@

# GNU gcj

Tom Tromey

For the GCC-3.2 Version*

Published by the Free Software Foundation
59 Temple Place - Suite 330
Boston, MA 02111-1307, USA

目次

# 導入

このマニュアルでは、Java プログラミング言語用の GNU コンパイラ `gcj`の使い方を説明します。`gcj`は、クラスファイル (‘`.class`’)、オブジェクトファイルの両方を生成することができます。また、Java ソースコード、クラスファイル (‘`.class`’) の両方を読み込むことができます。

# GNU GENERAL PUBLIC LICENSE

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) *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., 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.

# GNU Free Documentation License

Version 1.1, March 2000



0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. In any section entitled "Acknowledgments" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgments and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant

Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgments", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document 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.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation 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. See `http://www.gnu.org/copyleft/`.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

## ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (C) *year* *your name*.
> Permission is granted to copy, distribute and/or modify this document
> under the terms of the GNU Free Documentation License, Version 1.1
> or any later version published by the Free Software Foundation;
> with the Invariant Sections being *list their titles*, with the
> Front-Cover Texts being *list*, and with the Back-Cover Texts being *list*.
> A copy of the license is included in the section entitled ``GNU
> Free Documentation License''.

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being *list*"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

# 1 gcj の実行

gcjは、gccに対するフロントエンドの 1 つに過ぎませんので、gcc と同一のオプションを数多くサポートしています。gcc のマニュアル Using the GNU Compiler Collection (GCC) の Option Summary のセクションを参照してください。このマニュアルでは、gcjに固有のオプションのみを説明します。

## 1.1 入出力ファイル

gcjコマンドは、いくつかのオプションと特有のファイル名を持つという点で、gccコマンドに類似しています。以下のような入力ファイル名がサポートされています。

*file*`.java` Java ソースファイル。

*file*`.class` Java バイトコードファイル。

*file*`.zip`
*file*`.jar` １つ以上のコンパイル済みクラスファイル (`.class`) を持つアーカイブ。アーカイブは圧縮されていることもあります。

`@`*file* 空白で区切られた (複数の) 入力ファイル名を持つファイル (現時点ではこれらのファイルはすべて`.java`ソースファイルでなくてはなりませんが、将来この点については変更される可能性があります)。名前が指定されたファイルはすべて、コマンドライン上で指定されたかのようにコンパイルされます。

*library*`.a`
*library*`.so`
`-l`*libname* リンク時に使われるライブラリ。gccマニュアルを参照して下さい。

gcjコマンドライン上には複数の入力ファイルを指定することができます。この場合、指定されたすべてのファイルがコンパイルされます。`-o` *FILENAME* オプションを指定すると、すべての入力ファイルがコンパイルされて、*FILENAME* という名前の出力ファイルが 1 つ生成されます。このオプションは、`-S`オプションや`-c`オプションが使われているときにも使うことができます。しかし、`-C`オプションや`--resource`オプションが使われているときには使うことはできません (これは、通常のgccでは許されない拡張機能です)。(複数の入力ファイルが指定された場合、現在のバージョンでは、それらはすべて`.java`ファイルでなければなりません。この点については将来修正したいと考えています。)

## 1.2 入力オプション

gcjには、必要なファイルを探す場所を制御するオプションがあります。例えば、gcjは、コンパイルするよう求められたファイルが参照しているクラスをロードする必要があるかもしれません。他の Java 言語用コンパイラと同様、gcjにもクラスパスという概念があります。いくつかのオプションおよび環境変数によってこのクラスパスを操作することができます。gcjがあるクラスを探すとき、該当する‘`.class`’ファイルまたは‘`.java`’ファイルを見つけるためにクラスパスを探索します。gcjには組み込みのクラスパスがあり、それはインストールされた‘`libgcj.jar`’を指します。このファイルにはすべての標準的なクラス群が含まれています。

以下において、ディレクトリまたはパス要素は、ファイルシステム上にある実際のディレクトリを指すことも、‘`.zip`’ファイルまたは‘`.jar`’ファイルを指すこともできます。gcjは、‘`.zip`’ファイルや‘`.jar`’ファイルを、それがあたかもディレクトリであるかのように探索します。

`-I`*dir* `-I`によって指定されたすべてのディレクトリは、そのままの順序で他のオプションから組み立てられたクラスパスの先頭に追加されます。`javac`のようなツールとの互換性が重要ではない限り、クラスパスを操作するときには、他のオプションではなく常に`-I`オプションを使うことをお勧めします。

`--classpath=`*path*
クラスパスを *path* にします。*path* は、コロンで区切られたパスの一覧です (Windows ベースのシステムでは、セミコロンで区切られたパスの一覧です)。これは、組み込みサーチパス (「ブートクラスパス」) を無効にすることはありません。

`--CLASSPATH=`*path*
`--classpath`と同義ですが、推奨されません。

`--bootclasspath=`*path*
`java.lang.String`のような標準組み込みクラスを見つける場所を指定します。

`--extdirs=`*path*
*path* の中の個々のディレクトリについて、そのディレクトリ内に存在するファイルをクラスパスの終端に追加します。

`CLASSPATH`
パスの一覧を保持する環境変数です。

最終的なクラスパスは以下のように組み立てられます。

- 最初に、`-I`に指定されたすべてのディレクトリが使われます。
- '`--classpath`'が指定されると、その値が追加されます。'`--classpath`'が指定されていなくて、`CLASSPATH`環境変数が指定されている場合は、その値が追加されます。どちらも指定されていない場合はカレントディレクトリ (`"."`) が追加されます。
- `--bootclasspath`が指定されると、その値が追加されます。`--bootclasspath`が指定されていないと、組み込みのシステムディレクトリに相当する '`libgcj.jar`'が追加されます。
- 最後に、`--extdirs`が指定されると、指定されたディレクトリ内に存在するファイルがクラスパスの終端に追加されます。`--extdirs`が指定されていないと、組み込みの拡張ディレクトリ (extdir) である`$(prefix)/share/java/ext`内に存在するファイルが追加されます。

`gcj`によって作成された (`libgcj.jar`に格納されている) `java.lang.Object`クラスのクラスファイルには、長さゼロの特殊な`gnu.gcj.gcj-compiled`という属性情報が含まれています。コンパイラは`java.lang.Object`をロードするときにこの属性情報を探して、見つけることができなければエラーを報告します。ただし、コンパイルの出力形式がバイトコードであるときはエラーを報告しません (`-fforce-classes-archive-check`オプションを使って、この特定のケースにおける振る舞いを無効にすることができます)。

`-fforce-classes-archive-check`
コンパイラに対して、常に `java.lang.Object`の長さゼロの特殊な `gnu.gcj.gcj-compiled`属性情報をチェックさせて、見つからない場合にはエラーを出力することを強制します。

## 1.3 エンコーディング

Java プログラミング言語では一貫して Unicode が使われます。`gcj`は、他のロケールも適切に統合することに努めていて、ほとんどすべてのエンコーディングを使って '`.java`' ファイルを書くこと

ができます。gcjは、これらのエンコーディングをコンパイル時に内部エンコーディングに変換する方法を知っています。

--encoding=*NAME*オプションを使って、ソースファイルにおいて使われている (特定の文字セットの) エンコーディングを指定することができます。このオプションが指定されていないときは、その時点におけるカレントロケールがデフォルトのエンコーディングとなります。ホストシステムに十分なロケールサポートがない場合、gcjは、デフォルトエンコーディングが Unicode の 'UTF-8'エンコーディングであるものと想定します。

--encodingを実装するために、gcjは単に、ホストプラットフォームの iconv変換ルーチンを使っているだけです。このことは、gcjができることは、そのホストプラットフォームにできる範囲に事実上限定されていることを意味しています。

--encodingの引数に指定することのできる名前はプラットフォームによって異なります (標準が存在しないからです)。しかし、gcjは 'UTF-8'という名前のエンコーディングを内部的に実装していますので、ソースファイルにおいてこのエンコーディングを使うことにすれば、すべてのホスト上で動作することが保証されます。

## 1.4 警告

gcjはいくつかの警告を実装しています。他の一般的な gccの警告と同様、ある警告が-Wfooという形式のオプションによって有効になるとき、それを無効にするには-Wno-fooという形式を指定します。以下においては、効力を持つ警告の形式を記述することにしました。すなわち、以下の一覧に示されているものの反対がデフォルトであるということです。

-Wredundant-modifiers
: このフラグが指定されると、gcjは冗長な修飾子について警告を出力します。例えば、あるインタフェースメソッドが publicとして宣言されていると警告を出力します。

-Wextraneous-semicolon
: 空の文に対して gcjに警告を出力させます。空の文は推奨されなくなっています。

-Wno-out-of-date
: このオプションを指定すると、ソースファイルが対応するクラスファイルよりも新しいときに gcjは警告を出力しなくなります。デフォルトでは、gcjはこのことを警告します。

-Wunused
: gccの-Wunusedと同じです。

-Wall
: -Wredundant-modifiers -Wextraneous-semicolon-Wunusedと同じです。

## 1.5 コード生成

コード生成を制御する gccの数多くのオプションに加えて、gcjは固有のオプションをいくつか持っています。

--main=*CLASSNAME*
: このオプションは、リンクの際に、生成される実行可能ファイルが実行されるときに呼び出されるべき mainメソッドを持つクラスの名前を指定するのに使われます。[1]

[1] リンカは、デフォルトでは mainという名前のグローバル関数を探します。Java にはグローバル関数というものは存在しない上に、複数の Java クラスが mainメソッドを持っている可能性があるため、アプリケーションを開始するときに呼び出すべき mainメソッドがどれであるかをリンカに教える必要があります。

`-D`*name*`[=`*value*`]`
: このオプションは`--main`と組み合わせてのみ使うことができます。これは、*value* という値を持つ *name* という名前のシステムプロパティを定義します。*value* が指定されていない場合、値はデフォルトで空文字列となります。これらのシステムプロパティはプログラムの開始時に初期化され、実行時には `java.lang.System.getProperty`メソッドによって値を取得することができます。

`-C`
: このオプションは、オブジェクトコードではなくバイトコード (‘`.class`’ファイル) を生成するよう `gcj`に知らせるために使われます。

`--resource` *resource-name*
: このオプションは、実行時にコアプロトコルハンドラを使って‘`core:/`*resource-name*’としてアクセスすることができるようにするために、指定されたファイルの内容をオブジェクトコードに変換するよう `gcj`に知らせるために使われます。*resource-name* は、実行時に見つけるときのリソース名であることに注意してください。例えば、その名前は `ResourceBundle.getBundle`に対する呼び出しにおいて使われることがあります。このようにコンパイルされる実際のファイルの名前は別に指定されなければなりません。

`-d` *directory*
: `-C`と組み合わせて使われると、生成されたすべての‘`.class`’ファイルは *directory* の下の適切なサブディレクトリに置かれます。デフォルトでは、その時点における作業ディレクトリのサブディレクトリに置かれます。

`-fno-bounds-check`
: デフォルトでは、`gcj`は、配列のインデックスを指定するすべてのオペレーションにおいて境界チェックを行なうコードを生成します。このオプションを指定すると、これらのチェックは省略され、配列を頻繁に使用するコードのパフォーマンスを向上させることができます。この結果、問題となるコードが実際に配列境界の制約に違反した場合に、予期できない振る舞いを示す可能性があるということに注意してください。コードが絶対に `ArrayIndexOutOfBoundsException`を投げることがないと確信できる場合にこのオプションを使うのが安全です。

`-fno-store-check`
: 配列への値の格納に際してチェックを行なうコードを生成しません。オブジェクトを配列に格納するとき、そのオブジェクトが配列の要素型と代入互換性があること (このことはコンパイル時には分からない可能性があります) を確認するために、実行時にチェックを行なうコードが通常は生成されます。このオプションを指定すると、このチェックが省略されます。これにより、配列に頻繁にオブジェクトを格納するコードのパフォーマンスを向上させることができます。コードが絶対に `ArrayStoreException`を投げることがないと確信できる場合にこのオプションを使うのが安全です。

`-fjni`
: `gcj`には、ネィティブメソッドを書く方法が 2 つあります。CNI と JNI です。デフォルトでは、`gcj`は CNI が使われているものと想定します。ネィティブメソッドを持つクラスをコンパイルする場合、それらのメソッドが JNI によって実装されているのであれば、`-fjni`を指定しなければなりません。このオプションを指定すると、`gcj`は、JNI メソッドを呼び出すスタブコードを生成します。

`-fno-optimize-static-class-initialization`
: 最適化レベルが`-O2`以上のとき、`gcj`は、静的クラスをそれが最初に使われたときに初期化するようランタイムを呼び出す方法を最適化しようと試みます (この最適化は`-C`が指定されているときには実行されません)。ネィティブコードにコンパイルするときは、使われて

いる最適化レベルにかかわらず、-fno-optimize-static-class-initialization によってこの最適化は無効化されます。

## 1.6 コンフィグレーション時のオプション

gcjのコード生成オプションのいくつかは、最終的に使われる ABI に影響を与えます。したがって、これらのオプションを指定して意味があるのは、ランタイムパッケージである libgcjのコンフィグレーションを行なうときだけです。libgcjは、これらのグループの中から適切なオプションを 'spec' ファイルに書き込みます。このファイルは gcjによって読み込まれます。これらのオプションを以下に一覧にして示しているのは完全を期すためです。libgcjを使っているのであれば、これらのオプションを操作することはしないほうが良いでしょう。

-fuse-boehm-gc
: Boehm GC のビットマップマーキングコードの使用を有効化します。具体的には、gcj が個々の vtable にオブジェクトマーキング用のディスクリプタを出力するようになります。

-fhash-synchronization
: デフォルトでは、シンクロナイゼーションデータ (synchronize、wait、notifyにおいて使われるデータ) は個々のオブジェクトの中にある 1 ワードのデータとして参照されます。このオプションを指定すると、gcjは、この情報がオブジェクト自身の中にではなく、あるハッシュテーブルの中に格納されているものと想定します。

-fuse-divide-subroutine
: いくつかのシステムにおいて、整数型の除算を実行するのにライブラリルーチンが呼び出されます。これは、ゼロによる除算を行なった際の例外を正しく処理するために必要とされます。

-fcheck-references
: いくつかのシステムにおいて、リファレンス経由でオブジェクトにアクセスするところすべてにインラインチェックを行なうコードを挿入する必要があります。他のシステムにおいては、このようなことは必要ありません。ヌルポインタによるアクセスはプロセッサによって自動的に捕捉されるからです。

# 2 Javaプラットフォームとの互換性

私たちはJavaプラットフォームが分裂しないことが重要であると考えていますので、gcjとlibgcjが、関連するJava仕様に準拠するよう努力しています。しかし、人的資源の不足と不完全かつ不明瞭なドキュメントがこの努力に水を差しています。そのため、gcjを使うにあたっては注意すべき点があります。

この互換性に関する問題の一覧は決して完全なものではありません。

- gcjはJDK 1.1のJava言語を実装しています。内部クラスもサポートしていますが、まだバグの多いことが分かっています。Java 2 strictfpキーワードはまだサポートしていません(このキーワードを認識はしますが無視します)。
- libgcjは、JDK 1.2ライブラリとおおむね互換性があります。しかし、libgcjには欠けているパッケージも数多くあります。これは、java.awtにおいて特に顕著です。個々のクラスやメソッドで欠けているものもあります。現在のところ、libgcjとJava 2プラットフォームの相違点を一覧にしたものはありません。
- libgcjによるメソッドまたはクラスの実装がJDKの実装と異なることもあります。これは必ずしもバグであるとは限りません。しかし、その相違による影響があるのであれば、私たちが適切な対応を議論することができるようにするためにも、報告をすることにおそらく意味があるでしょう。

# 3 gcjh の実行

gcjhプログラムは、クラスファイルからヘッダファイルを生成するのに使われます。CNI ヘッダファイル、JNI ヘッダファイルの両方を生成することができます。また、必要とされるネィティブメソッドを実装するための基盤として使うことのできるスタブ実装ファイルも生成します。

`-stubs` gcjhに、ヘッダファイルではなくスタブファイルを生成させます。デフォルトでは、スタブファイルの名前はクラスの名前に拡張子 '`.cc`'を付けたものになります。JNI モードでは、出力ファイルにはデフォルトで '`.c`'という拡張子が付きます。

`-jni` gcjhに対して、JNI ヘッダまたは JNI スタブを生成するよう知らせます。デフォルトでは CNI ヘッダが生成されます。

`-add` *text* *text* をクラス本体に挿入します。JNI モードでは無視されます。

`-append` *text*
*text* をヘッダファイルのクラス宣言の後ろに挿入します。JNI モードでは無視されます。

`-friend` *text*
*text* を `friend`宣言としてクラスの中に挿入します。JNI モードでは無視されます。

`-prepend` *text*
*text* をヘッダファイルのクラス宣言の前に挿入します。JNI モードでは無視されます。

`--classpath=`*path*
`--CLASSPATH=`*path*
`-I`*directory*
`-d` *directory*
`-o` *file* これらのオプションはすべて `gcj`の対応するオプションと同じものです。

`-o` *file* 出力ファイル名を設定します。コマンドライン上に複数のクラスが指定されている場合、このオプションを使うことはできません。

`-td` *directory*
一時ファイルの作成場所として使うディレクトリの名前を設定します。

`-M` すべての依存情報を標準出力に出力します。通常の出力は抑止されます。

`-MM` システム以外の依存性情報を標準出力に出力します。通常の出力は抑止されます。

`-MD` すべての依存性情報を標準出力に出力します。

`-MMD` システム以外の依存性情報を標準出力に出力します。

`--help` gcjhに関するヘルプ情報を出力して終了します。それ以上の処理は行なわれません。

`--version`
gcjhのバージョン情報を出力して終了します。それ以上の処理は行なわれません。

`-v, --verbose`
実行中に追加情報を出力します。

これ以外のオプションはすべてクラスの名前とみなされます。

# 4 jv-scan の実行

jv-scanプログラムは Java ソースファイル (‘.java’ファイル) に関する情報を出力するのに使うことができます。

`--complexity`
: 個々の入力ファイルについて、繰り返し処理の複雑性 (cyclomatic complexity) の観点から、複雑性の度合いを出力します。

`--encoding=`*name*
: 対応する gcjオプションと同じように働きます。

`--print-main`
: このファイルの中にある mainメソッドを持つクラスの名前を出力します。

`--list-class`
: 入力ファイルの中に定義されているすべてのクラスの名前を一覧にして出力します。

`--list-filename`
: --list-classが指定されていると、このオプションは、個々のクラスが見つかったファイルの名前も jv-scanに出力させます。

`-o` *file*
: 指定された名前のファイルを出力先とします。

`--help`
: ヘルプ情報を出力して終了します。

`--version`
: バージョン番号を出力して終了します。

# 5 jcf-dump の実行

クラスファイルを検査するプログラムです。javapに似ています。クラス名またはファイル名によって指定されたいくつかのクラスに関する情報を出力します。

`-c` メソッド本体を逆アセンブルします。デフォルトでは、メソッド本体は出力されません。

`--javap` javapフォーマットで出力を生成します。この機能の実装は非常に不完全です。

`--classpath=`*path*
`--CLASSPATH=`*path*
`-I`*directory*
`-o` *file* これらのオプションは、対応する gcjオプションと同じです。

`--help` ヘルプ情報を出力して終了します。

`--version`
バージョン番号を出力して終了します。

`-v, --verbose`
実行中に追加情報を出力します。

# 6 gij の実行

gijは libgcjに付属している Java バイトコード インタープリタです。gijはすべてのプラットフォームにおいて利用できるわけではありません。移植するためにはアセンブリ言語によるプログラミングが少々必要ですが、gcjがサポートするターゲットプラットフォームでも、まだこの作業が完了していないものがあります。

gijに対する主たる引数はクラスの名前です。また、-jarオプションが指定されている場合は jar ファイルの名前です。この引数の前に指定されたオプションは gijによって解釈されます。この引数の後ろに指定されたオプションはインタープリタが実行するプログラムに渡されます。

クラス名が指定されていて、そのクラスが適切なシグニチャ(唯一の引数の型が String[]、戻り値の型が static void) を持つ mainメソッドを持っていない場合、gijはエラー情報を出力して終了します。

jar ファイルが指定されると、gijは、そのファイルの中の情報を使って、どのクラスの mainメソッドを呼び出すかを決定します。

gijは、コマンドラインにおいてファイル名の後ろに指定されたすべてのオプションを指定して、mainメソッドを呼び出します。

gijはインタープリタによって解釈されるコードにしか使えないわけではないことに注意してください。libgcjには、共用オブジェクトを動的にロードすることのできるクラスローダが含まれていますので、コンパイルされてクラスパス上にある共用ライブラリの中に置かれたクラスの名前を gijに渡すことも可能です。

-D*name*[=*value*]
: 名前 *name* と値 *value* を持つシステムプロパティを定義します。*value* が指定されていないときは、デフォルトで空文字列が値となります。これらのシステムプロパティはプログラムの実行開始時に初期化され、実行時には java.lang.System.getProperty メソッドを使って値を取得することができます。

-ms=*number*
: ヒープサイズの初期値を設定します。

-mx=*number*
: ヒープサイズの最大値を設定します。

-jar
: gijに渡された名前を、クラスの名前としてではなく jar ファイルの名前として解釈すべきであることを示します。

--help
: ヘルプ情報を出力して終了します。

--version
: バージョン番号を出力して終了します。

# 7 jv-convert の実行

`jv-convert` [‘`OPTION`’] ... [*INPUTFILE* [*OUTPUTFILE*]]

`jv-convert`は `libgcj`に付属するユーティリティで、ファイルのエンコーディングを変更します。Unix の `iconv`ユーティリティに似ています。

`jv-convert`がサポートするエンコーディングはプラットフォーム依存です。現在のところ、サポートされるすべてのエンコーディングの一覧を取得する方法はありません。

`--encoding` *name*
`--from` *name*
: *name* で指定されるエンコーディングを入力エンコーディングとして使います。デフォルトはカレントロケールのエンコーディングです。

`--to` *name*
: *name* で指定されるエンコーディングを出力エンコーディングとして使います。デフォルトは `JavaSrc`エンコーディングです。このエンコーディングでは、非 ASCII 文字は、‘\u’でエスケープした ASCII コードになります。

`-i` *file*
: *file* から読み込みます。デフォルトでは標準入力から読み込みます。

`-o` *file*
: *file* に書き込みます。デフォルトでは標準出力に書き込みます。

`--reverse`
: 入力エンコーディングと出力エンコーディングを入れ替えます。

`--help`
: ヘルプメッセージを出力して終了します。

`--version`
: バージョン情報を出力して終了します。

# 8 rmic の実行

rmic [‘OPTION’] ... *class* ...

rmicは libgcjに付属しているユーティリティで、リモートオブジェクト用のスタブを生成します。

このプログラムはまだ JDK の rmicと完全な互換性はないことに注意してください。例えば‘-classpath’のようないくつかのオプションは、認識はされますが現在のところ無視されます。現在のところ、これらのオプションについてはドキュメントに記述していません。

オプションは、‘--’で始まる GNU スタイルで指定することもできます。例えば、‘--help’も受け付けられます。

-keep
-keepgenerated
: デフォルトでは、rmicは中間ファイルを削除します。これらのオプションのいずれかを指定すると、中間ファイルは削除されなくなります。

-v1.1
: プロトコルバージョン 1.1 に対応したスタブとスケルトンを rmicに作成させます。

-vcompat
: プロトコルバージョン 1.1、1.2 の両方と互換性のあるスタブとスケルトンを rmicに作成させます。これがデフォルトです。

-v1.2
: プロトコルバージョン 1.2 に対応したスタブとスケルトンを rmicに作成させます。

-nocompile
: 生成されたファイルをコンパイルしません。

-verbose
: rmicが実行中の処理に関する情報を出力します。

-d *directory*
: 出力ファイルを *directory* に置きます。デフォルトでは、出力ファイルはカレント作業ディレクトリに置かれます。

-help
: ヘルプメッセージを出力して終了します。

-version
: バージョン情報を出力して終了します。

# 9 rmiregistry の実行

`rmiregistry` [‘`OPTION`’] ... [*port*]

`rmiregistry`は、カレントホスト上においてリモートオブジェクトレジストリを実行開始します。ポート番号が指定されていない場合、ポート番号 1099 が使われます。

`--help` ヘルプメッセージを出力して終了します。

`--version`
バージョン番号を出力して終了します。

# 10 CNI

ここではCNIについて記載します。CNIは、Cygnus Native Interfaceの略であり、C++を使ってJavaネィティブメソッドを書くための便利な方法です。標準的なJNI (Javaネィティブインタフェース) に対する、より効率的かつより便利な、しかし移植性は弱い代替策です。

## 10.1 基本的概念

言語の機能という観点からすると、Javaはほぼ C++のサブセットになっています。Javaにはいくつか重要な拡張機能があり、強力な標準クラスライブラリも付いていますが、全体としては、これらも基本的な類似性を変えるものではありません。Javaはハイブリッドなオブジェクト指向言語であり、クラス型に加えていくつかのネィティブ型を持っています。Javaにおいては、クラスがベースになっていて、クラスには、オブジェクトのインスタンスごとに存在するフィールドに加えて静的フィールドがあります。また、インスタンスメソッドに加えて静的メソッドがあります。静的でないメソッドは仮想メソッドにすることができ、オーバロードすることができます。オーバローディングは、実際の引数の型をパラメータの型に対応させることによって実行時に解決されます。仮想メソッドは、ディスパッチテーブル (仮想関数テーブル) を経由する間接呼び出しを使って実装されます。オブジェクトはヒープ上に割り当てられ、コンストラクタメソッドを使って初期化されます。クラスはパッケージ階層に組織化されます。

これまで列挙したような属性はすべて C++においても妥当します。ただし、C++にはさらに追加的な特徴があります (例えば、C++オブジェクトは、ヒープ上に割り当てることもできますが、静的に、あるいは、ローカルスタックフレーム上に割り当てることもできます)。`gcj`は G++と同一のコンパイラ技術 (GNU C++コンパイラ) を使っていますので、両方の言語の共通部分において同一の ABI (オブジェクト表現、および、呼び出し規約) を使うようにすることが可能です。CNIにおける主たるアイデアは、Javaオブジェクトは C++オブジェクトであり、すべての Javaクラスは C++クラスである (しかしその逆ではない) というところにあります。そこで、Javaと C++を統合するにあたって最重要なタスクは、余計な非互換性を除去することにあります。

CNIコードは通常の C++ソースファイルとして書きます。(Java/CNIを認識する C++コンパイラ、具体的には最近のバージョンの G++を使わなければなりません。)

CNI C++ソースファイルには以下の記述がなければなりません。

```
#include <gcj/cni.h>
```

さらに、個々の Javaクラスごとに 1 つのヘッダファイルをインクルードしなければなりません。以下に例を示します。

```
#include <java/lang/Character.h>
#include <java/util/Date.h>
#include <java/lang/IndexOutOfBoundsException.h>
```

これらのヘッダファイルは`gcjh`によって自動的に生成されます。

CNIは、C++から Javaオブジェクトや Javaプリミティブ型を使うのを簡単にするための関数やマクロをいくつか提供しています。これらの CNI関数やマクロの名前は一般的には`Jv`という接頭語で始まります。例えば、`JvNewObjectArray`です。他のライブラリとの衝突を回避するためにこの慣習が使われています。CNIの内部関数の名前は`_Jv_`という接頭語で始まります。これらの関数を呼び出すべきではありません。もしそうする必要があるのであれば私たちに連絡をください。代わりとなる解決策を見つけるよう努力します。(このマニュアルでは、`_Jv_AllocBytes`を 1 つの例として取り上げています。CNIは、本来は`JvAllocBytes`という関数を提供すべきであると思います。)

### 10.1.1 制限

Java クラスが C++クラスであるということは、Java の拘束から解放されるということを意味しません。CNI C++クラスは、Java 言語の規則に従わなければなりません。

例えば、CNI クラスの中に C の文字列 (`char*`) を引数として取るメソッドを宣言することはできません。また、Java のデータ型以外のデータ型のメンバ変数を宣言することもできません。

## 10.2 パッケージ

Java における唯一のグローバル名はクラス名とパッケージ名です。*package* (パッケージ) は、ゼロ個以上のクラスを持つことができ、ゼロ個以上のサブパッケージを持つこともできます。すべてのクラスは無名パッケージ、あるいは、階層的かつグローバルに一意な名前を持つあるパッケージのいずれかに属します。

Java のパッケージは C++の *namespace* (ネームスペース) にマップされます。`java.lang.String` という Java クラスは、`java`パッケージのサブパッケージである `java.lang`パッケージに属しています。このクラスの C++版は `java::lang::String`です。これは、`java`ネームスペースの中の `java::lang`ネームスペースに属しています。

以下にこのことを表現する方法を示します。

```
// クラスを宣言する。これはおそらくヘッダファイルの中で行なわれるであろう。
namespace java {
  namespace lang {
    class Object;
    class String;
    ...
  }
}

class java::lang::String : public java::lang::Object
{
  ...
};
```

`gcjh`ツールは、必要となるネームスペース宣言を自動的に生成します。

### 10.2.1 パッケージ名の省略

常に完全修飾された Java クラス名を使うのはうんざりするほど冗長になることがあります。また完全修飾名を使うと、コードが単一のパッケージに結び付けられてしまい、クラスのパッケージが変更されるとコードの変更が必要になってしまいます。Java の `package`宣言は、それ以下のクラス宣言が指定されたパッケージに属することを指定するものです。したがって、完全なパッケージ名を明示的に指定する必要はなくなります。`package`宣言に続けてゼロ個以上の `import`宣言を指定することができます。これによって、あるパッケージに属する 1 個のクラス、あるいは、すべてのクラスを、単にその識別名だけで指定することができるようになります。C++は、これに似たものを `using`宣言と `using`指示子によって提供しています。

Java においては、

```
import package-name.class-name;
```

によって、プログラムテキストは、完全修飾された名前 *package-name.class-name*の省略表現として *class-name* という形でクラスを参照できるようになります。

C++において同じことを達成するには、以下のようにしなければなりません。

```
using package-name::class-name;
```

Javaでは必要に応じてインポートを行なわせることもできます。以下のようにします。

```
import package-name.*;
```

こうすることによって、プログラムテキストの中で*package-name*パッケージの任意のクラスをそのクラス名だけで参照することができるようになります。

C++において同じことを達成するには、以下のようにしなければなりません。

```
using namespace package-name;
```

## 10.3 プリミティブ型

Javaは、整数、浮動小数点数、文字、論理値を表わす8種類のプリミティブ型(および、void型)を提供しています。C++も非常に良く似た具体的な型を独自に持っています。しかし、C++におけるこれらの型の実装は必ずしも常に同一であるとは限りません(例えばintは、16ビット、32ビット、あるいは、64ビットとして実装される可能性があります)。したがって、CNIは、Javaの各プリミティブ型に対応した特殊なC++型を提供しています。

| **Java**の型 | **C/C++**の型 | 説明 |
|---|---|---|
| `char` | `jchar` | 16ビットUnicode文字 |
| `boolean` | `jboolean` | 論理値(true/false) |
| `byte` | `jbyte` | 8ビット符号付整数 |
| `short` | `jshort` | 16ビット符号付整数 |
| `int` | `jint` | 32ビット符号付整数 |
| `long` | `jlong` | 64ビット符号付整数 |
| `float` | `jfloat` | 32ビットIEEE浮動小数点数 |
| `double` | `jdouble` | 64ビットIEEE浮動小数点数 |
| `void` | `void` | 値なし |

Javaの型に言及する際には、期待外れな誤解を避けるためにも、これらのC++型名(例えば`jint`)を使うべきです。

### 10.3.1 プリミティブ型に対応するリファレンス型

Javaの各プリミティブ型には対応するリファレンス型があります。例えば、`boolean`に対応するのは`java.lang.Boolean`クラスです。こうしたクラスの扱いを簡単にするために、GCJは`JvPrimClass`マクロを提供しています。

マクロ JvPrimClass(型)

*Return a pointer to the* `Class` *object corresponding to the type supplied.*
指定された型に対応する`Class`オブジェクトへのポインタを返します。

```
JvPrimClass(void) ⇒ java.lang.Void.TYPE
```

## 10.4 インタフェース

Java クラスは、単一の基底クラスからの継承に加えて、ゼロ個以上のインタフェースを実装することができます。

CNI では、CNI のコードにインタフェースのメソッドを実装させることができます。また、いくつか制限はありますが、インタフェースに対するリファレンスを通じてメソッドを呼び出すこともできます。

CNI は、インタフェースの継承をまったく認識しません。したがって、あるインタフェースメソッドを呼び出すことができるのは、呼び出しの対象となるフィールドの宣言上の型が、そのメソッドを宣言しているインタフェースと一致している場合だけです。そのインタフェースリファレンスを正しい上位インタフェースにキャストすることによって、この制限を回避することができます。

例えば、

```
interface A
{
  void a();
}

interface B extends A
{
  void b();
}
```

において、C++のコードで型が Bの変数を宣言している場合、その変数をまず Aにキャストしない限り a()を呼び出すことはできません。

## 10.5 オブジェクトとクラス

### 10.5.1 クラス

すべての Java クラスは java.lang.Objectを継承しています。C++には唯一のルートクラスというものは存在しませんが、私たちは、Java の java.lang.Objectクラスに対応する C++クラス java::lang::Objectを使っています。その他のすべての Java クラスは、java::lang::Objectを継承する、対応する C++クラスにマップされます。

インタフェース継承 (implementsキーワード) は、現在のところ C++へのマッピングには反映されません。

### 10.5.2 オブジェクトフィールド

個々のオブジェクトにはオブジェクトヘッダがあり、その後ろに、クラスのインスタンスフィールドがあります。オブジェクトヘッダは、ディスパッチテーブル、あるいは、仮想関数テーブルへの単一のポインタです。(オブジェクトの前に追加的なフィールドが存在する可能性はあります。例えば、メモリ管理用のフィールドです。しかし、これはアプリケーションからは不可視であり、オブジェクトに対するリファレンスはディスパッチテーブルへのポインタを指しています。)

フィールドは、C++の場合と同一の順序、境界整列、サイズによって展開されます。具体的には、8 ビットおよび 16 ビットのネィティブ型 (byte、short、char、boolean) は 32 ビットに拡張されないということです。Java VM は、8 ビットおよび 16 ビットの型が VM のスタック上や一時レジスタに置かれるときには、それを 32 ビットに拡張します。

gcjhが生成したクラスのヘッダファイルをインクルードすれば、Java クラスのフィールドに自然な方法でアクセスすることができます。例えば、以下のような Java クラスがあるとしましょう。

```
public class Int
{
  public int i;
  public Integer (int i) { this.i = i; }
  public static zero = new Integer(0);
}
```

この場合、以下のように書くことができます。

```
#include <gcj/cni.h>;
#include <Int>;

Int*
mult (Int *p, jint k)
{
  if (k == 0)
    return Int::zero;  // 静的メンバに対するアクセス
  return new Int(p->i * k);
}
```

### 10.5.3 アクセス指定子

CNI は、Java のアクセス指定子を厳密に守ってはいません。Java におけるアクセス許可は C++ におけるアクセス許可に直接マップできないからです。Java の private フィールドおよび private メソッドは、C++の private フィールドおよび private メソッドにマップされます。しかし、その他のフィールドおよびメソッドは、public フィールドおよび public メソッドにマップされます。

## 10.6 クラスの初期化

Java においては、個々のクラスが最初に実際に使われるときに自動的に初期化されることが必要です。クラスの初期化には、静的フィールドの初期化、クラスの初期化メソッドの実行、基底クラスの初期化などが伴います。ほかにも、例えばコード中の文字列リテラルに対応する Stringオブジェクトの割り当てなど、実装固有のアクションが発生する可能性もあります。

GCJ コンパイラは、必要なときに確実にクラスが初期化されるようにするために、適切なところに JvInitClassの呼び出しを挿入します。C++コンパイラは、これらの呼び出しを自動的に挿入してはくれません。すなわち、クラスが確実に初期化されるようにするのはプログラマの責任なのです。しかし、Java システムが想定している慣習により、非常に簡単に実現できます。

まず libgcjが、オブジェクトのインスタンスが作成される前に確実にクラスが初期化されるようにします。これは newオペレーションの責任の一部です。Java コードおよび C++コードの両方で対処されます。(G++コンパイラが Java クラスの newを見つけると、そのオブジェクトを割り当てるために libgcjの中のルーチンを呼び出します。そのルーチンが、クラスの初期化に対処します。) こうして、クラスとそのすべての基底クラスが初期化済みであることが分かっているので、インスタンスフィールドにアクセスしたり、インスタンス (非静的) メソッドを呼び出したりしても安全なのです。

静的メソッドの呼び出しも安全です。Java コンパイラが、そのクラスが初期化済みであることを確認するコードを静的メソッドの先頭に追加しているからです。しかし、C++コンパイラはこの追加コードを挿入してはくれません。したがって、CNI を使って静的なネィティブメソッドを書くので

あれば、(そうしなくても安全であるという確信があれば別ですが) そのメソッドの中でまず最初にJvInitClassを呼び出す責任は、あなたにあります。

静的フィールドへのアクセスも、そのフィールドのクラスが初期化済みであることを必要とします。Java コンパイラのコードは、フィールドの値を取得したり設定したりする前に Jv_InitClass を呼び出すコードを生成します。しかし、C++コンパイラはこの追加コードを挿入してはくれません。したがって、C++のコードから静的フィールドにアクセスする前にそのクラスが確実に初期化されているようにする責任は、あなたにあります。

## 10.7 オブジェクト割り当て

新規に作成された Java オブジェクトはクラスインスタンス作成式を使って割り当てられます。例えば、以下のような式です。

```
new Type ( ... )
```

同一の構文が C++においても使えます。主な相違点は、C++オブジェクトは明示的に削除されなければならないというところにあります。Java では、オブジェクトはガーベジコレクタによって自動的に削除されます。CNI を使えば、新規の Java オブジェクトを標準的な C++の構文を使って割り当てることができます。この場合、C++コンパイラはガーベジコレクタからメモリを割り当てます。オーバロードされたコンストラクタがあれば、コンパイラは、標準的な C++のオーバロード解決ルールを使って正しいコンストラクタを選択します。

以下に例を示します。

```
java::util::Hashtable *ht = new java::util::Hashtable(120);
```

`void*` **_Jv_AllocBytes** (`jsize` *size*) Function

*size* バイトをヒープから割り当てます。このメモリはガーベジコレクタによって検査されることはありませんが、そのメモリへのリファレンスが見つからなければ解放されます。

## 10.8 配列

Java は多くの点において C/C++に似ていますが、配列の取り扱いは非常に異なります。C の配列は、ポインタに対する算術演算というアイデアに基づいていますが、これは Java のセキュリティ要件とは両立しないでしょう。Java の配列は本物のオブジェクトです (配列型は java.lang.Object を継承しています)。配列値を取る変数は、配列オブジェクトへのリファレンス (ポインタ) を持つ変数です。

C++コードから Java の配列を参照するのには、JArrayテンプレートが使われています。これは以下のように定義されています。

```
class __JArray : public java::lang::Object
{
public:
  int length;
};

template<class T>
class JArray : public __JArray
{
  T data[0];
public:
```

```
  T& operator[](jint i) { return data[i]; }
};
```

JNI の typedefに対応する typedefがいくつか存在します。それぞれが、関連する型のオブジェクトを保持する配列の型です。

```
typedef __JArray *jarray;
typedef JArray<jobject> *jobjectArray;
typedef JArray<jboolean> *jbooleanArray;
typedef JArray<jbyte> *jbyteArray;
typedef JArray<jchar> *jcharArray;
typedef JArray<jshort> *jshortArray;
typedef JArray<jint> *jintArray;
typedef JArray<jlong> *jlongArray;
typedef JArray<jfloat> *jfloatArray;
typedef JArray<jdouble> *jdoubleArray;
```

**T* elements** (JArray<T> *array*) Method on template<class T>

このテンプレート関数を使って、arrayで指定される配列の要素に対するポインタを取得することができます。例えば、int[]を構成する整数型要素に対するポインタは、以下のようにして取得することができます。

```
extern jintArray foo;
jint *intp = elements (foo);
```

この関数の名前は将来変更される可能性があります。

**jobjectArray JvNewObjectArray** (jsize *length*, jclass *klass*, jobject *init*) Function

ここでの klassは配列要素の型、initは配列の各スロットにセットされる初期値です。

### 10.8.1 配列の作成

各プリミティブ型に対応して、その型の配列を新規に作成するのに使うことのできる関数が提供されています。関数の名前は以下のような形式となります。

JvNew*Type*Array

例えば、

JvNewBooleanArray

は、Java プリミティブ型である boolean の配列を作成するのに使うことができます。

以下の関数定義は、これらの関数すべてに対するテンプレートです。

**jbooleanArray JvNewBooleanArray** (jint *length*) Function

インデックス長が *length* である配列を作成します。

**jsize JvGetArrayLength** (jarray *array*) Function

*array* で指定される配列の長さを返します。

## 10.9 メソッド

Java のメソッドは直接 C++のメソッドにマップされます。gcjhによって生成されるヘッダファイルには、適切なメソッド定義が含まれています。基本的には、生成されたメソッドは、Java メソッドと同一の名前と対応する型を持っていて、自然な方法で呼び出されます。

### 10.9.1 オーバロード

Java および C++はいずれもメソッドのオーバロードを提供しています。これは、1 つのクラスが同じ名前のメソッドを複数持っていて、正しいメソッドが (コンパイル時に) 引数の型をもとに選択されるというものです。正しいメソッドを選択するためのルールは (予想どおり) Java においてよりも C++においてのほうが複雑です。しかし、C++コンパイラは、gcjhによって生成された複数のオーバロードメソッドをもとに、期待どおりのメソッドを選択するでしょう。

一般的なアセンブラやリンカは、C++オーバロードのことを知っていません。したがって、標準的な実装方法は、メソッドの引数の型をアセンブリレベルの名前の一部にエンコードすることです。このエンコードのことをマングル(mangling) と呼びます。また、エンコードされた名前のことをマングルドネーム(mangled name) と呼びます。これと同じ機構が Java におけるオーバロードの実装にも使われます。C++と Java の相互運用性のためには、Java コンパイラと C++コンパイラが同一のエンコードスキーマを使うことが重要です。

### 10.9.2 静的メソッド

CNI では、Java の静的メソッドは標準的な C++構文を使って呼び出されます。すなわち、.オペレータではなく::オペレータが使われます。

以下に例を示します。

```
jint i = java::lang::Math::round((jfloat) 2.3);
```

静的ネィティブメソッドの定義には C++のメソッド定義構文が使われます。以下に例を示します。

```
#include <java/lang/Integer>
java::lang::Integer*
java::lang::Integer::getInteger(jstring str)
{
  ...
}
```

### 10.9.3 オブジェクトコンストラクタ

コンストラクタは、newオペレータを使ったオブジェクト割り当ての一環として暗黙のうちに呼び出されます。

以下に例を示します。

```
java::lang::Integer *x = new java::lang::Integer(234);
```

Java では、コンストラクタがネィティブメソッドであることは許されません。しかし、コンストラクタがネィティブメソッドを呼び出すことは可能なので、この制限を回避するコードを書くことは可能です。

### 10.9.4 インスタンスメソッド

C++ CNI メソッドから Java のインスタンスメソッドを呼び出すのには、標準的な C++構文が使われます。以下に例を示します。

```
// 最初に Java オブジェクトを作成する
java::lang::Integer *x = new java::lang::Integer(234);
// 次にメソッドを呼び出す
jint prim_value = x->intValue();
if (x->longValue == 0)
  ...
```

Java ネィティブメソッドの定義もまた自然な形で行なわれます。

```
#include <java/lang/Integer.h>

jdouble
java::lang:Integer::doubleValue()
{
  return (jdouble) value;
}
```

### 10.9.5 インタフェースメソッド

Java においては、メソッドをインタフェースリファレンスを使って呼び出すことができます。この方法は不完全な形でサポートされています。⟨undefined⟩ [Interfaces], ページ ⟨undefined⟩ を参照してください。

## 10.10 文字列

CNI は、Java の Stringオブジェクトを取り扱うためのユーティリティ関数をいくつか提供しています。その名前やインタフェースは、JNI におけるものに似ています。

`jstring` **JvNewString** (`const char*` *chars*, `jsize` *len*) Function

C の文字列 *chars* の配列中のインデックス *len* までの文字列から構成される Java の `String` オブジェクトを返します。

`jstring` **JvNewStringLatin1** (`const char*` *bytes*, `jsize` *len*) Function

*bytes* の長さ *len* のバイト列から構成される Java の Stringオブジェクトを返します。

`jstring` **JvNewStringLatin1** (`const char*` *bytes*) Function

Stringの長さが `strlen`(*bytes*)である点を除き、前の関数と同じです。

`jstring` **JvNewStringUTF** (`const char*` *bytes*) Function

C の文字列 *bytes* の中にある UTF エンコードされた文字列から構成される Stringを返します。

`jchar*` **JvGetStringChars** (`jstring` *str*) Function

Stringオブジェクト *str* の文字列から構成される配列へのポインタを返します。

`int` **JvGetStringUTFLength** (`jstring` *str*) Function

`String`オブジェクト *str* の内容を UTF-8 エンコードするために必要とされるバイト数を返します。

`jsize` **JvGetStringUTFRegion** (`jstring` *str*, `jsize` *start*, `jsize` *len*, `char*` *buf*) Function

`String`オブジェクト *str* のある部分を UTF-8 エンコードしたものをバッファ`buf`に入れます。取り出すべき部分は先頭位置 *start* と終端位置 *len* で示されます。

*buf* はバッファであり、C の文字列ではないことに注意してください。その終端は NULL ではありません。

## 10.11 C/C++との相互運用

CNI は、Java のクラスやメソッドを表現するために設計されていますので、そのまま直ちに C/C++ の型を混在させることはできません。

重要な制限の中に、Java のクラスが、Java に無い型のインスタンス変数、静的変数を持つことができないこと、また、Java に無い型を引数または戻り値の型として取るメソッドを持つことができないということがあります。

CNI においては以下のどれも可能ではありません。

```
class ::MyClass : public java::lang::Object
{
   char* variable;  // char*は Java においては正当な型ではない
}


uint
::SomeClass::someMethod (char *arg)
{
  .
  .
  .
}   // uintも char*も Java においては正当な型ではない
```

もちろん、メソッドの有効範囲内において C/C++型を使うことには問題はありません。

```
jint
::SomeClass::otherMethod (jstring str)
{
   char *arg = ...
   .
   .
   .
}
```

しかし、この制限は問題を引き起こします。そのため、CNI には`gnu.gcj.RawData`クラスが含まれています。`RawData`クラスは、検査されないリファレンス型です。言い換えると、`RawData`型として宣言された変数には任意のデータを持たせることができ、コンパイラによるチェックはいかなるものであれ行なわれないということです。

このことは、適切なキャストを使いさえすれば、CNI クラスの中に C/C++の (クラスも含む) データ構造を持つことができるということを意味しています。

以下にいくつか例を示します。

```
class ::MyClass : public java::lang::Object
{
   gnu.gcj.RawData string;

   MyClass ();
   gnu.gcj.RawData getText ();
   void printText ();
}

::MyClass::MyClass ()
{
   char* text = ...
   string = text;
}

gnu.gcj.RawData
::MyClass::getText ()
{
   return string;
}

void
::MyClass::printText ()
{
  printf("%s\n", (char*) string);
}
```

## 10.12 例外処理

C++と Java は一般的な例外処理フレームワークを共有していますが、両者は完全に統合されているわけではありません。主要な問題点は、これら 2 つの言語が持つ、実行時の型情報に関する機能が、統合されていないことです。

それにもかかわらず、実際にはかなりうまく機能します。通常の throwを使って C++から Java の例外を投げることができます。そして、この例外を Java のコードの中で捕捉することができます。また同様に、Java から投げられた例外を C++の catchを使って捕捉することもできます。

以下に例を示します。

```
if (i >= count)
   throw new java::lang::IndexOutOfBoundsException();
```

通常、Java の例外を使う C++コードを書くと、G++はそのことを自動的に検出して、それらの例外を適切に処理します。しかし、Java の例外が C++側に投げられたときに、C++コードがする必要のあることが単にデストラクタを呼び出すことだけである場合、GCC はそのことを正しく推測してくれないでしょう。この問題を持つコードのサンプルを以下に示します。

```
struct S { ~S(); };

extern void bar();     // Javaで実装されていて、例外を投げる。

void foo()
{
  S s;
  bar();
}
```

正しい推測を行なえないことの結果として、通常はリンクが失敗します。その際、__gxx_personality_v0という名前のルーチンが存在しないというメッセージが出力されます。

ファイルの先頭に#pragma GCC java_exceptionsと書くことによって、コンパイラの推測の如何にかかわらず、翻訳単位の中でJavaの例外が使われることをコンパイラに知らせることができます。この#pragmaは、例外を投げる関数、例外を捕捉する関数、または、例外が投げられたときにデストラクタを実行する関数のどれよりも前になければなりません。

## 10.13 同期化

個々のJavaオブジェクトはどれも暗黙的なモニタを持っています。Java VMは、monitorenter命令を使ってモニタのロックを取得し、monitorexit命令を使ってモニタのロックを解放します。

これらに対応するCNIマクロはJvMonitorEnterとJvMonitorExitです(JNIにも類似のメソッドMonitorEnterとMonitorExitがあります)。

Javaソース言語は、これらのプリミティブに直接アクセスする手段を提供していません。その代わりになるものとしてsynchronized文があります。これは、ブロックに入る前に暗黙的にmonitorenterを呼び出し、ブロックから出るときにmonitorexitを呼び出します。そのブロックの実行が例外によって異常中断させられる場合でもロックは解放されなければならないということに注意してください。このことは、同期化ロックを囲む暗黙的なtry finallyが存在することを意味しています。

C++においては、ロックを解放するのにデストラクタを使うのが合理的です。CNIは、以下に示すユーティリティクラスを定義しています。

```
class JvSynchronize() {
  jobject obj;
  JvSynchronize(jobject o) { obj = o; JvMonitorEnter(o); }
  ~JvSynchronize() { JvMonitorExit(obj); }
};
```

したがって、以下のJavaコード

```
synchronized (OBJ)
{
   CODE
}
```

は、例えば以下のようなC++コードになるでしょう。

```
{
   JvSynchronize dummy (OBJ);
   CODE;
}
```

Javaにはsynchronized属性を持つメソッドもあります。これは、メソッド本体全体をsynchronized文で囲むことと同じです。(あるいは、呼び出し側で同期化を行なわせる必要がある実装というのもあ

り得るでしょう。しかし、これはコンパイラにとっては現実的ではありません。すべての仮想メソッドの呼び出しにおいて、同期化が必要かどうかを実行時にテストしなければならないからです。) gcjにおいては、synchronized属性はメソッドの実装側で処理されるので、synchronized属性を持つネィティブメソッドの場合は、(そのメソッドの C++実装において) 同期化を処理するかどうかは、そのメソッドのプログラマ次第となります。言葉を換えれば、native synchornizedメソッドにおいては JvSynchronizeの呼び出しを手作業で追加する必要があるということです。

## 10.14 呼び出し

CNI は、Java コードから C++を呼び出すことができるようにするだけでなく、C++アプリケーションが Java クラスを呼び出すこともできるようにします。呼び出し *API* として知られているいくつかの関数が、そのために提供されています。

`jint` **JvCreateJavaVM** (`void*` *vm_args*) Function

Java ランタイムを初期化します。この関数は、スレッドインタフェース、ガーベジコレクタ、例外処理、および、ランタイムのその他の主側面の基本的な初期化を実行します。これは、アプリケーションによって、他の Java 呼び出し、CNI 呼び出しのすべての先立って、非 Java の `main()` 関数から一度呼び出されなければなりません. 推奨はされませんが、`JvCreateJavaVM()`を複数回呼び出すことは、その呼び出しが単一のスレッドから行なわれるいう条件のもとでは安全です。*vm_args* パラメータは、Java ランタイムの初期化パラメータを指定するのに使うことができます。`NULL`であっても構いません。この関数は、成功したときには `0`を返します。また、ランタイムが既に初期化済みであるときは、`-1`を返します。

注： GCJ 3.1 では、`vm_args`パラメータは無視されます。将来のリリースでは使われる可能性があります。

`java::lang::Thread*` **JvAttachCurrentThread** (`jstring` *name*, `java::lang::ThreadGroup*` *group*) Function

既存スレッドを Java ランタイムに登録します。これは、その他の Java 呼び出し、CNI 呼び出しを行なう個々のスレッドによって、それらの呼び出しすべてに先立って一度呼び出されなければなりません。また、その呼び出しは `JvCreateJavaVM`の呼び出しのあとでなければなりません。*name* には、スレッドの名前を指定します。`NULL`であっても構いません。その場合は、名前は生成されることになります。*group* は、このスレッドがメンバとなる ThreadGroup です。`NULL`が指定されると、スレッドはメインスレッドグループのメンバとなります。戻り値は、このスレッドを表わす Java の `Thread`オブジェクトです。同一のスレッドから `JvAttachCurrentThread()` を複数回呼び出しても安全です。そのスレッドに既にアタッチ済みであれば、この関数の呼び出しは無視され、カレントスレッドオブジェクトが返されます。

`jint` **JvDetachCurrentThread** () Function

Java ランタイムからスレッドの登録を取り消します。これは、`JvAttachCurrentThread()` を使ってアタッチされたスレッドによって、Java コードの呼び出しが終了した後に、呼び出されるべきものです。この呼び出しによって、そのスレッドに関連付けされていたすべてのリソースが、ガーベジコレクションの適格候補となります。この関数は、成功したときには `0`を返します。カレントスレッドがアタッチされていない場合は、`-1`を返します。

### 10.14.1 捕捉されなかった例外の処理

呼び出し API を使って呼び出された Java コードから例外が投げられて、その例外のハンドラが見つからなかった場合、ランタイムはそのアプリケーションを異常停止します。アプリケーションをもっと堅牢なものにするためにも、すべての Java 例外を捕捉するトップレベルの try/catch ブロックによって、呼び出し API を使うコードを囲むことが推奨されます。

### 10.14.2 実例

以下のコードは、呼び出し API の使い方を示すものです。この例での C++アプリケーションは、Java ランタイムを初期化して自分自身をアタッチします。outフィールドにアクセスするために、java.lang.Systemクラスが初期化されて、Java の文字列が出力されています。最後に、Java の呼び出しが終了した後に、スレッドがランタイムからディタッチされています。捕捉されなかったすべての例外に対するデフォルトハンドラを提供するべく、すべてのコードが try/catch ブロックによって囲まれています。

この例は、c++ test.cc -lgcjによってコンパイルできます。

```
// test.cc
#include <gcj/cni.h>
#include <java/lang/System.h>
#include <java/io/PrintStream.h>
#include <java/lang/Throwable.h>

int main(int argc, char *argv)
{
  using namespace java::lang;

  try
  {
    JvCreateJavaVM(NULL);
    JvAttachCurrentThread(NULL, NULL);

    String *message = JvNewStringLatin1("Hello from C++");
    JvInitClass(&System.class$);
    System::out->println(message);

    JvDetachCurrentThread();
  }
  catch (Throwable *t)
  {
    System::err->println(JvNewStringLatin1("Unhandled Java exception:"));
    t->printStackTrace();
  }
}
```

## 10.15 リフレクション

CNI コードにおいてリフレクションを使うことも可能です。JNI におけるリフレクションと同様に機能します。

jfieldID型と jmethodID型は、JNI のものと同様です。

以下の関数

JvFromReflectedField,

JvFromReflectedMethod,

JvToReflectedField

JvToFromReflectedMethod

が、まもなく追加されるでしょう。JNI の関数に対応する他の関数も同様です。

# 11 リソース

gcjと libgcjを実装するにあたっては、当然のことながら、Sun Microsystems 社のドキュメントに大きく依存しました。具体的に挙げると、Java 言語仕様 (第 1 版および第 2 版)、Java クラスライブラリ (第 1 巻および第 2 巻)、Java 仮想マシン仕様です。さらに、http://java.sun.com/ にあるオンラインドキュメントも利用しました。

現時点における gcjのホームページは http://gcc.gnu.org/java/です。

gcc に関する情報については、http://gcc.gnu.org/ を参照してください。

libgcjのテストのいくつかは、Mauve テストスィートを使って行なわれました。これは、フリーソフトウェアの Java クラスライブラリテストスィートで、JCK がフリーでないために作られたものです。これについては、http://sources.redhat.com/mauve/を参照してください。