistribution in binary form must reproduct the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.

Neither the name of Sun Microsystems, Inc. or the names of
contributors may be used to endorse or promote products derived from
this software without specific prior written permission.

This software is provided "AS IS," without a warranty of any kind. ALL
EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES,
INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A
PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SUN AND
ITS LICENSORS SHALL NOT BE LIABLE FOR ANY DAMAGES OR LIABILITIES
SUFFERED BY LICENSEE AS A RESULT OF OR RELATING TO USE, MODIFICATION
OR DISTRIBUTION OF THE SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL
SUN OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA,
OR FOR DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR
PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF
LIABILITY, ARISING OUT OF THE USE OF OR INABILITY TO USE SOFTWARE,
EVEN IF SUN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
The CyberNeko Software License, Version 1.0

(C) Copyright 2002-2005, Andy Clark. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in
the documentation and/or other materials provided with the
distribution.

3. The end-user documentation included with the redistribution,
if any, must include the following acknowledgment:
"This product includes software developed by Andy Clark."
Alternately, this acknowledgment may appear in the software itself,
if and wherever such third-party acknowledgments normally appear.

4. The names "CyberNeko" and "NekoHTML" must not be used to endorse

or promote products derived from this software without prior written permission. For written permission, please contact andyc@cyberneko.net.

5. Products derived from this software may not be called "CyberNeko", nor may "CyberNeko" appear in their name, without prior written permission of the author.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR OTHER CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

==========

This license is based on the Apache Software License, version 1.1.
Copyright 2001 Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, California, 94303, U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in this product. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at http://www.sun.com/patents and one or more additional patents or pending patent applications in the U.S. and other countries. This product is distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third party software, including font technology, is copyrighted and licensed from Sun suppliers. Sun, the Sun logo, and Sun Microsystems are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. This product includes software developed by the Apache Software Foundation (http://www.apache.org/). Federal Acquisitions: Commercial Software - Government Users Subject to Standard License Terms and Conditions.

Copyright 2001 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, Californie 94303 tats-Unis. Tous droits rservs. Distribue par des licences qui en restreignent l'utilisation. Sun Microsystems,

Inc. a les droits de proprit intellectuels relatants la
technologie incorpore dans ce produit. En particulier, et sans la
limitation, ces droits de proprit intellectuels peuvent inclure un
ou plus des brevets amricains numrs http://www.sun.com/patents
et un ou les brevets plus supplmentaires ou les applications de
brevet en attente dans les Etats Unis et les autres pays. Ce produit
ou document est protg par un copyright et distribu avec des
licences qui en restreignent l'utilisation, la copie, la distribution,
et la dcompilation. Aucune partie de ce produit ou document ne peut
tre reproduite sous aucune forme, par quelque moyen que ce soit, sans
l'autorisation pralable et crite de Sun et de ses bailleurs de
licence, s'il y en a. Le logiciel dtenu par des tiers, et qui
comprend la technologie relative aux polices de caractres, est
protg par un copyright et licenci par des fournisseurs de Sun. Sun,
le logo Sun, Sun Microsystems et sont des marques de fabrique ou des
marques d?pos?es de Sun Microsystems, Inc. aux Etats-Unis et dans
d'autres pays. Ce produit inclut le logiciel dvelopp par la base de
Apache Software Foundation (http://www.apache.org/). L'accord du
gouvernement des tats Unis est requis avant l'exportation du produit.
COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL)Version 1.1

1. Definitions.

1.1. "Contributor" means each individual or entity that creates or contributes to the creation of Modifications.

1.2. "Contributor Version" means the combination of the Original Software, prior Modifications used by a Contributor (if any), and the Modifications made by that particular Contributor.

1.3. "Covered Software" means (a) the Original Software, or (b) Modifications, or (c) the combination of files containing Original Software with files containing Modifications, in each case including portions thereof.

1.4. "Executable" means the Covered Software in any form other than Source Code.

1.5. "Initial Developer" means the individual or entity that first makes Original Software available under this License.

1.6. "Larger Work" means a work which combines Covered Software or portions thereof with code not governed by the terms of this License.

1.7. "License" means this document.

1.8. "Licensable" means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently acquired, any and all of the rights conveyed herein.

1.9. "Modifications" means the Source Code and Executable form of any of the following:

A. Any file that results from an addition to, deletion from or modification of the contents of a file containing Original Software or previous Modifications;

B. Any new file that contains any part of the Original Software or previous Modification; or

C. Any new file that is contributed or otherwise made available under the terms of this License.

1.10. "Original Software" means the Source Code and Executable form of computer software code that is originally released under this License.

1.11. "Patent Claims" means any patent claim(s), now owned or hereafter acquired, including without limitation, method, process, and apparatus claims, in any patent Licensable by grantor.

1.12. "Source Code" means (a) the common form of computer software code in which modifications are made and (b) associated documentation included in or with such code.

1.13. "You" (or "Your") means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License. For legal entities, "You" includes any entity which controls, is controlled by, or is under common control with You. For purposes of this definition, "control" means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. License Grants.

2.1. The Initial Developer Grant.

Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, the Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license:

(a) under intellectual property rights (other than patent or trademark) Licensable by Initial Developer, to use, reproduce, modify, display, perform, sublicense and distribute the Original Software (or portions thereof), with or without Modifications, and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using or selling of Original Software, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Software (or portions thereof).

(c) The licenses granted in Sections 2.1(a) and (b) are effective on the date Initial Developer first distributes or otherwise makes the Original Software available to a third party under the terms of this License.

(d) Notwithstanding Section 2.1(b) above, no patent license is granted: (1) for code that You delete from the Original Software, or (2) for infringements caused by: (i) the modification of the Original Software, or (ii) the combination of the Original Software with other software or devices.

2.2. Contributor Grant.

Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:

(a) under intellectual property rights (other than patent or trademark) Licensable by Contributor to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions

thereof), either on an unmodified basis, with other Modifications, as Covered Software and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: (1) Modifications made by that Contributor (or portions thereof); and (2) the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination).

(c) The licenses granted in Sections 2.2(a) and 2.2(b) are effective on the date Contributor first distributes or otherwise makes the Modifications available to a third party.

(d) Notwithstanding Section 2.2(b) above, no patent license is granted: (1) for any code that Contributor has deleted from the Contributor Version; (2) for infringements caused by: (i) third party modifications of Contributor Version, or (ii) the combination of Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or (3) under Patent Claims infringed by Covered Software in the absence of Modifications made by that Contributor.

3. Distribution Obligations.

3.1. Availability of Source Code.

Any Covered Software that You distribute or otherwise make available in Executable form must also be made available in Source Code form and that Source Code form must be distributed only under the terms of this License. You must include a copy of this License with every copy of the Source Code form of the Covered Software You distribute or otherwise make available. You must inform recipients of any such Covered Software in Executable form as to how they can obtain such Covered Software in Source Code form in a reasonable manner on or through a medium customarily used for software exchange.

3.2. Modifications.

The Modifications that You create or to which You contribute are governed by the terms of this License. You represent that You believe Your Modifications are Your original creation(s) and/or You have sufficient rights to grant the rights conveyed by this License.

3.3. Required Notices.

You must include a notice in each of Your Modifications that identifies You as the Contributor of the Modification. You may not remove or alter any copyright, patent or trademark notices contained within the Covered Software, or any notices of licensing or any descriptive text giving attribution to any Contributor or the Initial Developer.

3.4. Application of Additional Terms.

You may not offer or impose any terms on any Covered Software in Source Code form that alters or restricts the applicable version of this License or the recipients' rights hereunder. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However, you may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must

make it absolutely clear that any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

3.5. Distribution of Executable Versions.

You may distribute the Executable form of the Covered Software under the terms of this License or under the terms of a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable form does not attempt to limit or alter the recipient's rights in the Source Code form from the rights set forth in this License. If You distribute the Covered Software in Executable form under a different license, You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.

3.6. Larger Works.

You may create a Larger Work by combining Covered Software with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Software.

4. Versions of the License.

4.1. New Versions.

Oracle is the initial license steward and may publish revised and/or new versions of this License from time to time. Each version will be given a distinguishing version number. Except as provided in Section 4.3, no one other than the license steward has the right to modify this License.

4.2. Effect of New Versions.

You may always continue to use, distribute or otherwise make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. If the Initial Developer includes a notice in the Original Software prohibiting it from being distributed or otherwise made available under any subsequent version of the License, You must distribute and make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. Otherwise, You may also choose to use, distribute or otherwise make the Covered Software available under the terms of any subsequent version of the License published by the license steward.

4.3. Modified Versions.

When You are an Initial Developer and You want to create a new license for Your Original Software, You may create and use a modified version of this License if You: (a) rename the license and remove any references to the name of the license steward (except to note that the license differs from this License); and (b) otherwise make it clear that the license contains terms which differ from this License.

5. DISCLAIMER OF WARRANTY.

COVERED SOFTWARE IS PROVIDED UNDER THIS LICENSE ON AN "AS IS" BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED SOFTWARE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED SOFTWARE IS WITH YOU. SHOULD ANY COVERED SOFTWARE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED SOFTWARE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

6. TERMINATION.

6.1. This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.

6.2. If You assert a patent infringement claim (excluding declaratory judgment actions) against Initial Developer or a Contributor (the Initial Developer or Contributor against whom You assert such claim is referred to as "Participant") alleging that the Participant Software (meaning the Contributor Version where the Participant is a Contributor or the Original Software where the Participant is the Initial Developer) directly or indirectly infringes any patent, then any and all rights granted directly or indirectly to You by such Participant, the Initial Developer (if the Initial Developer is not the Participant) and all Contributors under Sections 2.1 and/or 2.2 of this License shall, upon 60 days notice from Participant terminate prospectively and automatically at the expiration of such 60 day notice period, unless if within such 60 day period You withdraw Your claim with respect to the Participant Software against such Participant either unilaterally or pursuant to a written agreement with Participant.

6.3. If You assert a patent infringement claim against Participant alleging that the Participant Software directly or indirectly infringes any patent where such claim is resolved (such as by license or settlement) prior to the initiation of patent infringement litigation, then the reasonable value of the licenses granted by such Participant under Sections 2.1 or 2.2 shall be taken into account in determining the amount or value of any payment or license.

6.4. In the event of termination under Sections 6.1 or 6.2 above, all end user licenses that have been validly granted by You or any distributor hereunder prior to termination (excluding licenses granted to You by any distributor) shall survive termination.

7. LIMITATION OF LIABILITY.

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL YOU, THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED SOFTWARE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO ANY PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTY'S NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR

CONSEQUENTIAL DAMAGES, SO THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

8. U.S. GOVERNMENT END USERS.

The Covered Software is a "commercial item," as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of "commercial computer software" (as that term is defined at 48 C.F.R. ? 252.227-7014(a)(1)) and "commercial computer software documentation" as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Software with only those rights set forth herein. This U.S. Government Rights clause is in lieu of, and supersedes, any other FAR, DFAR, or other clause or provision that addresses Government rights in computer software under this License.

9. MISCELLANEOUS.

This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall be governed by the law of the jurisdiction specified in a notice contained within the Original Software (except to the extent applicable law, if any, provides otherwise), excluding such jurisdiction's conflict-of-law provisions. Any litigation relating to this License shall be subject to the jurisdiction of the courts located in the jurisdiction and venue specified in a notice contained within the Original Software, with the losing party responsible for costs, including, without limitation, court costs and reasonable attorneys' fees and expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License. You agree that You alone are responsible for compliance with the United States export administration regulations (and the export control laws and regulation of any other countries) when You use, distribute or otherwise make available any Covered Software.

10. RESPONSIBILITY FOR CLAIMS.

As between Initial Developer and the Contributors, each party is responsible for claims and damages arising, directly or indirectly, out of its utilization of rights under this License and You agree to work with Initial Developer and Contributors to distribute such responsibility on an equitable basis. Nothing herein is intended or shall be deemed to constitute any admission of liability.

NOTICE PURSUANT TO SECTION 9 OF THE COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL)

The code released under the CDDL shall be governed by the laws of the State of California (excluding conflict-of-law provisions). Any litigation relating to this License shall be subject to the jurisdiction of the Federal Courts of the Northern District of California and the state courts of the State of California, with venue lying in Santa Clara County, California.

The GNU General Public License (GPL) Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications

and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only

way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE

PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

One line to give the program's name and a brief idea of what it does.

Copyright (C)

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'. This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

"CLASSPATH" EXCEPTION TO THE GPL VERSION 2

Certain source files distributed by Oracle are subject to the following clarification and special exception to the GPL Version 2, but only where Oracle has expressly included in the particular source file's header the words "Oracle designates this particular file as subject to the "Classpath" exception as provided by Oracle in the License file that accompanied this code."

Linking this library statically or dynamically with other modules is making a combined work based on this library. Thus, the terms and conditions of the GNU General Public License Version 2 cover the whole combination.

As a special exception, the copyright holders of this library give you permission to link this library with independent modules to produce an executable, regardless of the license terms of these independent modules, and to copy and distribute the resulting executable under terms of your choice, provided that you also meet, for each linked independent module, the terms and conditions of the license of that module. An independent module is a module which is not derived from or based on this library. If you modify this library, you may extend this exception to your version of the library, but you are not obligated to do so. If you do not wish to do so, delete this exception statement from your version.

```
/*
*                                 Apache License
*                           Version 2.0, January 2004
*                        http://www.apache.org/licenses/
*
*   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION
*
*   1. Definitions.
*
*      "License" shall mean the terms and conditions for use, reproduction,
*      and distribution as defined by Sections 1 through 9 of this document.
*
*      "Licensor" shall mean the copyright owner or entity authorized by
*      the copyright owner that is granting the License.
*
*      "Legal Entity" shall mean the union of the acting entity and all
*      other entities that control, are controlled by, or are under common
*      control with that entity. For the purposes of this definition,
*      "control" means (i) the power, direct or indirect, to cause the
```

```
*      direction or management of such entity, whether by contract or
*      otherwise, or (ii) ownership of fifty percent (50%) or more of the
*      outstanding shares, or (iii) beneficial ownership of such entity.
*
*      "You" (or "Your") shall mean an individual or Legal Entity
*      exercising permissions granted by this License.
*
*      "Source" form shall mean the preferred form for making modifications,
*      including but not limited to software source code, documentation
*      source, and configuration files.
*
*      "Object" form shall mean any form resulting from mechanical
*      transformation or translation of a Source form, including but
*      not limited to compiled object code, generated documentation,
*      and conversions to other media types.
*
*      "Work" shall mean the work of authorship, whether in Source or
*      Object form, made available under the License, as indicated by a
*      copyright notice that is included in or attached to the work
*      (an example is provided in the Appendix below).
*
*      "Derivative Works" shall mean any work, whether in Source or Object
*      form, that is based on (or derived from) the Work and for which the
*      editorial revisions, annotations, elaborations, or other modifications
*      represent, as a whole, an original work of authorship. For the purposes
*      of this License, Derivative Works shall not include works that remain
*      separable from, or merely link (or bind by name) to the interfaces of,
*      the Work and Derivative Works thereof.
*
*      "Contribution" shall mean any work of authorship, including
*      the original version of the Work and any modifications or additions
*      to that Work or Derivative Works thereof, that is intentionally
*      submitted to Licensor for inclusion in the Work by the copyright owner
*      or by an individual or Legal Entity authorized to submit on behalf of
*      the copyright owner. For the purposes of this definition, "submitted"
*      means any form of electronic, verbal, or written communication sent
*      to the Licensor or its representatives, including but not limited to
*      communication on electronic mailing lists, source code control systems,
*      and issue tracking systems that are managed by, or on behalf of, the
*      Licensor for the purpose of discussing and improving the Work, but
*      excluding communication that is conspicuously marked or otherwise
*      designated in writing by the copyright owner as "Not a Contribution."
*
*      "Contributor" shall mean Licensor and any individual or Legal Entity
*      on behalf of whom a Contribution has been received by Licensor and
*      subsequently incorporated within the Work.
*
*   2. Grant of Copyright License. Subject to the terms and conditions of
```

```
*     this License, each Contributor hereby grants to You a perpetual,
*     worldwide, non-exclusive, no-charge, royalty-free, irrevocable
*     copyright license to reproduce, prepare Derivative Works of,
*     publicly display, publicly perform, sublicense, and distribute the
*     Work and such Derivative Works in Source or Object form.
*
*  3. Grant of Patent License. Subject to the terms and conditions of
*     this License, each Contributor hereby grants to You a perpetual,
*     worldwide, non-exclusive, no-charge, royalty-free, irrevocable
*     (except as stated in this section) patent license to make, have made,
*     use, offer to sell, sell, import, and otherwise transfer the Work,
*     where such license applies only to those patent claims licensable
*     by such Contributor that are necessarily infringed by their
*     Contribution(s) alone or by combination of their Contribution(s)
*     with the Work to which such Contribution(s) was submitted. If You
*     institute patent litigation against any entity (including a
*     cross-claim or counterclaim in a lawsuit) alleging that the Work
*     or a Contribution incorporated within the Work constitutes direct
*     or contributory patent infringement, then any patent licenses
*     granted to You under this License for that Work shall terminate
*     as of the date such litigation is filed.
*
*  4. Redistribution. You may reproduce and distribute copies of the
*     Work or Derivative Works thereof in any medium, with or without
*     modifications, and in Source or Object form, provided that You
*     meet the following conditions:
*
*     (a) You must give any other recipients of the Work or
*         Derivative Works a copy of this License; and
*
*     (b) You must cause any modified files to carry prominent notices
*         stating that You changed the files; and
*
*     (c) You must retain, in the Source form of any Derivative Works
*         that You distribute, all copyright, patent, trademark, and
*         attribution notices from the Source form of the Work,
*         excluding those notices that do not pertain to any part of
*         the Derivative Works; and
*
*     (d) If the Work includes a "NOTICE" text file as part of its
*         distribution, then any Derivative Works that You distribute must
*         include a readable copy of the attribution notices contained
*         within such NOTICE file, excluding those notices that do not
*         pertain to any part of the Derivative Works, in at least one
*         of the following places: within a NOTICE text file distributed
*         as part of the Derivative Works; within the Source form or
*         documentation, if provided along with the Derivative Works; or,
*         within a display generated by the Derivative Works, if and
```

```
*       wherever such third-party notices normally appear. The contents
*       of the NOTICE file are for informational purposes only and
*       do not modify the License. You may add Your own attribution
*       notices within Derivative Works that You distribute, alongside
*       or as an addendum to the NOTICE text from the Work, provided
*       that such additional attribution notices cannot be construed
*       as modifying the License.
*
*     You may add Your own copyright statement to Your modifications and
*     may provide additional or different license terms and conditions
*     for use, reproduction, or distribution of Your modifications, or
*     for any such Derivative Works as a whole, provided Your use,
*     reproduction, and distribution of the Work otherwise complies with
*     the conditions stated in this License.
*
*   5. Submission of Contributions. Unless You explicitly state otherwise,
*     any Contribution intentionally submitted for inclusion in the Work
*     by You to the Licensor shall be under the terms and conditions of
*     this License, without any additional terms or conditions.
*     Notwithstanding the above, nothing herein shall supersede or modify
*     the terms of any separate license agreement you may have executed
*     with Licensor regarding such Contributions.
*
*   6. Trademarks. This License does not grant permission to use the trade
*     names, trademarks, service marks, or product names of the Licensor,
*     except as required for reasonable and customary use in describing the
*     origin of the Work and reproducing the content of the NOTICE file.
*
*   7. Disclaimer of Warranty. Unless required by applicable law or
*     agreed to in writing, Licensor provides the Work (and each
*     Contributor provides its Contributions) on an "AS IS" BASIS,
*     WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
*     implied, including, without limitation, any warranties or conditions
*     of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
*     PARTICULAR PURPOSE. You are solely responsible for determining the
*     appropriateness of using or redistributing the Work and assume any
*     risks associated with Your exercise of permissions under this License.
*
*   8. Limitation of Liability. In no event and under no legal theory,
*     whether in tort (including negligence), contract, or otherwise,
*     unless required by applicable law (such as deliberate and grossly
*     negligent acts) or agreed to in writing, shall any Contributor be
*     liable to You for damages, including any direct, indirect, special,
*     incidental, or consequential damages of any character arising as a
*     result of this License or out of the use or inability to use the
*     Work (including but not limited to damages for loss of goodwill,
*     work stoppage, computer failure or malfunction, or any and all
*     other commercial damages or losses), even if such Contributor
```

```
*     has been advised of the possibility of such damages.
*
*   9. Accepting Warranty or Additional Liability. While redistributing
*     the Work or Derivative Works thereof, You may choose to offer,
*     and charge a fee for, acceptance of support, warranty, indemnity,
*     or other liability obligations and/or rights consistent with this
*     License. However, in accepting such obligations, You may act only
*     on Your own behalf and on Your sole responsibility, not on behalf
*     of any other Contributor, and only if You agree to indemnify,
*     defend, and hold each Contributor harmless for any liability
*     incurred by, or claims asserted against, such Contributor by reason
*     of your accepting any such warranty or additional liability.
*
*   END OF TERMS AND CONDITIONS
*
*   APPENDIX: How to apply the Apache License to your work.
*
*     To apply the Apache License to your work, attach the following
*     boilerplate notice, with the fields enclosed by brackets "[]"
*     replaced with your own identifying information. (Don't include
*     the brackets!)  The text should be enclosed in the appropriate
*     comment syntax for the file format. We also recommend that a
*     file or class name and description of purpose be included on the
*     same "printed page" as the copyright notice for easier
*     identification within third-party archives.
*
*   Copyright [yyyy] [name of copyright owner]
*
*   Licensed under the Apache License, Version 2.0 (the "License");
*   you may not use this file except in compliance with the License.
*   You may obtain a copy of the License at
*
*       http://www.apache.org/licenses/LICENSE-2.0
*
*   Unless required by applicable law or agreed to in writing, software
*   distributed under the License is distributed on an "AS IS" BASIS,
*   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
*   See the License for the specific language governing permissions and
*   limitations under the License.
*/
/* ====================================================================
* The Apache Software License, Version 1.1
*
* Copyright (c) 2001 The Apache Software Foundation.  All rights
* reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
```

* are met:
*
* 1. Redistributions of source code must retain the above copyright
*   notice, this list of conditions and the following disclaimer.
*
* 2. Redistributions in binary form must reproduce the above copyright
*   notice, this list of conditions and the following disclaimer in
*   the documentation and/or other materials provided with the
*   distribution.
*
* 3. The end-user documentation included with the redistribution,
*   if any, must include the following acknowledgment:
*   "This product includes software developed by the
*   Apache Software Foundation (http://www.apache.org/)."
*   Alternately, this acknowledgment may appear in the software itself,
*   if and wherever such third-party acknowledgments normally appear.
*
* 4. The names "Apache" and "Apache Software Foundation" and
*   "Apache BCEL" must not be used to endorse or promote products
*   derived from this software without prior written permission. For
*   written permission, please contact apache@apache.org.
*
* 5. Products derived from this software may not be called "Apache",
*   "Apache BCEL", nor may "Apache" appear in their name, without
*   prior written permission of the Apache Software Foundation.
*
* THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED
* WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
* OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
* DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR
* ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
* LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
* USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
* ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
* OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
* OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
* ==
*
* This software consists of voluntary contributions made by many
* individuals on behalf of the Apache Software Foundation. For more
* information on the Apache Software Foundation, please see
* <http://www.apache.org/>.
*/
Copyright (c) 2000-2003 Daisuke Okajima and Kohsuke Kawaguchi.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.

3. The end-user documentation included with the redistribution, if
any, must include the following acknowledgment:

"This product includes software developed by Daisuke Okajima
and Kohsuke Kawaguchi (http://relaxngcc.sf.net/)."

Alternately, this acknowledgment may appear in the software itself,
if and wherever such third-party acknowledgments normally appear.

4. The names of the copyright holders must not be used to endorse or
promote products derived from this software without prior written
permission. For written permission, please contact the copyright
holders.

5. Products derived from this software may not be called "RELAXNGCC",
nor may "RELAXNGCC" appear in their name, without prior written
permission of the copyright holders.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED
WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS
BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY,
OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
DO NOT TRANSLATE OR LOCALIZE

****

%%The following software may be included in this product:
XML-NamespaceSupport

Use of any of this software is governed by the terms of the license below:

The "Artistic License"

Preamble

The intent of this document is to state the conditions under which a Package may be copied, such that the Copyright Holder maintains some semblance of artistic control over the development of the package, while giving the users of the package the right to use and distribute the Package in a more-or-less customary fashion, plus the right to make reasonable modifications.

Definitions:

"Package" refers to the collection of files distributed by the Copyright Holder, and derivatives of that collection of files created through textual modification.

"Standard Version" refers to such a Package if it has not been modified, or has been modified in accordance with the wishes of the Copyright Holder as specified below.

"Copyright Holder" is whoever is named in the copyright or copyrights for the package.

"You" is you, if you're thinking about copying or distributing this Package.

"Reasonable copying fee" is whatever you can justify on the basis of media cost, duplication charges, time of people involved, and so on. (You will not be required to justify it to the Copyright Holder, but only to the computing community at large as a market that must bear the fee.)

"Freely Available" means that no fee is charged for the item itself, though there may be fees involved in handling the item. It also means that recipients of the item may redistribute it under the same conditions they received it.

1. You may make and give away verbatim copies of the source form of the Standard Version of this Package without restriction, provided that you duplicate all of the original copyright notices and associated disclaimers.

2. You may apply bug fixes, portability fixes and other modifications derived from the Public Domain or from the Copyright Holder. A Package modified in such a way shall still be considered the Standard Version.

3. You may otherwise modify your copy of this Package in any way, provided

that you insert a prominent notice in each changed file stating how and when you changed that file, and provided that you do at least ONE of the following:

a) place your modifications in the Public Domain or otherwise make them Freely Available, such as by posting said modifications to Usenet or an equivalent medium, or placing the modifications on a major archive site such as uunet.uu.net, or by allowing the Copyright Holder to include your modifications in the Standard Version of the Package.

b) use the modified Package only within your corporation or organization.

c) rename any non-standard executables so the names do not conflict with standard executables, which must also be provided, and provide a separate manual page for each non-standard executable that clearly documents how it differs from the Standard Version.

d) make other distribution arrangements with the Copyright Holder.

4. You may distribute the programs of this Package in object code or executable form, provided that you do at least ONE of the following:

a) distribute a Standard Version of the executables and library files, together with instructions (in the manual page or equivalent) on where to get the Standard Version.

b) accompany the distribution with the machine-readable source of the Package with your modifications.

c) give non-standard executables non-standard names, and clearly document the differences in manual pages (or equivalent), together with instructions on where to get the Standard Version.

d) make other distribution arrangements with the Copyright Holder.

5. You may charge a reasonable copying fee for any distribution of this Package. You may charge any fee you choose for support of this Package. You may not charge a fee for this Package itself. However, you may distribute this Package in aggregate with other (possibly commercial) programs as part of a larger (possibly commercial) software distribution provided that you do not advertise this Package as a product of your own. You may embed this Package's interpreter within an executable of yours (by linking); this shall be construed as a mere form of aggregation, provided that the complete Standard Version of the interpreter is so embedded.

6. The scripts and library files supplied as input to or produced as output from the programs of this Package do not automatically fall

under the copyright of this Package, but belong to whoever generated
them, and may be sold commercially, and may be aggregated with this
Package. If such scripts or library files are aggregated with this
Package via the so-called "undump" or "unexec" methods of producing a
binary executable image, then distribution of such an image shall
neither be construed as a distribution of this Package nor shall it
fall under the restrictions of Paragraphs 3 and 4, provided that you do
not represent such an executable image as a Standard Version of this
Package.

7. C subroutines (or comparably compiled subroutines in other
languages) supplied by you and linked into this Package in order to
emulate subroutines and variables of the language defined by this
Package shall not be considered part of this Package, but are the
equivalent of input as in Paragraph 6, provided these subroutines do
not change the language in any way that would cause it to fail the
regression tests for the language.

8. Aggregation of this Package with a commercial distribution is always
permitted provided that the use of this Package is embedded; that is,
when no overt attempt is made to make this Package's interfaces visible
to the end user of the commercial distribution. Such use shall not be
construed as a distribution of this Package.

9. The name of the Copyright Holder may not be used to endorse or promote
products derived from this software without specific prior written permission.

10. THIS PACKAGE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR
IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The End

Additional License(s)

Copyright (c) 2001-2005 Robin Berjon. All rights reserved.

****

%%The following software may be included in this product:
iso-relax.jar

Use of any of this software is governed by the terms of the license below:

The MIT License

Copyright (c)

Permission is hereby granted, free of charge, to
any person obtaining a copy of this software and
associated documentation files (the "Software"),
to deal in the Software without restriction,
including without limitation the rights to use,
copy, modify, merge, publish, distribute,
sublicense, and/or sell copies of the Software,
and to permit persons to whom the Software is
furnished to do so, subject to the following
conditions:

The above copyright notice and this permission
notice shall be included in all copies or
substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT
WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR
PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR
ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

Copyright 200

Additional License(s)

"copyright" and "license" results in the following hits:
> * The above copyright notice and this permission notice shall be included
> * distribute, sublicense, and/or sell copies of the Software, and to

GNU, GPL, LGPL reveals no hit. "?" hits a lot of things but none of them
are relevant to the licensing terms.

****

%%The following software may be included in this product:
relaxngDatatype.jar

Use of any of this software is governed by the terms of the license below:

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the names of the copyright holders nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Additional License(s)

Got the following hits. No hit for GNU, GPL, LGPL.

> Redistributions of source code must retain the above copyright

> Neither the names of the copyright holders nor the names of its

> this license is the BSD license.

****

%%The following software may be included in this product:
RELAX NG Object Model/Parser

Use of any of this software is governed by the terms of the license below:

The MIT License

Copyright (c)

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so,
subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Additional License(s)

See https://rngom.dev.java.net/doc/index.html

****

%%The following software may be included in this product:
RelaxNGCC

Use of any of this software is governed by the terms of the license below:

Copyright (c) 2000-2003 Daisuke Okajima and Kohsuke Kawaguchi.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.

3. The end-user documentation included with the redistribution, if

any, must include the following acknowledgment:

"This product includes software developed by Daisuke Okajima
and Kohsuke Kawaguchi (http://relaxngcc.sf.net/)."

Alternately, this acknowledgment may appear in the software itself,
if and wherever such third-party acknowledgments normally appear.

4. The names of the copyright holders must not be used to endorse or
promote products derived from this software without prior written
permission. For written permission, please contact the copyright
holders.

5. Products derived from this software may not be called "RELAXNGCC",
nor may "RELAXNGCC" appear in their name, without prior written
permission of the copyright holders.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED
WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS
BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY,
OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Additional License(s)

None found

****

%%The following software may be included in this product:
XML Resolver library

Use of any of this software is governed by the terms of the license below:

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems,

and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
whether in tort (including negligence), contract, or otherwise,
unless required by applicable law (such as deliberate and grossly
negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special,
incidental, or consequential damages of any character arising as a
result of this License or out of the use or inability to use the
Work (including but not limited to damages for loss of goodwill,
work stoppage, computer failure or malfunction, or any and all
other commercial damages or losses), even if such Contributor
has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
the Work or Derivative Works thereof, You may choose to offer,
and charge a fee for, acceptance of support, warranty, indemnity,
or other liability obligations and/or rights consistent with this
License. However, in accepting such obligations, You may act only
on Your own behalf and on Your sole responsibility, not on behalf
of any other Contributor, and only if You agree to indemnify,
defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following
boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and

limitations under the License.

Additional License(s)

All occurrences of copyright, license and (c) refer to the Apache 1.1 license.

No occurrences of GNU, GPL, LGPL.

****

%%The following software may be included in this product:
Stax API (only)

Use of any of this software is governed by the terms of the license below:

Streaming API for XML (JSR-173) Specification
Reference Implementation
License Agreement

READ THE TERMS OF THIS (THE "AGREEMENT") CAREFULLY BEFORE VIEWING OR USING THE SOFTWARE LICENS
ED HEREUNDER. BY VIEWING OR USING THE SOFTWARE, YOU AGREE TO THE TERMS OF THIS AGREEMENT. IF
YOU ARE ACCESSING THE SOFTWARE ELECTRONICALLY, INDICATE YOUR ACCEPTANCE OF THESE
TERMS BY SELE
CTING THE "ACCEPT" BUTTON AT THE END OF THIS AGREEMENT. IF YOU DO NOT AGREE TO ALL THESE TERMS
, PROMPTLY RETURN THE UNUSED SOFTWARE TO ORIGINAL CONTRIBUTOR, DEFINED HEREIN.

1.0 DEFINITIONS.

1.1. "BEA" means BEA Systems, Inc., the licensor of the Original Code.

1.2. "Contributor" means BEA and each entity that creates or contributes to the
creation of Mo
difications.

1.3. "Covered Code" means the Original Code or Modifications or the combination
of the Origina
l Code and Modifications, in each case including portions thereof and
corresponding documentat
ion released with the source code.

1.4. "Executable" means Covered Code in any form other than Source Code.

1.5. "FCS" means first commercial shipment of a product.

1.6. "Modifications" means any addition to or deletion from the substance or structure of eith
er the Original Code or any previous Modifications. When Covered Code is released as a series
of files, a Modification is:

(a) Any addition to or deletion from the contents of a file containing Original Code or previ
ous Modifications.

(b) Any new file that contains any part of the Original Code or previous Modifications.

1.7. "Original Code" means Source Code of computer software code Reference Implementation.

1.8. "Patent Claims" means any patent claim(s), now owned or hereafter acquired, including wit
hout limitation, method, process, and apparatus claims, in any patent for which the grantor ha
s the right to grant a license.

1.9. "Reference Implementation" means the prototype or "proof of concept" implementation of
the Specification developed and made available for license by or on behalf of BEA.

1.10. "Source Code" means the preferred form of the Covered Code for making modifications to i
t, including all modules it contains, plus any associated documentation, interface definition
files, scripts used to control compilation and installation of an Executable, or source code d
ifferential comparisons against either the Original Code or another well known, available Cove
red Code of the Contributor's choice.

1.11. "Specification" means the written specification for the Streaming API for XML , Java te
chnology developed pursuant to the Java Community Process.
1.12. "Technology Compatibility Kit" or "TCK" means the documentation, testing tools and test
suites associated with the Specification as may be revised by BEA from time to time, that is p
rovided so that an implementer of the Specification may determine if its implementation is co
mpliant with the Specification.

1.13. "You" (or "Your") means an individual or a legal entity exercising rights

under, and com
plying with all of the terms of, this Agreement or a future version of this Agreement issued u
nder Section 6.1. For legal entities, "You" includes any entity which controls, is controlled
by, or is under common control with You. For purposes of this definition, "control" means (a)
the power, direct or indirect, to cause the direction or management of such entity, whether by
contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding s
hares or beneficial ownership of such entity.

2.0 SOURCE CODE LICENSE.

2.1. Copyright Grant. Subject to the terms of this Agreement, each Contributor hereby grants
You a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare derivativ
e works of, publicly display, publicly perform, distribute and sublicense the Covered Code of
such Contributor, if any, and such derivative works, in Source Code and Executable form.

2.2. Patent Grant. Subject to the terms of this Agreement, each Contributor hereby grants Yo
u a non-exclusive, worldwide, royalty-free patent license under the Patent Claims to make, use
, sell, offer to sell, import and otherwise transfer the Covered Code prepared and provided by
such Contributor, if any, in Source Code and Executable form. This patent license shall apply
to the Covered Code if, at the time a Modification is added by the Contributor, such addition
of the Modification causes such combination to be covered by the Patent Claims. The patent li
cense shall not apply to any other combinations which include the Modification.

2.3. Conditions to Grants. You understand that although each Contributor grants the licenses
to the Covered Code prepared by it, no assurances are provided by any Contributor that the Co
vered Code does not infringe the patent or other intellectual property rights of any other ent
ity. Each Contributor disclaims any liability to You for claims brought by any other entity ba
sed on infringement of intellectual property rights or otherwise. As a condition to exercising

the rights and licenses granted hereunder, You hereby assume sole responsibility to secure an
y other intellectual property rights needed, if any. For example, if a third party patent lice
nse is required to allow You to distribute Covered Code, it is Your responsibility to acquire
that license before distributing such code.

2.4. Contributors' Representation. Each Contributor represents that to its knowledge it has
sufficient copyright rights in the Covered Code it provides , if any, to grant the copyright l
icense set forth in this Agreement.

3.0 DISTRIBUION RESTRICTIONS.

3.1. Application of Agreement.

The Modifications which You create or to which You contribute are governed by the terms of thi
s Agreement, including without limitation Section 2.0. The Source Code version of Covered Code
may be distributed only under the terms of this Agreement or a future version of this Agreeme
nt released under Section 6.1, and You must include a copy of this Agreement with every copy o
f the Source Code You distribute. You may not offer or impose any terms on any Source Code ver
sion that alters or restricts the applicable version of this Agreement or the recipients' righ
ts hereunder. However, You may include an additional document offering the additional rights d
escribed in Section 3.3.

3.2. Description of Modifications.

You must cause all Covered Code to which You contribute to contain a file documenting the chan
ges You made to create that Covered Code and the date of any change. You must include a promin
ent statement that the Modification is derived, directly or indirectly, from Original Code pro
vided by BEA and including the name of BEA in (a) the Source Code, and (b) in any notice in an
Executable version or related documentation in which You describe the origin or ownership of
the Covered Code.

****

%%The following software may be included in this product:
XMLWriter

Use of any of this software is governed by the terms of the license below:

XMLWriter IS FREE

I hereby abandon any property rights to XMLWriter 0.1, and release all
of the XMLWriter 0.1 source code, compiled code, and documentation
contained in this distribution into the Public Domain. XMLWriter
comes with NO WARRANTY or guarantee of fitness for any purpose.

David Megginson
david@megginson.com
2000-04-19

Additional License(s)

I grep-ed the source. GNU and GPL has no hits, '?' yieds 11 hits but none of them
are license related. "copyright" and "license" yield no hits either.

****

```
/*
* The Apache Software License, Version 1.1
*
*
* Copyright (c) 1999-2004 The Apache Software Foundation.  All rights
* reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
*
* 1. Redistributions of source code must retain the above copyright
*    notice, this list of conditions and the following disclaimer.
*
* 2. Redistributions in binary form must reproduce the above copyright
*    notice, this list of conditions and the following disclaimer in
*    the documentation and/or other materials provided with the
*    distribution.
*
* 3. The end-user documentation included with the redistribution,
*    if any, must include the following acknowledgment:
*       "This product includes software developed by the
```

* Apache Software Foundation (http://www.apache.org/)."
* Alternately, this acknowledgment may appear in the software itself,
* if and wherever such third-party acknowledgments normally appear.
*
* 4. The names "Xerces" and "Apache Software Foundation" must
* not be used to endorse or promote products derived from this
* software without prior written permission. For written
* permission, please contact apache@apache.org.
*
* 5. Products derived from this software may not be called "Apache",
* nor may "Apache" appear in their name, without prior written
* permission of the Apache Software Foundation.
*
* THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED
* WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
* OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
* DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR
* ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
* LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
* USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
* ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
* OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
* OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
* ==
*
* This software consists of voluntary contributions made by many
* individuals on behalf of the Apache Software Foundation and was
* originally based on software copyright (c) 1999, International
* Business Machines, Inc., http://www.ibm.com. For more
* information on the Apache Software Foundation, please see
* <http://www.apache.org/>.
*/
COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL) Version 1.0

1. Definitions.

1.1. Contributor means each individual or entity that creates or contributes to the creation of Modifications.

1.2. Contributor Version means the combination of the Original Software, prior Modifications used by a Contributor (if any), and the Modifications made by that particular Contributor.

1.3. Covered Software means (a) the Original Software, or (b) Modifications, or (c) the combination of files containing Original Software with files containing Modifications, in each case including portions thereof.

1.4. Executable means the Covered Software in any form other than Source Code.

1.5. Initial Developer means the individual or entity that first makes Original Software available under this License.

1.6. Larger Work means a work which combines Covered Software or portions thereof with code not governed by the terms of this License.

1.7. License means this document.

1.8. Licensable means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently acquired, any and all of the rights conveyed herein.

1.9. Modifications means the Source Code and Executable form of any of the following:

A. Any file that results from an addition to, deletion from or modification of the contents of a file containing Original Software or previous Modifications;

B. Any new file that contains any part of the Original Software or previous Modification; or

C. Any new file that is contributed or otherwise made available under the terms of this License.

1.10. Original Software means the Source Code and Executable form of computer software code that is originally released under this License.

1.11. Patent Claims means any patent claim(s), now owned or hereafter acquired, including without limitation, method, process, and apparatus claims, in any patent Licensable by grantor.

1.12. Source Code means (a) the common form of computer software code in which modifications are made and (b) associated documentation included in or with such code.

1.13. You (or Your) means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License. For legal entities, You includes any entity which controls, is controlled by, or is under common control with You. For purposes of this definition, control means (a)the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b)ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. License Grants.

2.1. The Initial Developer Grant.
Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, the Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license:
(a) under intellectual property rights (other than patent or trademark) Licensable by Initial Developer, to use, reproduce, modify, display, perform, sublicense and distribute the Original Software (or portions thereof), with or without Modifications, and/or as part of a Larger Work; and
(b) under Patent Claims infringed by the making, using or selling of Original Software, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Software (or portions thereof).
(c) The licenses granted in Sections2.1(a) and (b) are effective on the date Initial Developer first distributes or otherwise makes the Original Software available to a third party under the terms of this License.
(d) Notwithstanding Section2.1(b) above, no patent license is granted: (1)for code that You delete from the Original Software, or (2)for infringements caused by: (i)the modification of the Original Software, or (ii)the combination of

the Original Software with other software or devices.

2.2. Contributor Grant.
Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:
(a) under intellectual property rights (other than patent or trademark) Licensable by Contributor to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof), either on an unmodified basis, with other Modifications, as Covered Software and/or as part of a Larger Work; and
(b) under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: (1)Modifications made by that Contributor (or portions thereof); and (2)the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination).
(c) The licenses granted in Sections2.2(a) and 2.2(b) are effective on the date Contributor first distributes or otherwise makes the Modifications available to a third party.
(d) Notwithstanding Section2.2(b) above, no patent license is granted: (1)for any code that Contributor has deleted from the Contributor Version; (2)for infringements caused by: (i)third party modifications of Contributor Version, or (ii)the combination of Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or (3)under Patent Claims infringed by Covered Software in the absence of Modifications made by that Contributor.

3. Distribution Obligations.

3.1. Availability of Source Code.

Any Covered Software that You distribute or otherwise make available in Executable form must also be made available in Source Code form and that Source Code form must be distributed only under the terms of this License. You must include a copy of this License with every copy of the Source Code form of the Covered Software You distribute or otherwise make available. You must inform recipients of any such Covered Software in Executable form as to how they can obtain such Covered Software in Source Code form in a reasonable manner on or through a medium customarily used for software exchange.

3.2. Modifications.

The Modifications that You create or to which You contribute are governed by the terms of this License. You represent that You believe Your Modifications are Your original creation(s) and/or You have sufficient rights to grant the rights conveyed by this License.

3.3. Required Notices.
You must include a notice in each of Your Modifications that identifies You as the Contributor of the Modification. You may not remove or alter any copyright, patent or trademark notices contained within the Covered Software, or any notices of licensing or any descriptive text giving attribution to any Contributor or the Initial Developer.

3.4. Application of Additional Terms.
You may not offer or impose any terms on any Covered Software in Source Code form that alters or restricts the applicable version of this License or the recipients rights hereunder. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However,

you may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear that any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

3.5. Distribution of Executable Versions.
You may distribute the Executable form of the Covered Software under the terms of this License or under the terms of a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable form does not attempt to limit or alter the recipients rights in the Source Code form from the rights set forth in this License. If You distribute the Covered Software in Executable form under a different license, You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.

3.6. Larger Works.
You may create a Larger Work by combining Covered Software with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Software.

4. Versions of the License.

4.1. New Versions.
Sun Microsystems, Inc. is the initial license steward and may publish revised and/or new versions of this License from time to time. Each version will be given a distinguishing version number. Except as provided in Section 4.3, no one other than the license steward has the right to modify this License.

4.2. Effect of New Versions.

You may always continue to use, distribute or otherwise make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. If the Initial Developer includes a notice in the Original Software prohibiting it from being distributed or otherwise made available under any subsequent version of the License, You must distribute and make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. Otherwise, You may also choose to use, distribute or otherwise make the Covered Software available under the terms of any subsequent version of the License published by the license steward.
4.3. Modified Versions.

When You are an Initial Developer and You want to create a new license for Your Original Software, You may create and use a modified version of this License if You: (a)rename the license and remove any references to the name of the license steward (except to note that the license differs from this License); and (b)otherwise make it clear that the license contains terms which differ from this License.

5. DISCLAIMER OF WARRANTY.

COVERED SOFTWARE IS PROVIDED UNDER THIS LICENSE ON AN AS IS BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED SOFTWARE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A

PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED SOFTWARE IS WITH YOU. SHOULD ANY COVERED SOFTWARE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED SOFTWARE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

6. TERMINATION.

6.1. This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.

6.2. If You assert a patent infringement claim (excluding declaratory judgment actions) against Initial Developer or a Contributor (the Initial Developer or Contributor against whom You assert such claim is referred to as Participant) alleging that the Participant Software (meaning the Contributor Version where the Participant is a Contributor or the Original Software where the Participant is the Initial Developer) directly or indirectly infringes any patent, then any and all rights granted directly or indirectly to You by such Participant, the Initial Developer (if the Initial Developer is not the Participant) and all Contributors under Sections2.1 and/or 2.2 of this License shall, upon 60 days notice from Participant terminate prospectively and automatically at the expiration of such 60 day notice period, unless if within such 60 day period You withdraw Your claim with respect to the Participant Software against such Participant either unilaterally or pursuant to a written agreement with Participant.

6.3. In the event of termination under Sections6.1 or 6.2 above, all end user licenses that have been validly granted by You or any distributor hereunder prior to termination (excluding licenses granted to You by any distributor) shall survive termination.

7. LIMITATION OF LIABILITY.

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL YOU, THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED SOFTWARE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO ANY PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOST PROFITS, LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTYS NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

8. U.S. GOVERNMENT END USERS.

The Covered Software is a commercial item, as that term is defined in 48C.F.R.2.101 (Oct. 1995), consisting of commercial computer software (as that term is defined at 48 C.F.R. 252.227-7014(a)(1)) and commercial computer software documentation as such terms are used in 48C.F.R.12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212

and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Software with only those rights set forth herein. This U.S. Government Rights clause is in lieu of, and supersedes, any other FAR, DFAR, or other clause or provision that addresses Government rights in computer software under this License.

9. MISCELLANEOUS.

This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall be governed by the law of the jurisdiction specified in a notice contained within the Original Software (except to the extent applicable law, if any, provides otherwise), excluding such jurisdictions conflict-of-law provisions. Any litigation relating to this License shall be subject to the jurisdiction of the courts located in the jurisdiction and venue specified in a notice contained within the Original Software, with the losing party responsible for costs, including, without limitation, court costs and reasonable attorneys fees and expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License. You agree that You alone are responsible for compliance with the United States export administration regulations (and the export control laws and regulation of any other countries) when You use, distribute or otherwise make available any Covered Software.

10. RESPONSIBILITY FOR CLAIMS.

As between Initial Developer and the Contributors, each party is responsible for claims and damages arising, directly or indirectly, out of its utilization of rights under this License and You agree to work with Initial Developer and Contributors to distribute such responsibility on an equitable basis. Nothing herein is intended or shall be deemed to constitute any admission of liability.

NOTICE PURSUANT TO SECTION 9 OF THE COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL)
The GlassFish code released under the CDDL shall be governed by the laws of the State of California (excluding conflict-of-law provisions). Any litigation relating to this License shall be subject to the jurisdiction of the Federal Courts of the Northern District of California and the state courts of the State of California, with venue lying in Santa Clara County, California.
xml-commons/LICENSE.txt $Id: LICENSE.txt 226068 2003-07-06 03:27:45Z crossley $
See README.txt for additional licensing information.

```
/* ============================================================================
* The Apache Software License, Version 1.1
*
* Copyright (c) 2001-2003 The Apache Software Foundation.  All rights
* reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
*
* 1. Redistributions of source code must retain the above copyright
*    notice, this list of conditions and the following disclaimer.
```

```
*
* 2. Redistributions in binary form must reproduce the above copyright
*    notice, this list of conditions and the following disclaimer in
*    the documentation and/or other materials provided with the
*    distribution.
*
* 3. The end-user documentation included with the redistribution,
*    if any, must include the following acknowledgment:
*       "This product includes software developed by the
*        Apache Software Foundation (http://www.apache.org/)."
*    Alternately, this acknowledgment may appear in the software itself,
*    if and wherever such third-party acknowledgments normally appear.
*
* 4. The names "Apache" and "Apache Software Foundation" must
*    not be used to endorse or promote products derived from this
*    software without prior written permission. For written
*    permission, please contact apache@apache.org.
*
* 5. Products derived from this software may not be called "Apache",
*    nor may "Apache" appear in their name, without prior written
*    permission of the Apache Software Foundation.
*
* THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED
* WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
* OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
* DISCLAIMED.  IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR
* ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
* LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
* USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
* ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
* OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
* OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
* ====================================================================
*
* This software consists of voluntary contributions made by many
* individuals on behalf of the Apache Software Foundation.  For more
* information on the Apache Software Foundation, please see
* <http://www.apache.org/>.
*/
```

1.110 hibernate-validator 6.0.23.Final

1.110.1 Available under license :

No license file was found, but licenses were detected in source scan.

<!--
~ Hibernate Validator, declare and validate application constraints
~
~ License: Apache License, Version 2.0
~ See the license.txt file in the root directory or <http://www.apache.org/licenses/LICENSE-2.0>.
-->

Found in path(s):
* /opt/cola/permits/1473561845_1668493042.4157958/0/hibernate-validator-6-0-23-final-2-jar/META-INF/maven/org.hibernate/hibernate-validator/pom.xml

1.111 apache-commons-lang 3.10

1.111.1 Available under license :

Apache Commons Lang
Copyright 2001-2020 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (https://www.apache.org/).

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed

as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this

License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.112 expression-language-api 3.0.0

1.112.1 Available under license :

COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL) Version 1.0

1. Definitions.

1.1. Contributor. means each individual or entity that creates or contributes to the creation of Modifications.

1.2. Contributor Version. means the combination of the Original Software, prior Modifications used by a Contributor (if any), and the Modifications made by that particular Contributor.

1.3. Covered Software. means (a) the Original Software, or (b) Modifications, or (c) the combination of files

containing Original Software with files containing Modifications, in each case including portions thereof.

1.4. Executable. means the Covered Software in any form other than Source Code.

1.5. Initial Developer. means the individual or entity that first makes Original Software available under this License.

1.6. Larger Work. means a work which combines Covered Software or portions thereof with code not governed by the terms of this License.

1.7. License. means this document.

1.8. Licensable. means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently acquired, any and all of the rights conveyed herein.

1.9. Modifications. means the Source Code and Executable form of any of the following:

A. Any file that results from an addition to, deletion from or modification of the contents of a file containing Original Software or previous Modifications;

B. Any new file that contains any part of the Original Software or previous Modification; or

C. Any new file that is contributed or otherwise made available under the terms of this License.

1.10. Original Software. means the Source Code and Executable form of computer software code that is originally released under this License.

1.11. Patent Claims. means any patent claim(s), now owned or hereafter acquired, including without limitation, method, process, and apparatus claims, in any patent Licensable by grantor.

1.12. Source Code. means (a) the common form of computer software code in which modifications are made and (b) associated documentation included in or with such code.

1.13. You. (or .Your.) means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License. For legal entities, .You. includes any entity which controls, is controlled by, or is under common control with You. For purposes of this definition, .control. means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. License Grants.

2.1. The Initial Developer Grant.

Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, the Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license:

(a) under intellectual property rights (other than patent or trademark) Licensable by Initial Developer, to use, reproduce, modify, display, perform, sublicense and distribute the Original Software (or portions thereof), with or

without Modifications, and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using or selling of Original Software, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Software (or portions thereof).

(c) The licenses granted in Sections 2.1(a) and (b) are effective on the date Initial Developer first distributes or otherwise makes the Original Software available to a third party under the terms of this License.

(d) Notwithstanding Section 2.1(b) above, no patent license is granted: (1) for code that You delete from the Original Software, or (2) for infringements caused by: (i) the modification of the Original Software, or (ii) the combination of the Original Software with other software or devices.

2.2. Contributor Grant.

Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:

(a) under intellectual property rights (other than patent or trademark) Licensable by Contributor to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof), either on an unmodified basis, with other Modifications, as Covered Software and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: (1) Modifications made by that Contributor (or portions thereof); and (2) the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination).

(c) The licenses granted in Sections 2.2(a) and 2.2(b) are effective on the date Contributor first distributes or otherwise makes the Modifications available to a third party.

(d) Notwithstanding Section 2.2(b) above, no patent license is granted: (1) for any code that Contributor has deleted from the Contributor Version; (2) for infringements caused by: (i) third party modifications of Contributor Version, or (ii) the combination of Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or (3) under Patent Claims infringed by Covered Software in the absence of Modifications made by that Contributor.

3. Distribution Obligations.

3.1. Availability of Source Code.

Any Covered Software that You distribute or otherwise make available in Executable form must also be made available in Source Code form and that Source Code form must be distributed only under the terms of this License. You must include a copy of this License with every copy of the Source Code form of the Covered Software You distribute or otherwise make available. You must inform recipients of any such Covered Software in Executable form as to how they can obtain such Covered Software in Source Code form in a reasonable manner on or through a medium customarily used for software exchange.

3.2. Modifications.

The Modifications that You create or to which You contribute are governed by the terms of this License. You represent that You believe Your Modifications are Your original creation(s) and/or You have sufficient rights to grant the rights conveyed by this License.

3.3. Required Notices.

You must include a notice in each of Your Modifications that identifies You as the Contributor of the Modification. You may not remove or alter any copyright, patent or trademark notices contained within the Covered Software, or any notices of licensing or any descriptive text giving attribution to any Contributor or the Initial Developer.

3.4. Application of Additional Terms.

You may not offer or impose any terms on any Covered Software in Source Code form that alters or restricts the applicable version of this License or the recipients. rights hereunder. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However, you may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear that any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

3.5. Distribution of Executable Versions.

You may distribute the Executable form of the Covered Software under the terms of this License or under the terms of a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable form does not attempt to limit or alter the recipient.s rights in the Source Code form from the rights set forth in this License. If You distribute the Covered Software in Executable form under a different license, You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.

3.6. Larger Works.

You may create a Larger Work by combining Covered Software with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Software.

4. Versions of the License.

4.1. New Versions.

Sun Microsystems, Inc. is the initial license steward and may publish revised and/or new versions of this License from time to time. Each version will be given a distinguishing version number. Except as provided in Section 4.3, no one other than the license steward has the right to modify this License.

4.2. Effect of New Versions.

You may always continue to use, distribute or otherwise make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. If the Initial Developer includes a notice in the Original Software prohibiting it from being distributed or otherwise made available under any subsequent version of the License, You must distribute and make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. Otherwise, You may also choose to use, distribute or otherwise make the Covered Software available under the terms of any subsequent

version of the License published by the license steward.

4.3. Modified Versions.

When You are an Initial Developer and You want to create a new license for Your Original Software, You may create and use a modified version of this License if You: (a) rename the license and remove any references to the name of the license steward (except to note that the license differs from this License); and (b) otherwise make it clear that the license contains terms which differ from this License.

5. DISCLAIMER OF WARRANTY.

COVERED SOFTWARE IS PROVIDED UNDER THIS LICENSE ON AN .AS IS. BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED SOFTWARE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED SOFTWARE IS WITH YOU. SHOULD ANY COVERED SOFTWARE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED SOFTWARE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

6. TERMINATION.

6.1. This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.

6.2. If You assert a patent infringement claim (excluding declaratory judgment actions) against Initial Developer or a Contributor (the Initial Developer or Contributor against whom You assert such claim is referred to as .Participant.) alleging that the Participant Software (meaning the Contributor Version where the Participant is a Contributor or the Original Software where the Participant is the Initial Developer) directly or indirectly infringes any patent, then any and all rights granted directly or indirectly to You by such Participant, the Initial Developer (if the Initial Developer is not the Participant) and all Contributors under Sections 2.1 and/or 2.2 of this License shall, upon 60 days notice from Participant terminate prospectively and automatically at the expiration of such 60 day notice period, unless if within such 60 day period You withdraw Your claim with respect to the Participant Software against such Participant either unilaterally or pursuant to a written agreement with Participant.

6.3. In the event of termination under Sections 6.1 or 6.2 above, all end user licenses that have been validly granted by You or any distributor hereunder prior to termination (excluding licenses granted to You by any distributor) shall survive termination.

7. LIMITATION OF LIABILITY.

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL YOU, THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED SOFTWARE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO ANY PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOST PROFITS, LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR

MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTY.S NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

8. U.S. GOVERNMENT END USERS.

The Covered Software is a .commercial item,. as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of .commercial computer software. (as that term is defined at 48 C.F.R. ? 252.227-7014(a)(1)) and .commercial computer software documentation. as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Software with only those rights set forth herein. This U.S. Government Rights clause is in lieu of, and supersedes, any other FAR, DFAR, or other clause or provision that addresses Government rights in computer software under this License.

9. MISCELLANEOUS.

This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall be governed by the law of the jurisdiction specified in a notice contained within the Original Software (except to the extent applicable law, if any, provides otherwise), excluding such jurisdiction.s conflict-of-law provisions. Any litigation relating to this License shall be subject to the jurisdiction of the courts located in the jurisdiction and venue specified in a notice contained within the Original Software, with the losing party responsible for costs, including, without limitation, court costs and reasonable attorneys. fees and expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License. You agree that You alone are responsible for compliance with the United States export administration regulations (and the export control laws and regulation of any other countries) when You use, distribute or otherwise make available any Covered Software.

10. RESPONSIBILITY FOR CLAIMS.

As between Initial Developer and the Contributors, each party is responsible for claims and damages arising, directly or indirectly, out of its utilization of rights under this License and You agree to work with Initial Developer and Contributors to distribute such responsibility on an equitable basis. Nothing herein is intended or shall be deemed to constitute any admission of liability.

NOTICE PURSUANT TO SECTION 9 OF THE COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL)

The code released under the CDDL shall be governed by the laws of the State of California (excluding conflict-of-law provisions). Any litigation relating to this License shall be subject to the jurisdiction of the Federal Courts of the Northern District of California and the state courts of the State of California, with venue lying in Santa Clara County, California.

The GNU General Public License (GPL) Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the

conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF

ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

One line to give the program's name and a brief idea of what it does.

Copyright (C)

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'. This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be

mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

"CLASSPATH" EXCEPTION TO THE GPL VERSION 2

Certain source files distributed by Sun Microsystems, Inc. are subject to the following clarification and special exception to the GPL Version 2, but only where Sun has expressly included in the particular source file's header the words

"Sun designates this particular file as subject to the "Classpath" exception as provided by Sun in the License file that accompanied this code."

Linking this library statically or dynamically with other modules is making a combined work based on this library. Thus, the terms and conditions of the GNU General Public License Version 2 cover the whole combination.

As a special exception, the copyright holders of this library give you permission to link this library with independent modules to produce an executable, regardless of the license terms of these independent modules, and to copy and distribute the resulting executable under terms of your choice, provided that you also meet, for each linked independent module, the terms and conditions of the license of that module.? An independent module is a module which is not derived from or based on this library.? If you modify this library, you may extend this exception to your version of the library, but you are not obligated to do so.? If you do not wish to do so, delete this exception statement from your version.

1.113 snake-yaml 1.33

1.113.1 Available under license :

No license file was found, but licenses were detected in source scan.

<name>Apache License, Version 2.0</name>
<url>http://www.apache.org/licenses/LICENSE-2.0.txt</url>

Found in path(s):
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/META-INF/maven/org.yaml/snakeyaml/pom.xml

No license file was found, but licenses were detected in source scan.

```
/*
* Copyright (c) 2008 Google Inc.
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/
```

Found in path(s):
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/external/com/google/gdata/util/common/base/UnicodeEscaper.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/external/com/google/gdata/util/common/base/PercentEscaper.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/external/com/google/gdata/util/common/base/Escaper.java
No license file was found, but licenses were detected in source scan.

```
// This module is multi-licensed and may be used under the terms
// EPL, Eclipse Public License, V1.0 or later, http://www.eclipse.org/legal
// LGPL, GNU Lesser General Public License, V2.1 or later, http://www.gnu.org/licenses/lgpl.html
// GPL, GNU General Public License, V2 or later, http://www.gnu.org/licenses/gpl.html
// AL, Apache License, V2.0 or later, http://www.apache.org/licenses
// BSD, BSD License, http://www.opensource.org/licenses/bsd-license.php
/**
* A Base64 encoder/decoder.
*
* <p>
* This class is used to encode and decode data in Base64 format as described in RFC 1521.
*
* <p>
* Project home page: <a href="http://www.source-code.biz/base64coder/java/">www.
* source-code.biz/base64coder/java</a><br>
* Author: Christian d'Heureuse, Inventec Informatik AG, Zurich, Switzerland<br>
* Multi-licensed: EPL / LGPL / GPL / AL / BSD.
*/
```

Found in path(s):
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/external/biz/base64Coder/Base64Coder.java

No license file was found, but licenses were detected in source scan.

/**
* Copyright (c) 2008, SnakeYAML
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/

Found in path(s):
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/introspector/PropertySubstitute.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/introspector/FieldProperty.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/tokens/TagToken.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/parser/Production.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/events/MappingEndEvent.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/extensions/compactnotation/CompactConstructor.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/util/ArrayStack.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/LoaderOptions.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/error/MissingEnvironmentVariableException.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/extensions/compactnotation/CompactData.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/reader/ReaderException.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/tokens/CommentToken.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/comments/CommentLine.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/introspector/Property.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/events/CommentEvent.java

* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/tokens/StreamEndToken.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/resolver/Resolver.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/serializer/Serializer.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/tokens/ScalarToken.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/introspector/BeanAccess.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/constructor/CustomClassLoaderConstructor.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/constructor/AbstractConstruct.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/util/ArrayUtils.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/scanner/ScannerImpl.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/DumperOptions.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/resolver/ResolverTuple.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/tokens/FlowEntryToken.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/tokens/FlowMappingEndToken.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/reader/UnicodeReader.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/events/AliasEvent.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/composer/Composer.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/constructor/DuplicateKeyException.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/events/DocumentStartEvent.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/events/ImplicitTuple.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/tokens/KeyToken.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/events/StreamEndEvent.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/events/MappingStartEvent.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/nodes/ScalarNode.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/scanner/ScannerException.java

* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/representer/Represent.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/serializer/SerializerException.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/events/DocumentEndEvent.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/error/Mark.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/tokens/DocumentEndToken.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/tokens/FlowSequenceStartToken.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/constructor/Constructor.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/extensions/compactnotation/PackageCompactConstructor.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/tokens/AnchorToken.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/events/Event.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/tokens/DirectiveToken.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/tokens/BlockEntryToken.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/Yaml.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/nodes/CollectionNode.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/error/YAMLException.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/nodes/MappingNode.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/constructor/BaseConstructor.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/tokens/Token.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/TypeDescription.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/representer/SafeRepresenter.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/emitter/EmitterException.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/tokens/BlockEndToken.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/parser/Parser.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/scanner/Scanner.java

* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/events/StreamStartEvent.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/events/SequenceEndEvent.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/tokens/BlockMappingStartToken.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/parser/ParserImpl.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/events/CollectionEndEvent.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/events/NodeEvent.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/tokens/ValueToken.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/constructor/SafeConstructor.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/comments/CommentType.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/nodes/Node.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/events/ScalarEvent.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/serializer/NumberAnchorGenerator.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/nodes/Tag.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/scanner/Constant.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/constructor/Construct.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/parser/VersionTagsTuple.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/parser/ParserException.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/tokens/TagTuple.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/emitter/Emitable.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/emitter/Emitter.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/tokens/FlowSequenceEndToken.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/introspector/MethodProperty.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/constructor/ConstructorException.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/reader/StreamReader.java

* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/emitter/ScalarAnalysis.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/comments/CommentEventsCollector.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/tokens/AliasToken.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/tokens/BlockSequenceStartToken.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/events/SequenceStartEvent.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/serializer/AnchorGenerator.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/util/PlatformFeatureDetector.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/nodes/SequenceNode.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/nodes/NodeId.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/representer/BaseRepresenter.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/tokens/DocumentStartToken.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/introspector/PropertyUtils.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/introspector/MissingProperty.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/tokens/StreamStartToken.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/introspector/GenericProperty.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/util/UriEncoder.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/tokens/FlowMappingStartToken.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/util/EnumUtils.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/events/CollectionStartEvent.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/representer/Representer.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/composer/ComposerException.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/nodes/AnchorNode.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/emitter/EmitterState.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/nodes/NodeTuple.java

* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/env/EnvScalarConstructor.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/scanner/SimpleKey.java
* /opt/cola/permits/1446188159_1666171012.444366/0/snakeyaml-1-33-sources-1-jar/org/yaml/snakeyaml/error/MarkedYAMLException.java

1.114 metrics-health-checks 4.0.5

1.114.1 Available under license :

Apache-2.0

1.115 jvm-integration-for-metrics 4.0.5

1.115.1 Available under license :

No license file was found, but licenses were detected in source scan.

Manifest-Version: 1.0
Bnd-LastModified: 1545937890888
Build-Jdk: 1.8.0_191
Built-By: artem
Bundle-Description: A set of classes which allow you to monitor critic
al aspects of your Java Virtual Machine using Metrics.
Bundle-License: http://www.apache.org/licenses/LICENSE-2.0.html
Bundle-ManifestVersion: 2
Bundle-Name: JVM Integration for Metrics
Bundle-SymbolicName: io.dropwizard.metrics.jvm
Bundle-Version: 4.0.5
Created-By: Apache Maven Bundle Plugin
Export-Package: com.codahale.metrics.jvm;uses:="com.codahale.metrics,j
avax.management";version="4.0.5"
Implementation-Title: JVM Integration for Metrics
Implementation-URL: http://metrics.dropwizard.io/metrics-jvm
Implementation-Vendor-Id: io.dropwizard.metrics
Implementation-Version: 4.0.5
Import-Package: org.slf4j;version="[1.6.0,2.0.0)",com.sun.management;r
esolution:=optional,com.codahale.metrics;version="[4.0,5)",javax.mana
gement
Require-Capability: osgi.ee;filter:="(&(osgi.ee=JavaSE)(version=1.8))"
Tool: Bnd-3.3.0.201609221906

Found in path(s):
* /opt/cola/permits/1274701574_1648835908.24/0/metrics-jvm-4-0-5-jar/META-INF/MANIFEST.MF

1.116 datasketches-java 1.1.0-incubating

1.116.1 Available under license :

datasketches-java
Copyright 2015-2019 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work

(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses

granted to You under this License for that Work shall terminate
as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
Work or Derivative Works thereof in any medium, with or without
modifications, and in Source or Object form, provided that You
meet the following conditions:

(a) You must give any other recipients of the Work or
Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices
stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works
that You distribute, all copyright, patent, trademark, and
attribution notices from the Source form of the Work,
excluding those notices that do not pertain to any part of
the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its
distribution, then any Derivative Works that You distribute must
include a readable copy of the attribution notices contained
within such NOTICE file, excluding those notices that do not
pertain to any part of the Derivative Works, in at least one
of the following places: within a NOTICE text file distributed
as part of the Derivative Works; within the Source form or
documentation, if provided along with the Derivative Works; or,
within a display generated by the Derivative Works, if and
wherever such third-party notices normally appear. The contents
of the NOTICE file are for informational purposes only and
do not modify the License. You may add Your own attribution
notices within Derivative Works that You distribute, alongside
or as an addendum to the NOTICE text from the Work, provided
that such additional attribution notices cannot be construed
as modifying the License.

You may add Your own copyright statement to Your modifications and
may provide additional or different license terms and conditions
for use, reproduction, or distribution of Your modifications, or
for any such Derivative Works as a whole, provided Your use,
reproduction, and distribution of the Work otherwise complies with
the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
any Contribution intentionally submitted for inclusion in the Work
by You to the Licensor shall be under the terms and conditions of
this License, without any additional terms or conditions.

Notwithstanding the above, nothing herein shall supersede or modify
the terms of any separate license agreement you may have executed
with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
names, trademarks, service marks, or product names of the Licensor,
except as required for reasonable and customary use in describing the
origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
agreed to in writing, Licensor provides the Work (and each
Contributor provides its Contributions) on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied, including, without limitation, any warranties or conditions
of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
PARTICULAR PURPOSE. You are solely responsible for determining the
appropriateness of using or redistributing the Work and assume any
risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
whether in tort (including negligence), contract, or otherwise,
unless required by applicable law (such as deliberate and grossly
negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special,
incidental, or consequential damages of any character arising as a
result of this License or out of the use or inability to use the
Work (including but not limited to damages for loss of goodwill,
work stoppage, computer failure or malfunction, or any and all
other commercial damages or losses), even if such Contributor
has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
the Work or Derivative Works thereof, You may choose to offer,
and charge a fee for, acceptance of support, warranty, indemnity,
or other liability obligations and/or rights consistent with this
License. However, in accepting such obligations, You may act only
on Your own behalf and on Your sole responsibility, not on behalf
of any other Contributor, and only if You agree to indemnify,
defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following
boilerplate notice, with the fields enclosed by brackets "[]"

replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.117 appdynamics-java-agent-api 4.5.13.27526

1.117.1 Available under license :

No license file was found, but licenses were detected in source scan.

Copyright 2019 AppDynamics Inc. All rights reserved.

Found in path(s):
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-javadoc-jar/overview-tree.html
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-javadoc-jar/com/appdynamics/agent/api/package-tree.html
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-javadoc-jar/com/appdynamics/agent/api/ExitCall.html
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-javadoc-jar/deprecated-list.html
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-javadoc-jar/com/appdynamics/agent/api/impl/NoOpTransaction.html
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-javadoc-jar/com/appdynamics/agent/api/Transaction.html
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-javadoc-jar/com/appdynamics/instrumentation/sdk/logging/package-tree.html
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-javadoc-jar/com/appdynamics/instrumentation/sdk/logging/package-summary.html

* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-javadoc-jar/com/appdynamics/agent/api/AppdynamicsAgent.html
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-javadoc-jar/com/appdynamics/instrumentation/sdk/logging/ISDKLogger.html
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-javadoc-jar/com/appdynamics/agent/api/EventPublisher.html
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-javadoc-jar/com/appdynamics/agent/api/impl/package-summary.html
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-javadoc-jar/com/appdynamics/agent/api/EntryTypes.html
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-javadoc-jar/overview-summary.html
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-javadoc-jar/constant-values.html
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-javadoc-jar/com/appdynamics/agent/api/package-summary.html
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-javadoc-jar/index-all.html
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-javadoc-jar/com/appdynamics/agent/api/ExitTypes.html
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-javadoc-jar/help-doc.html
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-javadoc-jar/com/appdynamics/agent/api/impl/NoOpExitCall.html
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-javadoc-jar/com/appdynamics/agent/api/impl/package-tree.html
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-javadoc-jar/com/appdynamics/agent/api/MetricPublisher.html
No license file was found, but licenses were detected in source scan.

/*
* Copyright (c) AppDynamics, Inc., and its affiliates
* 2018
* All Rights Reserved
* THIS IS UNPUBLISHED PROPRIETARY CODE OF APPDYNAMICS, INC.
* The copyright notice above does not evidence any actual or intended publication of such source code
*/

Found in path(s):
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-sources-jar/com/appdynamics/agent/api/impl/NoOpTransaction.java
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-sources-jar/com/appdynamics/agent/api/EumDelegate.java
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-sources-jar/com/appdynamics/agent/api/impl/NoOpExitCall.java
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-sources-jar/com/appdynamics/agent/api/AppdynamicsAgent.java
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-sources-

jar/com/appdynamics/agent/api/ExitCall.java
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-sources-jar/com/appdynamics/agent/api/EventPublisher.java
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-sources-jar/com/appdynamics/agent/api/MetricPublisher.java
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-sources-jar/com/appdynamics/agent/api/Transaction.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright (c) AppDynamics, Inc., and its affiliates
* 2019
* All Rights Reserved
* THIS IS UNPUBLISHED PROPRIETARY CODE OF APPDYNAMICS, INC.
* The copyright notice above does not evidence any actual or intended publication of such source code
*/

Found in path(s):
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-sources-jar/com/appdynamics/agent/api/bootstrap/NoOpTransactionDelegate.java
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-sources-jar/com/appdynamics/agent/api/ExitTypes.java
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-sources-jar/com/appdynamics/agent/api/EntryTypes.java
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-sources-jar/com/appdynamics/agent/api/bootstrap/IApiBootstrapFacade.java
* /opt/cola/permits/1337070300_1654792344.40951/0/java-agent-api-4-5-13-27526-zip/agent-api-sources-jar/com/appdynamics/agent/api/bootstrap/IApiTransactionDelegate.java

1.118 jackson-jaxrs-base 2.14.0

1.118.1 Available under license :

This copy of Jackson JSON processor databind module is licensed under the
Apache (Software) License, version 2.0 ("the License").
See the License for details about distribution rights, and the
specific rights regarding derivate works.

You may obtain a copy of the License at:

http://www.apache.org/licenses/LICENSE-2.0

1.119 jackson-module:-guice 2.14.0

1.119.1 Available under license :

Jackson JSON processor

Jackson is a high-performance, Free/Open Source JSON processing library.
It was originally written by Tatu Saloranta (tatu.saloranta@iki.fi), and has
been in development since 2007.
It is currently developed by a community of developers, as well as supported
commercially by FasterXML.com.

Licensing

Jackson core and extension components may licensed under different licenses.
To find the details that apply to this artifact see the accompanying LICENSE file.
For more information, including possible other licensing options, contact
FasterXML.com (http://fasterxml.com).

Credits

A list of contributors may be found from CREDITS file, which is included
in some artifacts (usually source distributions); but is always available
from the source code management (SCM) system project uses.
This copy of Jackson JSON processor `jackson-module-guice` module is licensed under the
Apache (Software) License, version 2.0 ("the License").
See the License for details about distribution rights, and the
specific rights regarding derivate works.

You may obtain a copy of the License at:

http://www.apache.org/licenses/LICENSE-2.0

1.120 netty-project 4.1.84.Final

1.120.1 Available under license :

No license file was found, but licenses were detected in source scan.

The Netty Project licenses this file to you under the Apache License,
version 2.0 (the "License"); you may not use this file except in compliance
with the License. You may obtain a copy of the License at:
distributed under the License is distributed on an "AS IS" BASIS, WITHOUT

Found in path(s):
* /opt/cola/permits/1470281385_1668115980.505427/0/netty-codec-http-4-1-84-final-jar/META-INF/native-image/io.netty/codec-http/native-image.properties
No license file was found, but licenses were detected in source scan.

Manifest-Version: 1.0
Implementation-Title: Netty/Codec/HTTP

Bundle-Description: Netty is an asynchronous event-driven network appl
ication framework for rapid development of maintainable high perfo
rmance protocol servers and clients.
Automatic-Module-Name: io.netty.codec.http
Bundle-License: https://www.apache.org/licenses/LICENSE-2.0
Bundle-SymbolicName: io.netty.codec-http
Implementation-Version: 4.1.84.Final
Built-By: chris
Bnd-LastModified: 1665536154725
Bundle-ManifestVersion: 2
Implementation-Vendor-Id: io.netty
Bundle-DocURL: https://netty.io/
Bundle-Vendor: The Netty Project
Import-Package: com.aayushatharva.brotli4j.encoder;resolution:=optiona
l,com.jcraft.jzlib;resolution:=optional,io.netty.buffer;version="[4.1
,5)",io.netty.channel;version="[4.1,5)",io.netty.channel.embedded;ver
sion="[4.1,5)",io.netty.handler.codec,io.netty.handler.codec.base64;v
ersion="[4.1,5)",io.netty.handler.codec.compression;version="[4.1,5)"
,io.netty.handler.ssl;version="[4.1,5)",io.netty.handler.stream;versi
on="[4.1,5)",io.netty.util;version="[4.1,5)",io.netty.util.concurrent
;version="[4.1,5)",io.netty.util.internal;version="[4.1,5)",io.netty.
util.internal.logging;version="[4.1,5)",sun.nio.ch;resolution:=option
al,org.eclipse.jetty.npn;version="[1,2)";resolution:=optional,org.ecl
ipse.jetty.alpn;version="[1,2)";resolution:=optional
Require-Capability: osgi.ee;filter:="(&(osgi.ee=JavaSE)(version=1.6))"
Tool: Bnd-2.4.1.201501161923
Implementation-Vendor: The Netty Project
Export-Package: io.netty.handler.codec.http;uses:="io.netty.buffer,io.
netty.channel,io.netty.channel.embedded,io.netty.handler.codec,io.net
ty.handler.codec.compression,io.netty.handler.codec.http.cookie,io.ne
tty.handler.stream,io.netty.util";version="4.1.84",io.netty.handler.c
odec.http.cookie;version="4.1.84",io.netty.handler.codec.http.cors;us
es:="io.netty.channel,io.netty.handler.codec.http";version="4.1.84",i
o.netty.handler.codec.http.multipart;uses:="io.netty.buffer,io.netty.
channel,io.netty.handler.codec,io.netty.handler.codec.http,io.netty.h
andler.stream,io.netty.util";version="4.1.84",io.netty.handler.codec.
http.websocketx;uses:="io.netty.buffer,io.netty.channel,io.netty.hand
ler.codec,io.netty.handler.codec.http,io.netty.handler.stream,io.nett
y.util,io.netty.util.internal.logging";version="4.1.84",io.netty.hand
ler.codec.http.websocketx.extensions;uses:="io.netty.channel,io.netty
.handler.codec,io.netty.handler.codec.http.websocketx";version="4.1.8
4",io.netty.handler.codec.http.websocketx.extensions.compression;uses
:="io.netty.channel,io.netty.handler.codec.http.websocketx.extensions
";version="4.1.84",io.netty.handler.codec.rtsp;uses:="io.netty.buffer
,io.netty.channel,io.netty.handler.codec.http,io.netty.util";version=
"4.1.84",io.netty.handler.codec.spdy;uses:="io.netty.buffer,io.netty.
channel,io.netty.handler.codec,io.netty.handler.codec.http,io.netty.u
til";version="4.1.84"

Bundle-Name: Netty/Codec/HTTP
Bundle-Version: 4.1.84.Final
Created-By: Apache Maven Bundle Plugin
Build-Jdk: 1.8.0_312
Implementation-URL: https://netty.io/netty-codec-http/

Found in path(s):
* /opt/cola/permits/1470281385_1668115980.505427/0/netty-codec-http-4-1-84-final-jar/META-INF/MANIFEST.MF
No license file was found, but licenses were detected in source scan.

<!--
~ Copyright 2012 The Netty Project
~
~ The Netty Project licenses this file to you under the Apache License,
~ version 2.0 (the "License"); you may not use this file except in compliance
~ with the License. You may obtain a copy of the License at:
~
~   https://www.apache.org/licenses/LICENSE-2.0
~
~ Unless required by applicable law or agreed to in writing, software
~ distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
~ WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
~ License for the specific language governing permissions and limitations
~ under the License.
-->

Found in path(s):
* /opt/cola/permits/1470281385_1668115980.505427/0/netty-codec-http-4-1-84-final-jar/META-INF/maven/io.netty/netty-codec-http/pom.xml

1.121 junit 4.13.2

1.121.1 Available under license :

JUnit

Eclipse Public License - v 1.0

THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS ECLIPSE PUBLIC LICENSE ("AGREEMENT"). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.

1. DEFINITIONS

"Contribution" means:

a) in the case of the initial Contributor, the initial code and

documentation distributed under this Agreement, and

b) in the case of each subsequent Contributor:

i) changes to the Program, and

ii) additions to the Program;

where such changes and/or additions to the Program originate from and are distributed by that particular Contributor. A Contribution 'originates' from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor's behalf. Contributions do not include additions to the Program which: (i) are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program.

"Contributor" means any person or entity that distributes the Program.

"Licensed Patents " mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program.

"Program" means the Contributions distributed in accordance with this Agreement.

"Recipient" means anyone who receives the Program under this Agreement, including all Contributors.

2. GRANT OF RIGHTS

a) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, distribute and sublicense the Contribution of such Contributor, if any, and such derivative works, in source code and object code form.

b) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in source code and object code form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder.

c) Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other

intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.

d) Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement.

3. REQUIREMENTS

A Contributor may choose to distribute the Program in object code form under its own license agreement, provided that:

a) it complies with the terms and conditions of this Agreement; and

b) its license agreement:

i) effectively disclaims on behalf of all Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose;

ii) effectively excludes on behalf of all Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits;

iii) states that any provisions which differ from this Agreement are offered by that Contributor alone and not by any other party; and

iv) states that source code for the Program is available from such Contributor, and informs licensees how to obtain it in a reasonable manner on or through a medium customarily used for software exchange.

When the Program is made available in source code form:

a) it must be made available under this Agreement; and

b) a copy of this Agreement must be included with each copy of the Program.

Contributors may not remove or alter any copyright notices contained within the Program.

Each Contributor must identify itself as the originator of its Contribution, if any, in a manner that reasonably allows subsequent Recipients to identify the originator of the Contribution.

4. COMMERCIAL DISTRIBUTION

Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor ("Commercial Contributor") hereby agrees to defend and indemnify every other Contributor ("Indemnified Contributor") against any losses, damages and costs (collectively "Losses") arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must: a) promptly notify the Commercial Contributor in writing of such claim, and b) allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those damages.

5. NO WARRANTY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

6. DISCLAIMER OF LIABILITY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. GENERAL

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. The Eclipse Foundation is the initial Agreement Steward. The Eclipse Foundation may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to distribute the Program (including its Contributions) under the new version. Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether

expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved.

This Agreement is governed by the laws of the State of New York and the intellectual property laws of the United States of America. No party to this Agreement will bring a legal action under this Agreement more than one year after the cause of action arose. Each party waives its rights to a jury trial in any resulting litigation.

1.122 charset-normalizer 2.0.12

1.122.1 Available under license :

Included and Redistributed Files
---

17 files are included in the source distribution tar. They are used to verify the standard functions of this library. They are mandatory to run `pytest` but not required to make the lib usable after install. They DO NOT guarantee that the detection-coverage will not regress.

Those are EITHER pulled from Wikipedia _(CC-BY-SA)_ OR public domain archive.
You SHALL NOT modify any of those files without explicit approval.
MIT License

Copyright (c) 2019 TAHRI Ahmed R.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.123 config 5.3.1

1.123.1 Available under license :

MIT

1.124 zstd-jni 1.4.0-1

1.124.1 Available under license :

BSD License

For Zstandard software

Copyright (c) 2016-present, Facebook, Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name Facebook nor the names of its contributors may be used to
  endorse or promote products derived from this software without specific
  prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
Zstd-jni: JNI bindings to Zstd Library

Copyright (c) 2015-present, Luben Karavelov/ All rights reserved.

BSD License

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this
list of conditions and the following disclaimer in the documentation and/or
other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.,
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether

gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate

copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program

with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such

parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING

OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may

be called something other than `show w' and `show c'; they could even be
mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary. Here is a sample; alter the names:

 Yoyodyne, Inc., hereby disclaims all copyright interest in the program
 `Gnomovision' (which makes passes at compilers) written by James Hacker.

 <signature of Ty Coon>, 1 April 1989
 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into
proprietary programs. If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library. If this is what you want to do, use the GNU Lesser General
Public License instead of this License.

1.125 system-stubs-core 1.1.0

1.125.1 Available under license :

MIT

1.126 disruptor-framework 3.4.2

1.126.1 Available under license :

No license file was found, but licenses were detected in source scan.

/* Copyright 2016 Gil Tene
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-

jar/com/lmax/disruptor/util/ThreadHints.java
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright 2013 LMAX Ltd.
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
```

Found in path(s):
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/EventReleaser.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/EventReleaseAware.java
No license file was found, but licenses were detected in source scan.

```
/*
* Copyright 2011 LMAX Ltd.
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
```

Found in path(s):
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/LiteBlockingWaitStrategy.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/MultiProducerSequencer.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-

jar/com/lmax/disruptor/SequenceGroup.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/LifecycleAware.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/dsl/Disruptor.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/AlertException.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/EventHandler.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/NoOpEventProcessor.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/util/DaemonThreadFactory.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/AggregateEventHandler.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/EventTranslatorTwoArg.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/WorkHandler.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/YieldingWaitStrategy.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/SingleProducerSequencer.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/PhasedBackoffWaitStrategy.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/EventTranslator.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/SequenceReportingEventHandler.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/IgnoreExceptionHandler.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/BatchEventProcessor.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/EventProcessor.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/ExceptionHandler.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/BlockingWaitStrategy.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/WaitStrategy.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/EventTranslatorOneArg.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/util/Util.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/RingBuffer.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/AbstractSequencer.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-

jar/com/lmax/disruptor/EventFactory.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/EventTranslatorThreeArg.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/BusySpinWaitStrategy.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/dsl/EventProcessorInfo.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/SleepingWaitStrategy.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/WorkerPool.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/dsl/EventHandlerGroup.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/dsl/ConsumerRepository.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/FatalExceptionHandler.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/SequenceBarrier.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/WorkProcessor.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/EventTranslatorVararg.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/dsl/ExceptionHandlerSetting.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/ProcessingSequenceBarrier.java
No license file was found, but licenses were detected in source scan.

/*
* Copyright 2012 LMAX Ltd.
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

Found in path(s):
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/dsl/ProducerType.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/Sequence.java

* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/SequenceGroups.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/InsufficientCapacityException.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/DataProvider.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/Sequencer.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/Cursored.java
* /opt/cola/permits/1046386387_1649800803.39/0/disruptor-3-4-2-sources-jar/com/lmax/disruptor/FixedSequenceGroup.java

1.127 apache-avro 1.11.1

1.127.1 Available under license :

Apache Avro
Copyright 2010-2019 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (https://www.apache.org/).

NUnit license acknowledgement:

| Portions Copyright 2002-2012 Charlie Poole or Copyright 2002-2004 James
| W. Newkirk, Michael C. Two, Alexei A. Vorontsov or Copyright 2000-2002
| Philip A. Craig

Based upon the representations of upstream licensors, it is understood that
portions of the mapreduce API included in the Java implementation are licensed
from various contributors under one or more contributor license agreements to
Odiago, Inc. and were then contributed by Odiago to Apache Avro, which has now
made them available under the Apache 2.0 license. The original file header text
is:

| Licensed to Odiago, Inc. under one or more contributor license
| agreements. See the NOTICE file distributed with this work for
| additional information regarding copyright ownership. Odiago, Inc.
| licenses this file to you under the Apache License, Version 2.0
| (the "License"); you may not use this file except in compliance
| with the License. You may obtain a copy of the License at
|
| https://www.apache.org/licenses/LICENSE-2.0
|
| Unless required by applicable law or agreed to in writing, software
| distributed under the License is distributed on an "AS IS" BASIS,
| WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
| implied. See the License for the specific language governing
| permissions and limitations under the License.

The Odiago NOTICE at the time of the contribution:

| This product includes software developed by Odiago, Inc.
| (https://www.wibidata.com).

Apache Ivy includes the following in its NOTICE file:

| Apache Ivy
| Copyright 2007-2010 The Apache Software Foundation
|
| This product includes software developed by
| The Apache Software Foundation (https://www.apache.org/).
|
| Portions of Ivy were originally developed by
| Jayasoft SARL (http://www.jayasoft.fr/)
| and are licensed to the Apache Software Foundation under the
| "Software Grant License Agreement"
|
| SSH and SFTP support is provided by the JCraft JSch package,
| which is open source software, available under
| the terms of a BSD style license.
| The original software and related information is available
| at http://www.jcraft.com/jsch/.

Apache Log4Net includes the following in its NOTICE file:

| Apache log4net
| Copyright 2004-2015 The Apache Software Foundation
|
| This product includes software developed at
| The Apache Software Foundation (https://www.apache.org/).

csharp reflect serializers were contributed by Pitney Bowes Inc.

| Copyright 2019 Pitney Bowes Inc.
| Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License.
| You may obtain a copy of the License at https://www.apache.org/licenses/LICENSE-2.0.
| Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS,
| WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
| See the License for the specific language governing permissions and limitations under the License.

Apache License
Version 2.0, January 2004
https://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of

the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works

that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A

PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

https://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

License for Guava classes included in this binary artifact:

Copyright: 2006-2015 The Guava Authors
License: https://www.apache.org/licenses/LICENSE-2.0 (see above)
Apache Avro
Copyright 2010-2015 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (https://www.apache.org/).

Based upon the representations of upstream licensors, it is understood that
portions of the mapreduce API included in the Java implementation are licensed
from various contributors under one or more contributor license agreements to
Odiago, Inc. and were then contributed by Odiago to Apache Avro, which has now
made them available under the Apache 2.0 license. The original file header text
is:

| Licensed to Odiago, Inc. under one or more contributor license
| agreements. See the NOTICE file distributed with this work for
| additional information regarding copyright ownership. Odiago, Inc.
| licenses this file to you under the Apache License, Version 2.0
| (the "License"); you may not use this file except in compliance
| with the License. You may obtain a copy of the License at
|
| https://www.apache.org/licenses/LICENSE-2.0
|
| Unless required by applicable law or agreed to in writing, software
| distributed under the License is distributed on an "AS IS" BASIS,
| WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
| implied. See the License for the specific language governing
| permissions and limitations under the License.

The Odiago NOTICE at the time of the contribution:

| This product includes software developed by Odiago, Inc.
| (https://www.wibidata.com).

The documentation contains the default Apache Forrest skin.
Apache Forrest includes the following in its NOTICE file:

| Apache Forrest

| Copyright 2002-2007 The Apache Software Foundation.
|
| This product includes software developed at
| The Apache Software Foundation (https://www.apache.org/).
|
| See also the file LICENSE.txt
|
| --
| The purpose of this NOTICE.txt file is to contain notices that are
| required by the copyright owner and their license. Some of the
| accompanying products have an attribution requirement, so see below.
| Other accompanying products do not require attribution, so are not listed.
|
| --
| This product includes software developed by the OpenSymphony Group
| http://www.opensymphony.com/
|
| This product includes software developed for project Krysalis
| http://www.krysalis.org/
|
| This product includes software developed by Andy Clark.
| https://people.apache.org/~andyc/neko/
|
| This product includes software developed by the ExoLab Project
| https://www.exolab.org/
|
| This product includes software developed by TouchGraph LLC
| https://www.touchgraph.com/
|
| This product includes software developed by Marc De Scheemaecker
| http://nanoxml.cyberelf.be/
|
| This product includes software developed by the ANTLR project
| https://www.antlr.org/
|
| This product includes software developed by Chaperon
| http://chaperon.sourceforge.net/
|
| This product includes software developed by Sal Mangano (included in the XSLT Cookbook published by O'Reilly)
| https://www.oreilly.com/catalog/xsltckbk/
|
| This product includes software developed by The Werken Company.
| http://jaxen.werken.com/
|
| This product includes software developed by the jfor project
| http://www.jfor.org/

Apache License
Version 2.0, January 2004
https://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity
on behalf of whom a Contribution has been received by Licensor and
subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
copyright license to reproduce, prepare Derivative Works of,
publicly display, publicly perform, sublicense, and distribute the
Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
(except as stated in this section) patent license to make, have made,
use, offer to sell, sell, import, and otherwise transfer the Work,
where such license applies only to those patent claims licensable
by such Contributor that are necessarily infringed by their
Contribution(s) alone or by combination of their Contribution(s)
with the Work to which such Contribution(s) was submitted. If You
institute patent litigation against any entity (including a
cross-claim or counterclaim in a lawsuit) alleging that the Work
or a Contribution incorporated within the Work constitutes direct
or contributory patent infringement, then any patent licenses
granted to You under this License for that Work shall terminate
as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
Work or Derivative Works thereof in any medium, with or without
modifications, and in Source or Object form, provided that You
meet the following conditions:

(a) You must give any other recipients of the Work or
Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or

agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");

you may not use this file except in compliance with the License.
You may obtain a copy of the License at

https://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

License for the m4 macros used by the C++ implementation:

Files:
* lang/c++/m4/m4_ax_boost_system.m4
 Copyright (c) 2008 Thomas Porschberg <thomas@randspringer.de>
 Copyright (c) 2008 Michael Tindal
 Copyright (c) 2008 Daniel Casimiro <dan.casimiro@gmail.com>
* lang/c++/m4/m4_ax_boost_asio.m4
 Copyright (c) 2008 Thomas Porschberg <thomas@randspringer.de>
 Copyright (c) 2008 Pete Greenwell <pete@mu.org>
* lang/c++/m4/m4_ax_boost_filesystem.m4
 Copyright (c) 2009 Thomas Porschberg <thomas@randspringer.de>
 Copyright (c) 2009 Michael Tindal
 Copyright (c) 2009 Roman Rybalko <libtorrent@romanr.info>
* lang/c++/m4/m4_ax_boost_thread.m4
 Copyright (c) 2009 Thomas Porschberg <thomas@randspringer.de>
 Copyright (c) 2009 Michael Tindal
* lang/c++/m4/m4_ax_boost_regex.m4
 Copyright (c) 2008 Thomas Porschberg <thomas@randspringer.de>
 Copyright (c) 2008 Michael Tindal
* lang/c++/m4/m4_ax_boost_base.m4
 Copyright (c) 2008 Thomas Porschberg <thomas@randspringer.de>

License text:
| Copying and distribution of this file, with or without modification, are
| permitted in any medium without royalty provided the copyright notice
| and this notice are preserved. This file is offered as-is, without any
| warranty.

License for the AVRO_BOOT_NO_TRAIT code in the C++ implementation:
File: lang/c++/api/Boost.hh

| Boost Software License - Version 1.0 - August 17th, 2003
|
| Permission is hereby granted, free of charge, to any person or organization

| obtaining a copy of the software and accompanying documentation covered by
| this license (the "Software") to use, reproduce, display, distribute,
| execute, and transmit the Software, and to prepare derivative works of the
| Software, and to permit third-parties to whom the Software is furnished to
| do so, all subject to the following:
|
| The copyright notices in the Software and this entire statement, including
| the above license grant, this restriction and the following disclaimer,
| must be included in all copies of the Software, in whole or in part, and
| all derivative works of the Software, unless such copies or derivative
| works are solely in the form of machine-executable object code generated by
| a source language processor.
|
| THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
| IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
| FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT
| SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE
| FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE,
| ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
| DEALINGS IN THE SOFTWARE.

Apache License
Version 2.0, January 2004
https://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation

source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable

(except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and

may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify,

defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following
boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

https://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

License for portions of idl.jj in the Java compiler implementation:

Portions of idl.jj were modeled after the example Java 1.5
parser included with JavaCC. For those portions:

Copyright (c) 2006, Sun Microsystems, Inc.
All rights reserved.

| Redistribution and use in source and binary forms, with or without
| modification, are permitted provided that the following conditions are met:
|
| * Redistributions of source code must retain the above copyright notice,
| this list of conditions and the following disclaimer.
| * Redistributions in binary form must reproduce the above copyright
| notice, this list of conditions and the following disclaimer in the
| documentation and/or other materials provided with the distribution.

| * Neither the name of the Sun Microsystems, Inc. nor the names of its
| contributors may be used to endorse or promote products derived from
| this software without specific prior written permission.
|
| THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
| AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
| IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
| ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
| LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
| CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
| SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
| INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
| CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
| ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
| THE POSSIBILITY OF SUCH DAMAGE.

License for Jackson, included in this binary artifact:

Copyright: 2007-2015 Tatu Saloranta and other contributors
Home page: http://jackson.codehaus.org/
License: https://www.apache.org/licenses/LICENSE-2.0.txt (see above)

License for SLF4J, include in this binary artifact:

Copyright (c) 2004-2013 QOS.ch
All rights reserved.

Home page: https://www.slf4j.org/
License: https://slf4j.org/license.html (MIT license)
SLF4J license text (MIT):

| Permission is hereby granted, free of charge, to any person obtaining
| a copy of this software and associated documentation files (the
| "Software"), to deal in the Software without restriction, including
| without limitation the rights to use, copy, modify, merge, publish,
| distribute, sublicense, and/or sell copies of the Software, and to
| permit persons to whom the Software is furnished to do so, subject to
| the following conditions:
|
| The above copyright notice and this permission notice shall be
| included in all copies or substantial portions of the Software.
|
| THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
| EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
| MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
| NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE

| LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
| OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
| WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

License for snappy-java, included in this binary artifact:

Copyright: 2011 Taro L. Saito and other contributors
Home page: http://www.xerial.org/
License: https://www.apache.org/licenses/LICENSE-2.0.txt (see above)

License for Apache Commons Compress, included in this binary artifact:

Copyright: 2004-2015 The Apache Software Foundation
Home page: https://commons.apache.org/compress/
License: https://www.apache.org/licenses/LICENSE-2.0.txt (see above)

Commons Compress includes files derived from the LZMA SDK, version 9.20 (C/ and
CPP/7zip/), in the package org.apache.commons.compress.archivers.sevenz:

| LZMA SDK is placed in the public domain. (https://www.7-zip.org/sdk.html)

License for xz compression, included in this binary artifact:

Home page: https://tukaani.org/xz/java.html

| This Java implementation of XZ has been put into the public domain, thus you
| can do whatever you want with it. All the files in the package have been
| written by Lasse Collin, but some files are heavily based on public domain code
| written by Igor Pavlov.

License for Apache Commons Lang, included in this binary artifact:

Copyright: 2002-2014 The Apache Software Foundation
Home page: https://commons.apache.org/lang/
License: https://www.apache.org/licenses/LICENSE-2.0.txt (see above)

Commons Lang includes software from the Spring Framework, under the
Apache License 2.0:
* StringUtils.containsWhitespace()

License for Apache Velocity, included in this binary artifact:

Copyright: 2000-2015 The Apache Software Foundation

Home page: https://velocity.apache.org/
License: https://www.apache.org/licenses/LICENSE-2.0.txt (see above)

License for Apache Commons Collections, included in this binary artifact:

Copyright: 2001-2015 The Apache Software Foundation
Home page: https://commons.apache.org/proper/commons-collections/
License: https://www.apache.org/licenses/LICENSE-2.0.txt (see above)

License for Jetty, included in this binary artifact:

Copyright: 1995-2015 Mort Bay Consulting Pty Ltd.
Home page: https://eclipse.org/jetty/licenses.php
License: https://www.apache.org/licenses/LICENSE-2.0.txt (see above)

License for Netty, included in this binary artifact:

Copyright: 2011-2013 The Netty Project
Home page: https://netty.io/
License: https://www.apache.org/licenses/LICENSE-2.0.txt (see above)

Netty contains the following code (copied from its NOTICE file with licenses
added inline):
| This product contains the extensions to Java Collections Framework which has
| been derived from the works by JSR-166 EG, Doug Lea, and Jason T. Greene:
|
| * LICENSE:
|| The person or persons who have associated work with this document (the
|| "Dedicator" or "Certifier") hereby either (a) certifies that, to the best of
|| his knowledge, the work of authorship identified is in the public domain of
|| the country from which the work is published, or (b) hereby dedicates whatever
|| copyright the dedicators holds in the work of authorship identified below (the
|| "Work") to the public domain. A certifier, moreover, dedicates any copyright
|| interest he may have in the associated work, and for these purposes, is
|| described as a "dedicator" below.
||
|| A certifier has taken reasonable steps to verify the copyright status of this
|| work. Certifier recognizes that his good faith efforts may not shield him from
|| liability if in fact the work certified is not in the public domain.
||
|| Dedicator makes this dedication for the benefit of the public at large and to
|| the detriment of the Dedicator's heirs and successors. Dedicator intends this
|| dedication to be an overt act of relinquishment in perpetuate of all present
|| and future rights under copyright law, whether vested or contingent, in the
|| Work. Dedicator understands that such relinquishment of all rights includes

|| the relinquishment of all rights to enforce (by lawsuit or otherwise) those
|| copyrights in the Work.
||
|| Dedicator recognizes that, once placed in the public domain, the Work may be
|| freely reproduced, distributed, transmitted, used, modified, built upon, or
|| otherwise exploited by anyone for any purpose, commercial or non-commercial,
|| and in any way, including by methods that have not yet been invented or
|| conceived.
| * HOMEPAGE:
| * http://gee.cs.oswego.edu/cgi-bin/viewcvs.cgi/jsr166/
| * http://viewvc.jboss.org/cgi-bin/viewvc.cgi/jbosscache/experimental/jsr166/
|
| This product contains a modified version of Robert Harder's Public Domain
| Base64 Encoder and Decoder, which can be obtained at:
|
| * LICENSE: public domain (see JSR-166 license above)
| * HOMEPAGE:
| * http://iharder.sourceforge.net/current/java/base64/
|
| This product contains a modified version of 'JZlib', a re-implementation of
| zlib in pure Java, which can be obtained at:
|
| * LICENSE:
|| Copyright (c) 2000,2001,2002,2003,2004 ymnk, JCraft,Inc. All rights reserved.
||
|| Redistribution and use in source and binary forms, with or without
|| modification, are permitted provided that the following conditions are met:
||
|| 1. Redistributions of source code must retain the above copyright notice,
|| this list of conditions and the following disclaimer.
||
|| 2. Redistributions in binary form must reproduce the above copyright
|| notice, this list of conditions and the following disclaimer in
|| the documentation and/or other materials provided with the distribution.
||
|| 3. The names of the authors may not be used to endorse or promote products
|| derived from this software without specific prior written permission.
||
|| THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES,
|| INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
|| FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL JCRAFT,
|| INC. OR ANY CONTRIBUTORS TO THIS SOFTWARE BE LIABLE FOR ANY DIRECT, INDIRECT,
|| INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
|| LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,
|| OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
|| LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
|| NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
|| EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

| * HOMEPAGE:
| * http://www.jcraft.com/jzlib/

License for the javax.servlet API, included in this binary artifact:

Copyright (c) 2003-2004 The Apache Software Foundation
License: https://www.apache.org/licenses/LICENSE-2.0.txt (see above)
Source: http://grepcode.com/project/repo1.maven.org/maven2/javax.servlet/servlet-api/

License for Apache Commons Codec, included in this binary artifact:

Copyright: 2002-2015 The Apache Software Foundation
Home page: https://commons.apache.org/proper/commons-codec/
License: https://www.apache.org/licenses/LICENSE-2.0.txt (see above)

License for Apache Commons CLI, included in this binary artifact:

Copyright: 2001-2015 The Apache Software Foundation
Home page: https://commons.apache.org/proper/commons-cli/
License: https://www.apache.org/licenses/LICENSE-2.0.txt (see above)

License for Apache Commons Logging, included in this binary artifact:

Copyright: 2002-2014 The Apache Software Foundation
Home page: https://commons.apache.org/proper/commons-logging/
License: https://www.apache.org/licenses/LICENSE-2.0.txt (see above)

License for Apache Commons HttpClient, included in this binary artifact:

Copyright: 1999-2005 The Apache Software Foundation
License: https://www.apache.org/licenses/LICENSE-2.0.txt (see above)

License for Apache Hadoop, included in this binary artifact:

Copyright: 2001-2015 The Apache Software Foundation
Home page: https://commons.apache.org/proper/commons-collections/
License: https://www.apache.org/licenses/LICENSE-2.0.txt (see above)

Hadoop contains the following code (from its LICENSE file):

The org.apache.hadoop.util.bloom.* classes:
| Copyright (c) 2005, European Commission project OneLab under contract

| 034819 (http://www.one-lab.org)
| All rights reserved.
| Redistribution and use in source and binary forms, with or
| without modification, are permitted provided that the following
| conditions are met:
| - Redistributions of source code must retain the above copyright
| notice, this list of conditions and the following disclaimer.
| - Redistributions in binary form must reproduce the above copyright
| notice, this list of conditions and the following disclaimer in
| the documentation and/or other materials provided with the distribution.
| - Neither the name of the University Catholique de Louvain - UCL
| nor the names of its contributors may be used to endorse or
| promote products derived from this software without specific prior
| written permission.
|
| THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
| "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
| LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
| FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
| COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
| INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
| BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
| LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
| CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
| LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
| ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
| POSSIBILITY OF SUCH DAMAGE.

License for Google Guava, included in this binary artifact:

Copyright: 2007-2015 The Guava Authors
Home page: https://github.com/google/guava
License: https://www.apache.org/licenses/LICENSE-2.0 (see above)

License for Apache Commons Math, included in this binary artifact:

Copyright: 2001-2015 The Apache Software Foundation
Home page: https://commons.apache.org/proper/commons-math/
License: https://www.apache.org/licenses/LICENSE-2.0 (see above)

Commons Math includes other works under licenses compatible with the
Apache Software License:
| APACHE COMMONS MATH DERIVATIVE WORKS:
|
| The Apache commons-math library includes a number of subcomponents
| whose implementation is derived from original sources written

| in C or Fortran. License terms of the original sources
| are reproduced below.
|
| ==
| For the lmder, lmpar and qrsolv Fortran routine from minpack and translated in
| the LevenbergMarquardtOptimizer class in package
| org.apache.commons.math3.optimization.general
| Original source copyright and license statement:
|
| Minpack Copyright Notice (1999) University of Chicago. All rights reserved
|
| Redistribution and use in source and binary forms, with or
| without modification, are permitted provided that the
| following conditions are met:
|
| 1. Redistributions of source code must retain the above
| copyright notice, this list of conditions and the following
| disclaimer.
|
| 2. Redistributions in binary form must reproduce the above
| copyright notice, this list of conditions and the following
| disclaimer in the documentation and/or other materials
| provided with the distribution.
|
| 3. The end-user documentation included with the
| redistribution, if any, must include the following
| acknowledgment:
|
| "This product includes software developed by the
| University of Chicago, as Operator of Argonne National
| Laboratory.
|
| Alternately, this acknowledgment may appear in the software
| itself, if and wherever such third-party acknowledgments
| normally appear.
|
| 4. WARRANTY DISCLAIMER. THE SOFTWARE IS SUPPLIED "AS IS"
| WITHOUT WARRANTY OF ANY KIND. THE COPYRIGHT HOLDER, THE
| UNITED STATES, THE UNITED STATES DEPARTMENT OF ENERGY, AND
| THEIR EMPLOYEES: (1) DISCLAIM ANY WARRANTIES, EXPRESS OR
| IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES
| OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE
| OR NON-INFRINGEMENT, (2) DO NOT ASSUME ANY LEGAL LIABILITY
| OR RESPONSIBILITY FOR THE ACCURACY, COMPLETENESS, OR
| USEFULNESS OF THE SOFTWARE, (3) DO NOT REPRESENT THAT USE OF
| THE SOFTWARE WOULD NOT INFRINGE PRIVATELY OWNED RIGHTS, (4)
| DO NOT WARRANT THAT THE SOFTWARE WILL FUNCTION
| UNINTERRUPTED, THAT IT IS ERROR-FREE OR THAT ANY ERRORS WILL

| BE CORRECTED.
|
| 5. LIMITATION OF LIABILITY. IN NO EVENT WILL THE COPYRIGHT
| HOLDER, THE UNITED STATES, THE UNITED STATES DEPARTMENT OF
| ENERGY, OR THEIR EMPLOYEES: BE LIABLE FOR ANY INDIRECT,
| INCIDENTAL, CONSEQUENTIAL, SPECIAL OR PUNITIVE DAMAGES OF
| ANY KIND OR NATURE, INCLUDING BUT NOT LIMITED TO LOSS OF
| PROFITS OR LOSS OF DATA, FOR ANY REASON WHATSOEVER, WHETHER
| SUCH LIABILITY IS ASSERTED ON THE BASIS OF CONTRACT, TORT
| (INCLUDING NEGLIGENCE OR STRICT LIABILITY), OR OTHERWISE,
| EVEN IF ANY OF SAID PARTIES HAS BEEN WARNED OF THE
| POSSIBILITY OF SUCH LOSS OR DAMAGES.
| ==========
|
| Copyright and license statement for the odex Fortran routine developed by
| E. Hairer and G. Wanner and translated in GraggBulirschStoerIntegrator class
| in package org.apache.commons.math3.ode.nonstiff:
|
|
| Copyright (c) 2004, Ernst Hairer
|
| Redistribution and use in source and binary forms, with or without
| modification, are permitted provided that the following conditions are
| met:
|
| - Redistributions of source code must retain the above copyright
| notice, this list of conditions and the following disclaimer.
|
| - Redistributions in binary form must reproduce the above copyright
| notice, this list of conditions and the following disclaimer in the
| documentation and/or other materials provided with the distribution.
|
| THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
| IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
| TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
| PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR
| CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
| EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
| PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
| PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
| LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
| NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
| SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
| ==========
|
| Copyright and license statement for the original lapack fortran routines
| translated in EigenDecompositionImpl class in package
| org.apache.commons.math3.linear:

|
| Copyright (c) 1992-2008 The University of Tennessee. All rights reserved.
|
| $COPYRIGHT$
|
| Additional copyrights may follow
|
| $HEADER$
|
| Redistribution and use in source and binary forms, with or without
| modification, are permitted provided that the following conditions are
| met:
|
| - Redistributions of source code must retain the above copyright
| notice, this list of conditions and the following disclaimer.
|
| - Redistributions in binary form must reproduce the above copyright
| notice, this list of conditions and the following disclaimer listed
| in this license in the documentation and/or other materials
| provided with the distribution.
|
| - Neither the name of the copyright holders nor the names of its
| contributors may be used to endorse or promote products derived from
| this software without specific prior written permission.
|
| THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
| "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
| LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
| A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
| OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
| SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
| LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
| DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
| THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
| (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
| OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
| ==
|
| Copyright and license statement for the original Mersenne twister C
| routines translated in MersenneTwister class in package
| org.apache.commons.math3.random:
|
| Copyright (C) 1997 - 2002, Makoto Matsumoto and Takuji Nishimura,
| All rights reserved.
|
| Redistribution and use in source and binary forms, with or without
| modification, are permitted provided that the following conditions
| are met:

|
| 1. Redistributions of source code must retain the above copyright
| notice, this list of conditions and the following disclaimer.
|
| 2. Redistributions in binary form must reproduce the above copyright
| notice, this list of conditions and the following disclaimer in the
| documentation and/or other materials provided with the distribution.
|
| 3. The names of its contributors may not be used to endorse or promote
| products derived from this software without specific prior written
| permission.
|
| THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
| "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
| LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
| A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
| CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
| EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
| PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
| PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
| LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
| NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
| SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
|
| ==
|
| The class "org.apache.commons.math3.exception.util.LocalizedFormatsTest" is
| an adapted version of "OrekitMessagesTest" test class for the Orekit library
| The "org.apache.commons.math3.analysis.interpolation.HermiteInterpolator"
| has been imported from the Orekit space flight dynamics library.
|
| Th Orekit library is described at:
| https://www.orekit.org/forge/projects/orekit
| The original files are distributed under the terms of the Apache 2 license
| which is: Copyright 2010 CS Communication & Systmes

License for XMLenc, included in this binary artifact:

Copyright 2003-2011, Ernst de Haan
All rights reserved.

| Redistribution and use in source and binary forms, with or without
| modification, are permitted provided that the following conditions are met:
|
| 1. Redistributions of source code must retain the above copyright notice, this
| list of conditions and the following disclaimer.
|

| 2. Redistributions in binary form must reproduce the above copyright notice,
| this list of conditions and the following disclaimer in the documentation
| and/or other materials provided with the distribution.
|
| THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
| AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
| IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
| DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
| FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
| DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
| SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
| CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
| OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
| OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

License for Apache Commons IO, included in this binary artifact:

Copyright: 2002-2015 The Apache Software Foundation
Home page: https://commons.apache.org/proper/io/
License: https://www.apache.org/licenses/LICENSE-2.0 (see above)

License for Apache Commons Net, included in this binary artifact:

Copyright: 2001-2015 The Apache Software Foundation
Home page: https://commons.apache.org/proper/commons-net/
License: https://www.apache.org/licenses/LICENSE-2.0 (see above)

License for Apache Log4j, included in this binary artifact:

Copyright: 1999-2015 The Apache Software Foundation
Home page: https://logging.apache.org/log4j/
License: https://www.apache.org/licenses/LICENSE-2.0 (see above)

License for Apache Commons Configuration, included in this binary artifact:

Copyright: 2001-2015 The Apache Software Foundation
Home page: https://commons.apache.org/proper/commons-configuration/
License: https://www.apache.org/licenses/LICENSE-2.0 (see above)

License for Apache Commons Digester, included in this binary artifact:

Copyright: 2001-2015 The Apache Software Foundation
License: https://www.apache.org/licenses/LICENSE-2.0 (see above)

License for Apache Commons Beanutils, included in this binary artifact:

Copyright: 2000-2015 The Apache Software Foundation
License: https://www.apache.org/licenses/LICENSE-2.0 (see above)

License for Google Protocol Buffers, included in this binary artifact:

Copyright 2014, Google Inc. All rights reserved.

| Redistribution and use in source and binary forms, with or without
| modification, are permitted provided that the following conditions are
| met:
|
| * Redistributions of source code must retain the above copyright
| notice, this list of conditions and the following disclaimer.
| * Redistributions in binary form must reproduce the above
| copyright notice, this list of conditions and the following disclaimer
| in the documentation and/or other materials provided with the
| distribution.
| * Neither the name of Google Inc. nor the names of its
| contributors may be used to endorse or promote products derived from
| this software without specific prior written permission.
|
| THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
| "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
| LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
| A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
| OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
| SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
| LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
| DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
| THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
| (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
| OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
|
| Code generated by the Protocol Buffer compiler is owned by the owner
| of the input file used when generating it. This code is not
| standalone and requires a support library to be linked with it. This
| support library is itself covered by the above license.

License for Apache HttpClient, included in this binary artifact:

Copyright: 1999-2015 The Apache Software Foundation
License: https://www.apache.org/licenses/LICENSE-2.0 (see above)

HttpClient contains the following data under the terms of the MPL:
| This project includes Public Suffix List copied from
| <https://publicsuffix.org/list/effective_tld_names.dat>
| licensed under the terms of the Mozilla Public License, v. 2.0
|
| Full license text: META-INF/mpl-2.0.text

License for Apache Directory, included in this binary artifact:

Copyright: 2003-2015 The Apache Software Foundation
License: https://www.apache.org/licenses/LICENSE-2.0 (see above)

Apache Directory includes other works under licenses compatible with the
Apache Software License:

| --
|
| The OpenLDAP Public License
| Version 2.8, 17 August 2003
|
| Redistribution and use of this software and associated documentation
| ("Software"), with or without modification, are permitted provided
| that the following conditions are met:
|
| 1. Redistributions in source form must retain copyright statements
| and notices,
|
| 2. Redistributions in binary form must reproduce applicable copyright
| statements and notices, this list of conditions, and the following
| disclaimer in the documentation and/or other materials provided
| with the distribution, and
|
| 3. Redistributions must contain a verbatim copy of this document.
|
| The OpenLDAP Foundation may revise this license from time to time.
| Each revision is distinguished by a version number. You may use
| this Software under terms of this license revision or under the
| terms of any subsequent revision of the license.
|
| THIS SOFTWARE IS PROVIDED BY THE OPENLDAP FOUNDATION AND ITS
| CONTRIBUTORS ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES,
| INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
| AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
| SHALL THE OPENLDAP FOUNDATION, ITS CONTRIBUTORS, OR THE AUTHOR(S)
| OR OWNER(S) OF THE SOFTWARE BE LIABLE FOR ANY DIRECT, INDIRECT,
| INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,

| BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
| LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
| CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
| LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
| ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
| POSSIBILITY OF SUCH DAMAGE.
|
| The names of the authors and copyright holders must not be used in
| advertising or otherwise to promote the sale, use or other dealing
| in this Software without specific, written prior permission. Title
| to copyright in this Software shall at all times remain with copyright
| holders.
|
| OpenLDAP is a registered trademark of the OpenLDAP Foundation.
|
| Copyright 1999-2003 The OpenLDAP Foundation, Redwood City,
| California, USA. All Rights Reserved. Permission to copy and
| distribute verbatim copies of this document is granted.
|
| --
|
| Copyright (c) 2000 - 2011 The Legion Of The Bouncy Castle (https://www.bouncycastle.org)
|
| Permission is hereby granted, free of charge, to any person obtaining a
| copy of this software and associated documentation files (the "Software"),
| to deal in the Software without restriction, including without limitation
| the rights to use, copy, modify, merge, publish, distribute, sublicense,
| and/or sell copies of the Software, and to permit persons to whom the
| Software is furnished to do so, subject to the following conditions:
|
| The above copyright notice and this permission notice shall be included in
| all copies or substantial portions of the Software.
|
| THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
| IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
| FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
| AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
| LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
| OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
| THE SOFTWARE.
|
|
==
================
| slf4j 1.7.10 license:
| --
| Copyright (c) 2004-2013 QOS.ch
| All rights reserved.

|
| Permission is hereby granted, free of charge, to any person obtaining
| a copy of this software and associated documentation files (the
| "Software"), to deal in the Software without restriction, including
| without limitation the rights to use, copy, modify, merge, publish,
| distribute, sublicense, and/or sell copies of the Software, and to
| permit persons to whom the Software is furnished to do so, subject to
| the following conditions:
|
| The above copyright notice and this permission notice shall be
| included in all copies or substantial portions of the Software.
|
| THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
| EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
| MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
| NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
| LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
| OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
| WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
|
| ==
| For the AVL Set code : http://bobah.net/d4d/source-code/misc/java-avl-tree
| ---
| Copyright 2001-2014 Vladimir Lysyy
| Licensed under the Apache License, Version 2.0 (the "License");
| you may not use this source code except in compliance with the License.
| You may obtain a copy of the License at
|
| https://www.apache.org/licenses/LICENSE-2.0
|
| Unless required by applicable law or agreed to in writing, software
| distributed under the License is distributed on an "AS IS" BASIS,
| WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
| See the License for the specific language governing permissions and
| limitations under the License.

License for the JSR-305 annotations, included in this binary artifact:

Copyright: 2011-2015 Stephen Connolly, Greg Lucas
License: https://www.apache.org/licenses/LICENSE-2.0 (see above)

License for Apache ZooKeeper, included in this binary artifact:

Copyright: 2009-2015 The Apache Software Foundation
License: https://www.apache.org/licenses/LICENSE-2.0 (see above)

License for Jersey, included in this binary artifact:

Copyright (c) 2015 Oracle and/or its affiliates.
All rights reserved.
License: CDDL 1.1: META-INF/cddl-1.1.text
Source: https://github.com/jersey/jersey-1.x-old

License for LevelDB JNI, included in this binary artifact:

Copyright (c) 2011 FuseSource Corp. All rights reserved.

| Redistribution and use in source and binary forms, with or without
| modification, are permitted provided that the following conditions are
| met:
|
| * Redistributions of source code must retain the above copyright
| notice, this list of conditions and the following disclaimer.
| * Redistributions in binary form must reproduce the above
| copyright notice, this list of conditions and the following disclaimer
| in the documentation and/or other materials provided with the
| distribution.
| * Neither the name of FuseSource Corp. nor the names of its
| contributors may be used to endorse or promote products derived from
| this software without specific prior written permission.
|
| THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
| "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
| LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
| A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
| OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
| SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
| LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
| DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
| THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
| (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
| OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

License for iq80 LevelDB Java API, included in this binary artifact:

Copyright 2011 Dain Sundstrom <dain@iq80.com>
Copyright 2011 FuseSource Corp. http://fusesource.com
License: https://www.apache.org/licenses/LICENSE-2.0 (see above)

License for jquery and jquery-ui, included in this binary artifact:

License: The MIT License (MIT): https://tldrlegal.com/license/mit-license
Home page: https://jquery.org/license/

Copyright (c) <year> <copyright holders>

| Permission is hereby granted, free of charge, to any person obtaining a copy of
| this software and associated documentation files (the "Software"), to deal in
| the Software without restriction, including without limitation the rights to
| use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies
| of the Software, and to permit persons to whom the Software is furnished to do
| so, subject to the following conditions:
|
| The above copyright notice and this permission notice shall be included in all
| copies or substantial portions of the Software.
|
| THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
| IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
| FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
| AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
| LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
| OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
| SOFTWARE.

License for the javax.xml.bind jaxb API, included in this binary artifact:

Copyright (c) 2004-2010 Oracle and/or its affiliates.
All rights reserved.
License: CDDL 1.0: META-INF/cddl-1.0.text
Source: http://www.grepcode.com/project/repo1.maven.org/maven2/javax.xml.bind/jaxb-api/

License for the javax.xml.stream stax API, included in this binary artifact:

Copyright (c) 2004-2006 Oracle and/or its affiliates.
All rights reserved.
License: CDDL 1.0: META-INF/cddl-1.0.text
Source: http://grepcode.com/project/repo1.maven.org/maven2/javax.xml.stream/stax-api/

License for the javax.activation API, included in this binary artifact:

Copyright (c) 2004-2006 Oracle and/or its affiliates.
All rights reserved.
License: CDDL 1.0: META-INF/cddl-1.0.text
Source: http://grepcode.com/project/repo1.maven.org/maven2/javax.activation/activation/

License for the javax.ws.rs API, included in this binary artifact:

Copyright (c) 1996-2015, Oracle Corporation and/or its affiliates.
All rights reserved.
License: CDDL 1.1: META-INF/cddl-1.1.text
Source: http://grepcode.com/project/repo1.maven.org/maven2/javax.ws.rs/javax.ws.rs-api/

License for JOpt Simple, included in this binary artifact:

Copyright (c) 2004-2015 Paul R. Holser, Jr.

| Permission is hereby granted, free of charge, to any person obtaining
| a copy of this software and associated documentation files (the
| "Software"), to deal in the Software without restriction, including
| without limitation the rights to use, copy, modify, merge, publish,
| distribute, sublicense, and/or sell copies of the Software, and to
| permit persons to whom the Software is furnished to do so, subject to
| the following conditions:
|
| The above copyright notice and this permission notice shall be
| included in all copies or substantial portions of the Software.
|
| THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
| EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
| MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
| NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
| LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
| OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
| WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

License for SLF4J API, included in this binary artifact:

Copyright (c) 2004-2013 QOS.ch
All rights reserved.

| Permission is hereby granted, free of charge, to any person obtaining
| a copy of this software and associated documentation files (the
| "Software"), to deal in the Software without restriction, including
| without limitation the rights to use, copy, modify, merge, publish,
| distribute, sublicense, and/or sell copies of the Software, and to
| permit persons to whom the Software is furnished to do so, subject to
| the following conditions:
|
| The above copyright notice and this permission notice shall be
| included in all copies or substantial portions of the Software.

|
| THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
| EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
| MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
| NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
| LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
| OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
| WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

License for Guava classes included in this binary artifact:

Copyright: 2006-2015 The Guava Authors
License: https://www.apache.org/licenses/LICENSE-2.0 (see above)

Apache License
Version 2.0, January 2004
https://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You

institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
   any Contribution intentionally submitted for inclusion in the Work
   by You to the Licensor shall be under the terms and conditions of
   this License, without any additional terms or conditions.
   Notwithstanding the above, nothing herein shall supersede or modify
   the terms of any separate license agreement you may have executed
   with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

https://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

License for the Jansson C JSON parser used in the C implementation:

Copyright (c) 2009-2011 Petri Lehtinen <petri@digip.org>

Some files include an additional copyright notice:
* lang/c/jansson/src/pack_unpack.c
Copyright (c) 2011 Graeme Smecher <graeme.smecher@mail.mcgill.ca>
* lang/c/jansson/test/suites/api/test_unpack.c
Copyright (c) 2011 Graeme Smecher <graeme.smecher@mail.mcgill.ca>
* lang/c/jansson/src/memory.c
Copyright (c) 2011 Basile Starynkevitch <basile@starynkevitch.net>

| Permission is hereby granted, free of charge, to any person obtaining a copy
| of this software and associated documentation files (the "Software"), to deal
| in the Software without restriction, including without limitation the rights
| to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
| copies of the Software, and to permit persons to whom the Software is
| furnished to do so, subject to the following conditions:
|
| The above copyright notice and this permission notice shall be included in
| all copies or substantial portions of the Software.
|

| THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
| IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
| FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
| AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
| LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
| OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
| THE SOFTWARE.

License for msinttypes.h and msstdint.h used in the C implementation:
Source from:
https://code.google.com/p/msinttypes/downloads/detail?name=msinttypes-r26.zip

Copyright (c) 2006-2008 Alexander Chemeris

| Redistribution and use in source and binary forms, with or without
| modification, are permitted provided that the following conditions are met:
|
| 1. Redistributions of source code must retain the above copyright notice,
| this list of conditions and the following disclaimer.
|
| 2. Redistributions in binary form must reproduce the above copyright
| notice, this list of conditions and the following disclaimer in the
| documentation and/or other materials provided with the distribution.
|
| 3. The name of the author may be used to endorse or promote products
| derived from this software without specific prior written permission.
|
| THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS" AND ANY EXPRESS OR IMPLIED
| WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
| MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
| EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
| SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
| PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
| OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
| WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
| OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
| ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

License for st.c and st.h used in the C implementation:

| This is a public domain general purpose hash table package written by
| Peter Moore @ UCB.

License for Dirent API for Microsoft Visual Studio used in the C implementation:
Source from:

http://www.softagalleria.net/download/dirent/dirent-1.11.zip

Copyright (C) 2006 Toni Ronkko

| Permission is hereby granted, free of charge, to any person obtaining
| a copy of this software and associated documentation files (the
| ``Software''), to deal in the Software without restriction, including
| without limitation the rights to use, copy, modify, merge, publish,
| distribute, sublicense, and/or sell copies of the Software, and to
| permit persons to whom the Software is furnished to do so, subject to
| the following conditions:
|
| The above copyright notice and this permission notice shall be included
| in all copies or substantial portions of the Software.
|
| THE SOFTWARE IS PROVIDED ``AS IS'', WITHOUT WARRANTY OF ANY KIND, EXPRESS
| OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
| MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
| IN NO EVENT SHALL TONI RONKKO BE LIABLE FOR ANY CLAIM, DAMAGES OR
| OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
| ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
| OTHER DEALINGS IN THE SOFTWARE.

License for ivy-2.2.0.jar used in the python implementation:

Apache License version 2.0 (see above)

License for pyAntTasks-1.3.jar used in the python implementation:

Apache License version 2.0 (see above)

License for NUnit binary included with the C# implementation:
File: nunit.framework.dll

| NUnit License
|
| Copyright 2002-2015 Charlie Poole
| Copyright 2002-2004 James W. Newkirk, Michael C. Two, Alexei A. Vorontsov
| Copyright 2000-2002 Philip A. Craig
|
| This software is provided 'as-is', without any express or implied warranty. In
| no event will the authors be held liable for any damages arising from the use
| of this software.
|
| Permission is granted to anyone to use this software for any purpose, including

| commercial applications, and to alter it and redistribute it freely, subject to
| the following restrictions:
|
| The origin of this software must not be misrepresented; you must not claim that
| you wrote the original software. If you use this software in a product, an
| acknowledgment (see the following) in the product documentation is required.
|
| Portions Copyright 2002-2012 Charlie Poole or Copyright 2002-2004 James W.
| Newkirk, Michael C. Two, Alexei A. Vorontsov or Copyright 2000-2002 Philip A.
| Craig
|
| Altered source versions must be plainly marked as such, and must not be
| misrepresented as being the original software.
|
| This notice may not be removed or altered from any source distribution.
| License Note
|
| This license is based on the open source zlib/libpng license. The idea was to
| keep the license as simple as possible to encourage use of NUnit in free and
| commercial applications and libraries, but to keep the source code together and
| to give credit to the NUnit contributors for their efforts. While this license
| allows shipping NUnit in source and binary form, if shipping a NUnit variant is
| the sole purpose of your product, please let us know.

License for the Json.NET binary included with the C# implementation:
File: Newtonsoft.Json.dll

Copyright (c) 2007 James Newton-King

| Permission is hereby granted, free of charge, to any person obtaining
| a copy of this software and associated documentation files (the
| "Software"), to deal in the Software without restriction, including
| without limitation the rights to use, copy, modify, merge, publish,
| distribute, sublicense, and/or sell copies of the Software, and to
| permit persons to whom the Software is furnished to do so, subject to
| the following conditions:
|
| The above copyright notice and this permission notice shall be
| included in all copies or substantial portions of the Software.
|
| THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
| EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
| MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
| NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
| LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
| OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
| WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

License for the Castle Core binary included with the C# implementation:
File: Castle.Core.dll

Copyright (c) 2004-2015 Castle Project

License: Apache License version 2.0 (see above)
URL: https://opensource.org/licenses/Apache-2.0

License for the log4net binary included with the C# implementation:
File: log4net.dll

Copyright 2004-2015 The Apache Software Foundation.

License: Apache License version 2.0 (see above)

License for the m4 macros used by the C++ implementation:

Files:
* lang/c++/m4/m4_ax_boost_system.m4
 Copyright (c) 2008 Thomas Porschberg <thomas@randspringer.de>
 Copyright (c) 2008 Michael Tindal
 Copyright (c) 2008 Daniel Casimiro <dan.casimiro@gmail.com>
* lang/c++/m4/m4_ax_boost_asio.m4
 Copyright (c) 2008 Thomas Porschberg <thomas@randspringer.de>
 Copyright (c) 2008 Pete Greenwell <pete@mu.org>
* lang/c++/m4/m4_ax_boost_filesystem.m4
 Copyright (c) 2009 Thomas Porschberg <thomas@randspringer.de>
 Copyright (c) 2009 Michael Tindal
 Copyright (c) 2009 Roman Rybalko <libtorrent@romanr.info>
* lang/c++/m4/m4_ax_boost_thread.m4
 Copyright (c) 2009 Thomas Porschberg <thomas@randspringer.de>
 Copyright (c) 2009 Michael Tindal
* lang/c++/m4/m4_ax_boost_regex.m4
 Copyright (c) 2008 Thomas Porschberg <thomas@randspringer.de>
 Copyright (c) 2008 Michael Tindal
* lang/c++/m4/m4_ax_boost_base.m4
 Copyright (c) 2008 Thomas Porschberg <thomas@randspringer.de>

License text:
| Copying and distribution of this file, with or without modification, are
| permitted in any medium without royalty provided the copyright notice
| and this notice are preserved. This file is offered as-is, without any
| warranty.

License for the AVRO_BOOT_NO_TRAIT code in the C++ implementation:
File: lang/c++/api/Boost.hh

| Boost Software License - Version 1.0 - August 17th, 2003
|
| Permission is hereby granted, free of charge, to any person or organization
| obtaining a copy of the software and accompanying documentation covered by
| this license (the "Software") to use, reproduce, display, distribute,
| execute, and transmit the Software, and to prepare derivative works of the
| Software, and to permit third-parties to whom the Software is furnished to
| do so, all subject to the following:
|
| The copyright notices in the Software and this entire statement, including
| the above license grant, this restriction and the following disclaimer,
| must be included in all copies of the Software, in whole or in part, and
| all derivative works of the Software, unless such copies or derivative
| works are solely in the form of machine-executable object code generated by
| a source language processor.
|
| THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
| IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
| FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT
| SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE
| FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE,
| ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
| DEALINGS IN THE SOFTWARE.

License for jquery.tipsy.js, tipsy.js, and tipsy.css used by the Java IPC implementation:

Copyright (c) 2008 Jason Frame (jason@onehackoranother.com)

| Permission is hereby granted, free of charge, to any person obtaining a copy
| of this software and associated documentation files (the "Software"), to deal
| in the Software without restriction, including without limitation the rights
| to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
| copies of the Software, and to permit persons to whom the Software is
| furnished to do so, subject to the following conditions:
|
| The above copyright notice and this permission notice shall be included in
| all copies or substantial portions of the Software.
|
| THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
| IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
| FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
| AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
| LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,

| OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
| THE SOFTWARE.

License for protovis-r3.2.js used by the Java IPC implementation:

Copyright (c) 2010, Stanford Visualization Group
All rights reserved.

| Redistribution and use in source and binary forms, with or without modification,
| are permitted provided that the following conditions are met:
|
| * Redistributions of source code must retain the above copyright notice,
| this list of conditions and the following disclaimer.
|
| * Redistributions in binary form must reproduce the above copyright notice,
| this list of conditions and the following disclaimer in the documentation
| and/or other materials provided with the distribution.
|
| * Neither the name of Stanford University nor the names of its contributors
| may be used to endorse or promote products derived from this software
| without specific prior written permission.
|
| THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
| ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
| WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
| DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
| ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
| (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
| LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
| ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
| (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
| SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

License for g.Raphael 0.4.1 used by the Java IPC implementation:

Copyright (c) 2009 Dmitry Baranovskiy (http://g.raphaeljs.com)
Licensed under the MIT (https://www.opensource.org/licenses/mit-license.php) license.

License for jQuery v1.4.2 used by the Java IPC implementation:

Copyright 2010, John Resig
Dual licensed under the MIT or GPL Version 2 licenses.
https://jquery.org/license

jQuery includes Sizzle.js

https://sizzlejs.com/
Copyright 2010, The Dojo Foundation
Released under the MIT, BSD, and GPL Licenses.

Both are included under the terms of the MIT license:

| Permission is hereby granted, free of charge, to any person obtaining a copy
| of this software and associated documentation files (the "Software"), to deal
| in the Software without restriction, including without limitation the rights
| to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
| copies of the Software, and to permit persons to whom the Software is
| furnished to do so, subject to the following conditions:
|
| The above copyright notice and this permission notice shall be included in
| all copies or substantial portions of the Software.
|
| THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
| IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
| FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
| AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
| LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
| OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
| THE SOFTWARE.

License for portions of idl.jj in the Java compiler implementation:

Portions of idl.jj were modeled after the example Java 1.5
parser included with JavaCC. For those portions:

Copyright (c) 2006, Sun Microsystems, Inc.
All rights reserved.

| Redistribution and use in source and binary forms, with or without
| modification, are permitted provided that the following conditions are met:
|
| * Redistributions of source code must retain the above copyright notice,
| this list of conditions and the following disclaimer.
| * Redistributions in binary form must reproduce the above copyright
| notice, this list of conditions and the following disclaimer in the
| documentation and/or other materials provided with the distribution.
| * Neither the name of the Sun Microsystems, Inc. nor the names of its
| contributors may be used to endorse or promote products derived from
| this software without specific prior written permission.
|
| THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
| AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
| IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE

| ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
| LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
| CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
| SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
| INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
| CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
| ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
| THE POSSIBILITY OF SUCH DAMAGE.

Apache License
Version 2.0, January 2004
https://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
Work or Derivative Works thereof in any medium, with or without
modifications, and in Source or Object form, provided that You
meet the following conditions:

(a) You must give any other recipients of the Work or
Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices
stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works
that You distribute, all copyright, patent, trademark, and
attribution notices from the Source form of the Work,
excluding those notices that do not pertain to any part of
the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its
distribution, then any Derivative Works that You distribute must
include a readable copy of the attribution notices contained
within such NOTICE file, excluding those notices that do not
pertain to any part of the Derivative Works, in at least one
of the following places: within a NOTICE text file distributed
as part of the Derivative Works; within the Source form or
documentation, if provided along with the Derivative Works; or,
within a display generated by the Derivative Works, if and
wherever such third-party notices normally appear. The contents
of the NOTICE file are for informational purposes only and
do not modify the License. You may add Your own attribution
notices within Derivative Works that You distribute, alongside
or as an addendum to the NOTICE text from the Work, provided
that such additional attribution notices cannot be construed
as modifying the License.

You may add Your own copyright statement to Your modifications and
may provide additional or different license terms and conditions
for use, reproduction, or distribution of Your modifications, or
for any such Derivative Works as a whole, provided Your use,
reproduction, and distribution of the Work otherwise complies with
the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
any Contribution intentionally submitted for inclusion in the Work
by You to the Licensor shall be under the terms and conditions of
this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify
the terms of any separate license agreement you may have executed

with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate

comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

https://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

Mozilla Public License
Version 2.0

1. Definitions

1.1. Contributor
means each individual or legal entity that creates, contributes to the creation
of, or owns Covered Software.

1.2. Contributor Version
means the combination of the Contributions of others (if any) used by a
Contributor and that particular Contributors Contribution.

1.3. Contribution
means Covered Software of a particular Contributor.

1.4. Covered Software
means Source Code Form to which the initial Contributor has attached the notice
in Exhibit A, the Executable Form of such Source Code Form, and Modifications
of such Source Code Form, in each case including portions thereof.

1.5. Incompatible With Secondary Licenses
means

a. that the initial Contributor has attached the notice described in Exhibit B
to the Covered Software; or

b. that the Covered Software was made available under the terms of version 1.1
or earlier of the License, but not also under the terms of a Secondary License.

1.6. Executable Form
means any form of the work other than Source Code Form.

1.7. Larger Work
means a work that combines Covered Software with other material, in a separate file or files, that is not Covered Software.

1.8. License
means this document.

1.9. Licensable
means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently, any and all of the rights conveyed by this License.

1.10. Modifications
means any of the following:

a. any file in Source Code Form that results from an addition to, deletion from, or modification of the contents of Covered Software; or

b. any new file in Source Code Form that contains any Covered Software.

1.11. Patent Claims of a Contributor
means any patent claim(s), including without limitation, method, process, and apparatus claims, in any patent Licensable by such Contributor that would be infringed, but for the grant of the License, by the making, using, selling, offering for sale, having made, import, or transfer of either its Contributions or its Contributor Version.

1.12. Secondary License
means either the GNU General Public License, Version 2.0, the GNU Lesser General Public License, Version 2.1, the GNU Affero General Public License, Version 3.0, or any later versions of those licenses.

1.13. Source Code Form
means the form of the work preferred for making modifications.

1.14. You (or Your)
means an individual or a legal entity exercising rights under this License. For legal entities, You includes any entity that controls, is controlled by, or is under common control with You. For purposes of this definition, control means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. License Grants and Conditions

2.1. Grants

Each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:

a. under intellectual property rights (other than patent or trademark) Licensable by such Contributor to use, reproduce, make available, modify, display, perform, distribute, and otherwise exploit its Contributions, either on an unmodified basis, with Modifications, or as part of a Larger Work; and

b. under Patent Claims of such Contributor to make, use, sell, offer for sale, have made, import, and otherwise transfer either its Contributions or its Contributor Version.

2.2. Effective Date

The licenses granted in Section 2.1 with respect to any Contribution become effective for each Contribution on the date the Contributor first distributes such Contribution.

2.3. Limitations on Grant Scope

The licenses granted in this Section 2 are the only rights granted under this License. No additional rights or licenses will be implied from the distribution or licensing of Covered Software under this License. Notwithstanding Section 2.1(b) above, no patent license is granted by a Contributor:

a. for any code that a Contributor has removed from Covered Software; or

b. for infringements caused by: (i) Your and any other third partys modifications of Covered Software, or (ii) the combination of its Contributions with other software (except as part of its Contributor Version); or

c. under Patent Claims infringed by Covered Software in the absence of its Contributions.

This License does not grant any rights in the trademarks, service marks, or logos of any Contributor (except as may be necessary to comply with the notice requirements in Section 3.4).

2.4. Subsequent Licenses

No Contributor makes additional grants as a result of Your choice to distribute the Covered Software under a subsequent version of this License (see Section 10.2) or under the terms of a Secondary License (if permitted under the terms of Section 3.3).

2.5. Representation

Each Contributor represents that the Contributor believes its Contributions are its original creation(s) or it has sufficient rights to grant the rights to its Contributions conveyed by this License.

2.6. Fair Use

This License is not intended to limit any rights You have under applicable copyright doctrines of fair use, fair dealing, or other equivalents.

2.7. Conditions

Sections 3.1, 3.2, 3.3, and 3.4 are conditions of the licenses granted in Section 2.1.

3. Responsibilities

3.1. Distribution of Source Form

All distribution of Covered Software in Source Code Form, including any Modifications that You create or to which You contribute, must be under the terms of this License. You must inform recipients that the Source Code Form of the Covered Software is governed by the terms of this License, and how they can obtain a copy of this License. You may not attempt to alter or restrict the recipients rights in the Source Code Form.

3.2. Distribution of Executable Form

If You distribute Covered Software in Executable Form then:

a. such Covered Software must also be made available in Source Code Form, as described in Section 3.1, and You must inform recipients of the Executable Form how they can obtain a copy of such Source Code Form by reasonable means in a timely manner, at a charge no more than the cost of distribution to the recipient; and

b. You may distribute such Executable Form under the terms of this License, or sublicense it under different terms, provided that the license for the Executable Form does not attempt to limit or alter the recipients rights in the Source Code Form under this License.

3.3. Distribution of a Larger Work

You may create and distribute a Larger Work under terms of Your choice, provided that You also comply with the requirements of this License for the Covered Software. If the Larger Work is a combination of Covered Software with a work governed by one or more Secondary Licenses, and the Covered Software is

not Incompatible With Secondary Licenses, this License permits You to additionally distribute such Covered Software under the terms of such Secondary License(s), so that the recipient of the Larger Work may, at their option, further distribute the Covered Software under the terms of either this License or such Secondary License(s).

3.4. Notices

You may not remove or alter the substance of any license notices (including copyright notices, patent notices, disclaimers of warranty, or limitations of liability) contained within the Source Code Form of the Covered Software, except that You may alter any license notices to the extent required to remedy known factual inaccuracies.

3.5. Application of Additional Terms

You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However, You may do so only on Your own behalf, and not on behalf of any Contributor. You must make it absolutely clear that any such warranty, support, indemnity, or liability obligation is offered by You alone, and You hereby agree to indemnify every Contributor for any liability incurred by such Contributor as a result of warranty, support, indemnity or liability terms You offer. You may include additional disclaimers of warranty and limitations of liability specific to any jurisdiction.

4. Inability to Comply Due to Statute or Regulation

If it is impossible for You to comply with any of the terms of this License with respect to some or all of the Covered Software due to statute, judicial order, or regulation then You must: (a) comply with the terms of this License to the maximum extent possible; and (b) describe the limitations and the code they affect. Such description must be placed in a text file included with all distributions of the Covered Software under this License. Except to the extent prohibited by statute or regulation, such description must be sufficiently detailed for a recipient of ordinary skill to be able to understand it.

5. Termination

5.1. The rights granted under this License will terminate automatically if You fail to comply with any of its terms. However, if You become compliant, then the rights granted under this License from a particular Contributor are reinstated (a) provisionally, unless and until such Contributor explicitly and finally terminates Your grants, and (b) on an ongoing basis, if such Contributor fails to notify You of the non-compliance by some reasonable means prior to 60 days after You have come back into compliance. Moreover, Your grants from a particular Contributor are reinstated on an ongoing basis if such Contributor notifies You of the non-compliance by some reasonable means, this

is the first time You have received notice of non-compliance with this License from such Contributor, and You become compliant prior to 30 days after Your receipt of the notice.

5.2. If You initiate litigation against any entity by asserting a patent infringement claim (excluding declaratory judgment actions, counter-claims, and cross-claims) alleging that a Contributor Version directly or indirectly infringes any patent, then the rights granted to You by any and all Contributors for the Covered Software under Section 2.1 of this License shall terminate.

5.3. In the event of termination under Sections 5.1 or 5.2 above, all end user license agreements (excluding distributors and resellers) which have been validly granted by You or Your distributors under this License prior to termination shall survive termination.

6. Disclaimer of Warranty

Covered Software is provided under this License on an as is basis, without warranty of any kind, either expressed, implied, or statutory, including, without limitation, warranties that the Covered Software is free of defects, merchantable, fit for a particular purpose or non-infringing. The entire risk as to the quality and performance of the Covered Software is with You. Should any Covered Software prove defective in any respect, You (not any Contributor) assume the cost of any necessary servicing, repair, or correction. This disclaimer of warranty constitutes an essential part of this License. No use of any Covered Software is authorized under this License except under this disclaimer.

7. Limitation of Liability

Under no circumstances and under no legal theory, whether tort (including negligence), contract, or otherwise, shall any Contributor, or anyone who distributes Covered Software as permitted above, be liable to You for any direct, indirect, special, incidental, or consequential damages of any character including, without limitation, damages for lost profits, loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses, even if such party shall have been informed of the possibility of such damages. This limitation of liability shall not apply to liability for death or personal injury resulting from such partys negligence to the extent applicable law prohibits such limitation. Some jurisdictions do not allow the exclusion or limitation of incidental or consequential damages, so this exclusion and limitation may not apply to You.

8. Litigation

Any litigation relating to this License may be brought only in the courts of a jurisdiction where the defendant maintains its principal place of business and

such litigation shall be governed by laws of that jurisdiction, without reference to its conflict-of-law provisions. Nothing in this Section shall prevent a partys ability to bring cross-claims or counter-claims.

9. Miscellaneous

This License represents the complete agreement concerning the subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not be used to construe this License against a Contributor.

10. Versions of the License

10.1. New Versions

Mozilla Foundation is the license steward. Except as provided in Section 10.3, no one other than the license steward has the right to modify or publish new versions of this License. Each version will be given a distinguishing version number.

10.2. Effect of New Versions

You may distribute the Covered Software under the terms of the version of the License under which You originally received the Covered Software, or under the terms of any subsequent version published by the license steward.

10.3. Modified Versions

If you create software not governed by this License, and you want to create a new license for such software, you may create and use a modified version of this License if you rename the license and remove any references to the name of the license steward (except to note that such modified license differs from this License).

10.4. Distributing Source Code Form that is Incompatible With Secondary Licenses

If You choose to distribute Source Code Form that is Incompatible With Secondary Licenses under the terms of this version of the License, the notice described in Exhibit B of this License must be attached.

Exhibit A - Source Code Form License Notice

> This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at https://mozilla.org/MPL/2.0/.

If it is not possible or desirable to put the notice in a particular file, then
You may include the notice in a location (such as a LICENSE file in a relevant
directory) where a recipient would be likely to look for such a notice.

You may add additional accurate notices of copyright ownership.

Exhibit B - Incompatible With Secondary Licenses Notice

This Source Code Form is Incompatible With Secondary Licenses, as defined
by the Mozilla Public License, v. 2.0.

Apache License
Version 2.0, January 2004
https://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a

copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object
form, that is based on (or derived from) the Work and for which the
editorial revisions, annotations, elaborations, or other modifications
represent, as a whole, an original work of authorship. For the purposes
of this License, Derivative Works shall not include works that remain
separable from, or merely link (or bind by name) to the interfaces of,
the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity
on behalf of whom a Contribution has been received by Licensor and
subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
copyright license to reproduce, prepare Derivative Works of,
publicly display, publicly perform, sublicense, and distribute the
Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
(except as stated in this section) patent license to make, have made,
use, offer to sell, sell, import, and otherwise transfer the Work,
where such license applies only to those patent claims licensable
by such Contributor that are necessarily infringed by their
Contribution(s) alone or by combination of their Contribution(s)
with the Work to which such Contribution(s) was submitted. If You
institute patent litigation against any entity (including a
cross-claim or counterclaim in a lawsuit) alleging that the Work
or a Contribution incorporated within the Work constitutes direct

or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of

this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify
the terms of any separate license agreement you may have executed
with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
names, trademarks, service marks, or product names of the Licensor,
except as required for reasonable and customary use in describing the
origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
agreed to in writing, Licensor provides the Work (and each
Contributor provides its Contributions) on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied, including, without limitation, any warranties or conditions
of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
PARTICULAR PURPOSE. You are solely responsible for determining the
appropriateness of using or redistributing the Work and assume any
risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
whether in tort (including negligence), contract, or otherwise,
unless required by applicable law (such as deliberate and grossly
negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special,
incidental, or consequential damages of any character arising as a
result of this License or out of the use or inability to use the
Work (including but not limited to damages for loss of goodwill,
work stoppage, computer failure or malfunction, or any and all
other commercial damages or losses), even if such Contributor
has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
the Work or Derivative Works thereof, You may choose to offer,
and charge a fee for, acceptance of support, warranty, indemnity,
or other liability obligations and/or rights consistent with this
License. However, in accepting such obligations, You may act only
on Your own behalf and on Your sole responsibility, not on behalf
of any other Contributor, and only if You agree to indemnify,
defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following

boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

https://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

License for the Json.NET binary included with the C# implementation:
File: Newtonsoft.Json.dll

Copyright (c) 2007 James Newton-King

| Permission is hereby granted, free of charge, to any person obtaining
| a copy of this software and associated documentation files (the
| "Software"), to deal in the Software without restriction, including
| without limitation the rights to use, copy, modify, merge, publish,
| distribute, sublicense, and/or sell copies of the Software, and to
| permit persons to whom the Software is furnished to do so, subject to
| the following conditions:
|
| The above copyright notice and this permission notice shall be
| included in all copies or substantial portions of the Software.
|
| THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
| EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
| MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
| NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
| LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
| OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
| WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

License for the Castle Core binary included with the C# implementation:
File: Castle.Core.dll

Copyright (c) 2004-2015 Castle Project

License: Apache License version 2.0 (see above)
URL: https://opensource.org/licenses/Apache-2.0

License for the log4net binary included with the C# implementation:
File: log4net.dll

Copyright 2004-2015 The Apache Software Foundation.

License: Apache License version 2.0 (see above)
Apache Avro
Copyright 2010 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (https://www.apache.org/).
Apache Avro
Copyright 2010-2021 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (https://www.apache.org/).

Apache License
Version 2.0, January 2004
https://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object
form, that is based on (or derived from) the Work and for which the
editorial revisions, annotations, elaborations, or other modifications
represent, as a whole, an original work of authorship. For the purposes
of this License, Derivative Works shall not include works that remain
separable from, or merely link (or bind by name) to the interfaces of,
the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity
on behalf of whom a Contribution has been received by Licensor and
subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
copyright license to reproduce, prepare Derivative Works of,
publicly display, publicly perform, sublicense, and distribute the

Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside

or as an addendum to the NOTICE text from the Work, provided
that such additional attribution notices cannot be construed
as modifying the License.

You may add Your own copyright statement to Your modifications and
may provide additional or different license terms and conditions
for use, reproduction, or distribution of Your modifications, or
for any such Derivative Works as a whole, provided Your use,
reproduction, and distribution of the Work otherwise complies with
the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
any Contribution intentionally submitted for inclusion in the Work
by You to the Licensor shall be under the terms and conditions of
this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify
the terms of any separate license agreement you may have executed
with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
names, trademarks, service marks, or product names of the Licensor,
except as required for reasonable and customary use in describing the
origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
agreed to in writing, Licensor provides the Work (and each
Contributor provides its Contributions) on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied, including, without limitation, any warranties or conditions
of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
PARTICULAR PURPOSE. You are solely responsible for determining the
appropriateness of using or redistributing the Work and assume any
risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
whether in tort (including negligence), contract, or otherwise,
unless required by applicable law (such as deliberate and grossly
negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special,
incidental, or consequential damages of any character arising as a
result of this License or out of the use or inability to use the
Work (including but not limited to damages for loss of goodwill,
work stoppage, computer failure or malfunction, or any and all
other commercial damages or losses), even if such Contributor
has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
the Work or Derivative Works thereof, You may choose to offer,

and charge a fee for, acceptance of support, warranty, indemnity,
or other liability obligations and/or rights consistent with this
License. However, in accepting such obligations, You may act only
on Your own behalf and on Your sole responsibility, not on behalf
of any other Contributor, and only if You agree to indemnify,
defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following
boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

https://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

License for the AVRO_BOOT_NO_TRAIT code in the C++ implementation:
File: lang/c++/api/Boost.hh

| Boost Software License - Version 1.0 - August 17th, 2003
|
| Permission is hereby granted, free of charge, to any person or organization
| obtaining a copy of the software and accompanying documentation covered by
| this license (the "Software") to use, reproduce, display, distribute,
| execute, and transmit the Software, and to prepare derivative works of the
| Software, and to permit third-parties to whom the Software is furnished to
| do so, all subject to the following:

|
| The copyright notices in the Software and this entire statement, including
| the above license grant, this restriction and the following disclaimer,
| must be included in all copies of the Software, in whole or in part, and
| all derivative works of the Software, unless such copies or derivative
| works are solely in the form of machine-executable object code generated by
| a source language processor.
|
| THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
| IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
| FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT
| SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE
| FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE,
| ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
| DEALINGS IN THE SOFTWARE.

License for jQuery v1.7.1 used in the C# documentation

Copyright 2010-2011, John Resig
Dual licensed under the MIT or GPL Version 2 licenses.
https://jquery.org/license

jQuery includes Sizzle.js
https://sizzlejs.com/
Copyright 2010-2011, The Dojo Foundation
Released under the MIT, BSD, and GPL Licenses.

Both are included under the terms of the MIT license:

| Permission is hereby granted, free of charge, to any person obtaining a copy
| of this software and associated documentation files (the "Software"), to deal
| in the Software without restriction, including without limitation the rights
| to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
| copies of the Software, and to permit persons to whom the Software is
| furnished to do so, subject to the following conditions:
|
| The above copyright notice and this permission notice shall be included in
| all copies or substantial portions of the Software.
|
| THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
| IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
| FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
| AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
| LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
| OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
| THE SOFTWARE.

License for portions of idl.jj in the Java compiler implementation:

Portions of idl.jj were modeled after the example Java 1.5
parser included with JavaCC. For those portions:

| Redistribution and use in source and binary forms, with or without
| modification, are permitted provided that the following conditions are met:
|
| * Redistributions of source code must retain the above copyright notice,
| this list of conditions and the following disclaimer.
| * Redistributions in binary form must reproduce the above copyright
| notice, this list of conditions and the following disclaimer in the
| documentation and/or other materials provided with the distribution.
| * Neither the name of the Sun Microsystems, Inc. nor the names of its
| contributors may be used to endorse or promote products derived from
| this software without specific prior written permission.
|
| THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
| AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
| IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
| ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
| LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
| CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
| SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
| INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
| CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
| ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
| THE POSSIBILITY OF SUCH DAMAGE.

License for prototype.js included in the Avro documentation:

Prototype JavaScript framework, version 1.4.0_pre4
(c) 2005 Sam Stephenson <sam@conio.net>

| Prototype is freely distributable under the terms of an MIT-style license.
|
| For details, see the Prototype web site: http://prototype.conio.net/

For a copy of the MIT license text, see above.

License for Apache Forrest (skin), included in the Avro documentation:

Copyright: 2009-2015 The Apache Software Foundation
License: https://www.apache.org/licenses/LICENSE-2.0 (see above)

License for Doxygen-generated documentation for the C++ and C# implementations:

Copyright 1997-2015 by Dimitri van Heesch.

| Doxygen license
|
| Permission to use, copy, modify, and distribute this software and its
| documentation under the terms of the GNU General Public License is hereby
| granted. No representations are made about the suitability of this software for
| any purpose. It is provided "as is" without express or implied warranty. See
| the GNU General Public License for more details.
|
| Documents produced by doxygen are derivative works derived from the input
| used in their production; they are not affected by this license.
Apache Avro
Copyright 2010-2015 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (https://www.apache.org/).
Apache Avro
Copyright 2011-2015 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (https://www.apache.org/).
---
title: "License"
linkTitle: "License"
weight: 3
manualLink: https://www.apache.org/licenses/

<!--

Licensed to the Apache Software Foundation (ASF) under one
or more contributor license agreements. See the NOTICE file
distributed with this work for additional information
regarding copyright ownership. The ASF licenses this file
to you under the Apache License, Version 2.0 (the
"License"); you may not use this file except in compliance
with the License. You may obtain a copy of the License at

  https://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing,

software distributed under the License is distributed on an
"AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
KIND, either express or implied. See the License for the
specific language governing permissions and limitations
under the License.

-->

Apache Avro project is licensed under [Apache Software License 2.0](https://www.apache.org/licenses/LICENSE-2.0)
Apache Avro
Copyright 2010-2015 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (https://www.apache.org/).

Based upon the representations of upstream licensors, it is understood that
portions of the mapreduce API included in the Java implementation are licensed
from various contributors under one or more contributor license agreements to
Odiago, Inc. and were then contributed by Odiago to Apache Avro, which has now
made them available under the Apache 2.0 license. The original file header text
is:

| Licensed to Odiago, Inc. under one or more contributor license
| agreements. See the NOTICE file distributed with this work for
| additional information regarding copyright ownership. Odiago, Inc.
| licenses this file to you under the Apache License, Version 2.0
| (the "License"); you may not use this file except in compliance
| with the License. You may obtain a copy of the License at
|
| https://www.apache.org/licenses/LICENSE-2.0
|
| Unless required by applicable law or agreed to in writing, software
| distributed under the License is distributed on an "AS IS" BASIS,
| WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
| implied. See the License for the specific language governing
| permissions and limitations under the License.

The Odiago NOTICE at the time of the contribution:

| This product includes software developed by Odiago, Inc.
| (https://www.wibidata.com).

Apache Commons compress includes the following in its NOTICE file:

| Apache Commons Compress
| Copyright 2002-2014 The Apache Software Foundation
|

| This product includes software developed at
| The Apache Software Foundation (https://www.apache.org/).
|
| The files in the package org.apache.commons.compress.archivers.sevenz
| were derived from the LZMA SDK, version 9.20 (C/ and CPP/7zip/),
| which has been placed in the public domain:
|
| "LZMA SDK is placed in the public domain." (https://www.7-zip.org/sdk.html)

Apache Commons codec includes the following in its NOTICE file:

| Apache Commons Codec
| Copyright 2002-2015 The Apache Software Foundation
|
| This product includes software developed at
| The Apache Software Foundation (https://www.apache.org/).
|
| The content of package org.apache.commons.codec.language.bm has been translated
| from the original php source code available at https://stevemorse.org/phoneticinfo.htm
| with permission from the original authors.
| Original source copyright:
| Copyright (c) 2008 Alexander Beider & Stephen P. Morse.

Apache Commons lang includes the following in its NOTICE file:

| Apache Commons Lang
| Copyright 2001-2011 The Apache Software Foundation
|
| This product includes software developed by
| The Apache Software Foundation (https://www.apache.org/).

Apache Velocity includes the following in its NOTICE file:

| Apache Velocity
| Copyright (C) 2000-2007 The Apache Software Foundation
|
| This product includes software developed at
| The Apache Software Foundation (https://www.apache.org/).

Apache Commons collections includes the following in its NOTICE file:

| Apache Commons Collections
| Copyright 2001-2008 The Apache Software Foundation
|
| This product includes software developed by
| The Apache Software Foundation (https://www.apache.org/).

Apache Commons math includes the following in its NOTICE file:

| ==
|
| The BracketFinder (package org.apache.commons.math3.optimization.univariate)
| and PowellOptimizer (package org.apache.commons.math3.optimization.general)
| classes are based on the Python code in module "optimize.py" (version 0.5)
| developed by Travis E. Oliphant for the SciPy library (https://www.scipy.org/)
| Copyright 2003-2009 SciPy Developers.
| ==
|
| The LinearConstraint, LinearObjectiveFunction, LinearOptimizer,
| RelationShip, SimplexSolver and SimplexTableau classes in package
| org.apache.commons.math3.optimization.linear include software developed by
| Benjamin McCann (https://www.benmccann.com) and distributed with
| the following copyright: Copyright 2009 Google Inc.
| ==
|
| This product includes software developed by the
| University of Chicago, as Operator of Argonne National
| Laboratory.
| The LevenbergMarquardtOptimizer class in package
| org.apache.commons.math3.optimization.general includes software
| translated from the lmder, lmpar and qrsolv Fortran routines
| from the Minpack package
| Minpack Copyright Notice (1999) University of Chicago. All rights reserved
| ==
|
| The GraggBulirschStoerIntegrator class in package
| org.apache.commons.math3.ode.nonstiff includes software translated
| from the odex Fortran routine developed by E. Hairer and G. Wanner.
| Original source copyright:
| Copyright (c) 2004, Ernst Hairer
| ==
|
| The EigenDecompositionImpl class in package
| org.apache.commons.math3.linear includes software translated
| from some LAPACK Fortran routines. Original source copyright:
| Copyright (c) 1992-2008 The University of Tennessee. All rights reserved.
| ==
|
| The MersenneTwister class in package org.apache.commons.math3.random
| includes software translated from the 2002-01-26 version of
| the Mersenne-Twister generator written in C by Makoto Matsumoto and Takuji
| Nishimura. Original source copyright:
| Copyright (C) 1997 - 2002, Makoto Matsumoto and Takuji Nishimura,
| All rights reserved
| ==
|

| The LocalizedFormatsTest class in the unit tests is an adapted version of
| the OrekitMessagesTest class from the orekit library distributed under the
| terms of the Apache 2 licence. Original source copyright:
| Copyright 2010 CS Systmes d'Information
| ==
|
| The HermiteInterpolator class and its corresponding test have been imported from
| the orekit library distributed under the terms of the Apache 2 licence. Original
| source copyright:
| Copyright 2010-2012 CS Systmes d'Information
| ==
|
| The creation of the package "o.a.c.m.analysis.integration.gauss" was inspired
| by an original code donated by Sbastien Brisard.
| ==
|
| The complete text of licenses and disclaimers associated with the the original
| sources enumerated above at the time of code translation are in the LICENSE.txt
| file.

Jetty 6.1.26 includes the following in its NOTICE file:

| ==
| Jetty Web Container
| Copyright 1995-2009 Mort Bay Consulting Pty Ltd
| ==
|
| The Jetty Web Container is Copyright Mort Bay Consulting Pty Ltd
| unless otherwise noted. It is licensed under the apache 2.0
| license.
|
| The javax.servlet package used by Jetty is copyright
| Sun Microsystems, Inc and Apache Software Foundation. It is
| distributed under the Common Development and Distribution License.
| You can obtain a copy of the license at
| https://glassfish.dev.java.net/public/CDDLv1.0.html.
|
| The UnixCrypt.java code ~Implements the one way cryptography used by
| Unix systems for simple password protection. Copyright 1996 Aki Yoshida,
| modified April 2001 by Iris Van den Broeke, Daniel Deville.
| Permission to use, copy, modify and distribute UnixCrypt
| for non-commercial or commercial purposes and without fee is
| granted provided that the copyright notice appears in all copies.
|
| The default JSP implementation is provided by the Glassfish JSP engine
| from project Glassfish https://glassfish.dev.java.net. Copyright 2005
| Sun Microsystems, Inc. and portions Copyright Apache Software Foundation.
|

| Some portions of the code are Copyright:
| 2006 Tim Vernum
| 1999 Jason Gilbert.
|
| The jboss integration module contains some LGPL code.
| [JBOSS INTEGRATION IS NOT INCLUDED IN AVRO TOOLS.]
|
| The win32 Java Service Wrapper (v3.2.3) is Copyright (c) 1999, 2006
| Tanuki Software, Inc. and 2001 Silver Egg Technology. It is
| covered by an open license which is viewable at
| http://svn.codehaus.org/jetty/jetty/branches/jetty-6.1/extras/win32service/LICENSE.txt
| [WIN32 WRAPPER IS NOT INCLUDED IN AVRO TOOLS.]

Netty 3.5.13.Final includes the following in its NOTICE file:

| The Netty Project
| =================
|
| Please visit the Netty web site for more information:
|
| * https://netty.io/
|
| Copyright 2011 The Netty Project
|
| The Netty Project licenses this file to you under the Apache License,
| version 2.0 (the "License"); you may not use this file except in compliance
| with the License. You may obtain a copy of the License at:
|
| https://www.apache.org/licenses/LICENSE-2.0
|
| Unless required by applicable law or agreed to in writing, software
| distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
| WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
| License for the specific language governing permissions and limitations
| under the License.
|
| Also, please refer to each LICENSE.<component>.txt file, which is located in
| the 'license' directory of the distribution file, for the license terms of the
| components that this product depends on.
|
| ---
| This product contains the extensions to Java Collections Framework which has
| been derived from the works by JSR-166 EG, Doug Lea, and Jason T. Greene:
|
| * LICENSE:
| * license/LICENSE.jsr166y.txt (Public Domain)
| * HOMEPAGE:

| * http://gee.cs.oswego.edu/cgi-bin/viewcvs.cgi/jsr166/
| * http://viewvc.jboss.org/cgi-bin/viewvc.cgi/jbosscache/experimental/jsr166/
|
| This product contains a modified version of Robert Harder's Public Domain
| Base64 Encoder and Decoder, which can be obtained at:
|
| * LICENSE:
| * license/LICENSE.base64.txt (Public Domain)
| * HOMEPAGE:
| * http://iharder.sourceforge.net/current/java/base64/
|
| This product contains a modified version of 'JZlib', a re-implementation of
| zlib in pure Java, which can be obtained at:
|
| * LICENSE:
| * license/LICENSE.jzlib.txt (BSD Style License)
| * HOMEPAGE:
| * http://www.jcraft.com/jzlib/
|
| This product optionally depends on 'Protocol Buffers', Google's data
| interchange format, which can be obtained at:
|
| * LICENSE:
| * license/LICENSE.protobuf.txt (New BSD License)
| * HOMEPAGE:
| * https://code.google.com/p/protobuf/
|
| This product optionally depends on 'SLF4J', a simple logging facade for Java,
| which can be obtained at:
|
| * LICENSE:
| * license/LICENSE.slf4j.txt (MIT License)
| * HOMEPAGE:
| * https://www.slf4j.org/
|
| This product optionally depends on 'Apache Commons Logging', a logging
| framework, which can be obtained at:
|
| * LICENSE:
| * license/LICENSE.commons-logging.txt (Apache License 2.0)
| * HOMEPAGE:
| * https://commons.apache.org/logging/
|
| This product optionally depends on 'Apache Log4J', a logging framework,
| which can be obtained at:
|
| * LICENSE:
| * license/LICENSE.log4j.txt (Apache License 2.0)

| * HOMEPAGE:
| * https://logging.apache.org/log4j/
|
| This product optionally depends on 'JBoss Logging', a logging framework,
| which can be obtained at:
|
| * LICENSE:
| * license/LICENSE.jboss-logging.txt (GNU LGPL 2.1)
| * HOMEPAGE:
| * https://anonsvn.jboss.org/repos/common/common-logging-spi/
|
| [JBOSS LOGGING IS NOT INCLUDED IN AVRO TOOLS.]
|
| This product optionally depends on 'Apache Felix', an open source OSGi
| framework implementation, which can be obtained at:
|
| * LICENSE:
| * license/LICENSE.felix.txt (Apache License 2.0)
| * HOMEPAGE:
| * https://felix.apache.org/
|
| [FELIX IS NOT INCLUDED IN AVRO TOOLS.]
|
| This product optionally depends on 'Webbit', a Java event based
| WebSocket and HTTP server:
|
| * LICENSE:
| * license/LICENSE.webbit.txt (BSD License)
| * HOMEPAGE:
| * https://github.com/joewalnes/webbit
|
| [WEBBIT IS NOT INCLUDED IN AVRO TOOLS.]

Apache Commons CLI includes the following in its NOTICE file:

| Apache Commons CLI
| Copyright 2001-2009 The Apache Software Foundation
|
| This product includes software developed by
| The Apache Software Foundation (https://www.apache.org/).

Apache Commons logging includes the following in its NOTICE file:

| Apache Commons Logging
| Copyright 2003-2007 The Apache Software Foundation
|
| This product includes software developed by
| The Apache Software Foundation (https://www.apache.org/).

Apache Commons HttpClient includes the following in its NOTICE file:

| Apache Jakarta HttpClient
| Copyright 1999-2007 The Apache Software Foundation
|
| This product includes software developed by
| The Apache Software Foundation (https://www.apache.org/).

Apache Hadoop includes the following in its NOTICE file:

| This product includes software developed by The Apache Software
| Foundation (https://www.apache.org/).

Apache Commons IO includes the following in its NOTICE file:

| Apache Commons IO
| Copyright 2002-2012 The Apache Software Foundation
|
| This product includes software developed by
| The Apache Software Foundation (https://www.apache.org/).

Apache Commons Net includes the following in its NOTICE file:

| Apache Commons Net
| Copyright 2001-2012 The Apache Software Foundation
|
| This product includes software developed by
| The Apache Software Foundation (https://www.apache.org/).

Apache Log4j includes the following in its NOTICE file:

| Apache log4j
| Copyright 2010 The Apache Software Foundation
|
| This product includes software developed at
| The Apache Software Foundation (https://www.apache.org/).

Apache Commons configuration includes the following in its NOTICE file:

| Apache Commons Configuration
| Copyright 2001-2008 The Apache Software Foundation
|
| This product includes software developed by
| The Apache Software Foundation (https://www.apache.org/).

Apache Commons digester includes the following in its NOTICE file:

| Apache Jakarta Commons Digester
| Copyright 2001-2006 The Apache Software Foundation
|
| This product includes software developed by
| The Apache Software Foundation (https://www.apache.org/).

Apache Commons beanutils includes the following in its NOTICE file:

| Apache Commons BeanUtils
| Copyright 2000-2008 The Apache Software Foundation
|
| This product includes software developed by
| The Apache Software Foundation (https://www.apache.org/).

Apache Directory includes the following in its NOTICE file:

| ApacheDS
| Copyright 2003-2015 The Apache Software Foundation
|
| This product includes software developed at
| The Apache Software Foundation (https://www.apache.org/).

Apache Zookeeper includes the following in its NOTICE file:

| Apache ZooKeeper
| Copyright 2009-2014 The Apache Software Foundation
|
| This product includes software developed at
| The Apache Software Foundation (https://www.apache.org/).
Apache Avro
Copyright 2010-2015 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (https://www.apache.org/).

This library was original developed by Yann Kerherve with the following
copyright notice:

| Copyright (C) 2010 Yann Kerherve. All rights reserved.
Apache Avro
Copyright 2010 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (https://www.apache.org/).

Based upon the representations of upstream licensors, it is understood that
portions of the mapreduce API included in the Java implementation are licensed
from various contributors under one or more contributor license agreements to

Odiago, Inc. and were then contributed by Odiago to Apache Avro, which has now made them available under the Apache 2.0 license. The original file header text is:

| Licensed to Odiago, Inc. under one or more contributor license
| agreements. See the NOTICE file distributed with this work for
| additional information regarding copyright ownership. Odiago, Inc.
| licenses this file to you under the Apache License, Version 2.0
| (the "License"); you may not use this file except in compliance
| with the License. You may obtain a copy of the License at
|
| https://www.apache.org/licenses/LICENSE-2.0
|
| Unless required by applicable law or agreed to in writing, software
| distributed under the License is distributed on an "AS IS" BASIS,
| WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
| implied. See the License for the specific language governing
| permissions and limitations under the License.

The Odiago NOTICE at the time of the contribution:

| This product includes software developed by Odiago, Inc.
| (https://www.wibidata.com).
Apache Avro
Copyright 2010-2022 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (https://www.apache.org/).

Based upon the representations of upstream licensors, it is understood that portions of the mapreduce API included in the Java implementation are licensed from various contributors under one or more contributor license agreements to Odiago, Inc. and were then contributed by Odiago to Apache Avro, which has now made them available under the Apache 2.0 license. The original file header text is:

| Licensed to Odiago, Inc. under one or more contributor license
| agreements. See the NOTICE file distributed with this work for
| additional information regarding copyright ownership. Odiago, Inc.
| licenses this file to you under the Apache License, Version 2.0
| (the "License"); you may not use this file except in compliance
| with the License. You may obtain a copy of the License at
|
| https://www.apache.org/licenses/LICENSE-2.0
|
| Unless required by applicable law or agreed to in writing, software
| distributed under the License is distributed on an "AS IS" BASIS,
| WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or

| implied. See the License for the specific language governing
| permissions and limitations under the License.

|--
| This product includes software developed by The Docsy Authors.
| (https://www.docsy.dev/).
|
| This product includes software developed at
| The Apache Software Foundation (https://www.apache.org/).
|
| See also the file LICENSE.txt
|
| --
| The purpose of this NOTICE.txt file is to contain notices that are
| required by the copyright owner and their license. Some of the
| accompanying products have an attribution requirement, so see below.
| Other accompanying products do not require attribution, so are not listed.
|
| --

Apache License
Version 2.0, January 2004
https://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made,

use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions

for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability

incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following
boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

https://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

License for msinttypes.h and msstdint.h used in the C implementation:
Source from:
https://code.google.com/p/msinttypes/downloads/detail?name=msinttypes-r26.zip

Copyright (c) 2006-2008 Alexander Chemeris

| Redistribution and use in source and binary forms, with or without
| modification, are permitted provided that the following conditions are met:
|
| 1. Redistributions of source code must retain the above copyright notice,
| this list of conditions and the following disclaimer.
|
| 2. Redistributions in binary form must reproduce the above copyright
| notice, this list of conditions and the following disclaimer in the
| documentation and/or other materials provided with the distribution.
|
| 3. The name of the author may be used to endorse or promote products

| derived from this software without specific prior written permission.
|
| THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR IMPLIED
| WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
| MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
| EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
| SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
| PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
| OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
| WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
| OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
| ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

License for st.c and st.h used in the C implementation:

| This is a public domain general purpose hash table package written by
| Peter Moore @ UCB.

License for Dirent API for Microsoft Visual Studio used in the C implementation:
Source from:
http://www.softagalleria.net/download/dirent/dirent-1.11.zip

Copyright (C) 2006 Toni Ronkko

| Permission is hereby granted, free of charge, to any person obtaining
| a copy of this software and associated documentation files (the
| ``Software''), to deal in the Software without restriction, including
| without limitation the rights to use, copy, modify, merge, publish,
| distribute, sublicense, and/or sell copies of the Software, and to
| permit persons to whom the Software is furnished to do so, subject to
| the following conditions:
|
| The above copyright notice and this permission notice shall be included
| in all copies or substantial portions of the Software.
|
| THE SOFTWARE IS PROVIDED ``AS IS'', WITHOUT WARRANTY OF ANY KIND, EXPRESS
| OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
| MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
| IN NO EVENT SHALL TONI RONKKO BE LIABLE FOR ANY CLAIM, DAMAGES OR
| OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
| ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
| OTHER DEALINGS IN THE SOFTWARE.

Apache License
Version 2.0, January 2004
https://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner

or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works
that You distribute, all copyright, patent, trademark, and
attribution notices from the Source form of the Work,
excluding those notices that do not pertain to any part of
the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its
distribution, then any Derivative Works that You distribute must
include a readable copy of the attribution notices contained
within such NOTICE file, excluding those notices that do not
pertain to any part of the Derivative Works, in at least one
of the following places: within a NOTICE text file distributed
as part of the Derivative Works; within the Source form or
documentation, if provided along with the Derivative Works; or,
within a display generated by the Derivative Works, if and
wherever such third-party notices normally appear. The contents
of the NOTICE file are for informational purposes only and
do not modify the License. You may add Your own attribution
notices within Derivative Works that You distribute, alongside
or as an addendum to the NOTICE text from the Work, provided
that such additional attribution notices cannot be construed
as modifying the License.

You may add Your own copyright statement to Your modifications and
may provide additional or different license terms and conditions
for use, reproduction, or distribution of Your modifications, or
for any such Derivative Works as a whole, provided Your use,
reproduction, and distribution of the Work otherwise complies with
the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
any Contribution intentionally submitted for inclusion in the Work
by You to the Licensor shall be under the terms and conditions of
this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify
the terms of any separate license agreement you may have executed
with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
names, trademarks, service marks, or product names of the Licensor,
except as required for reasonable and customary use in describing the
origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
agreed to in writing, Licensor provides the Work (and each
Contributor provides its Contributions) on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied, including, without limitation, any warranties or conditions

of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

https://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

License for portions of idl.jj in the Java compiler implementation:

Portions of idl.jj were modeled after the example Java 1.5
parser included with JavaCC. For those portions:

| Redistribution and use in source and binary forms, with or without
| modification, are permitted provided that the following conditions are met:
|
| * Redistributions of source code must retain the above copyright notice,
| this list of conditions and the following disclaimer.
| * Redistributions in binary form must reproduce the above copyright
| notice, this list of conditions and the following disclaimer in the
| documentation and/or other materials provided with the distribution.
| * Neither the name of the Sun Microsystems, Inc. nor the names of its
| contributors may be used to endorse or promote products derived from
| this software without specific prior written permission.
|
| THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
| AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
| IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
| ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
| LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
| CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
| SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
| INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
| CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
| ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
| THE POSSIBILITY OF SUCH DAMAGE.