Java Assistant 1.6

On-the-fly Class Browser

www.docs.uu.se/~adavid/java/javaindex.html

Alexandre David
Uppsala University
Department of Computer Systems
adavid@docs.uu.se
www.docs.uu.se/~adavid
February 14, 2000

Contents

Chapter 1

The Project

1.1 Introduction

The aim of the project is to develop an assistant to help to program in Java. In my experience, books are not very practical reference and I prefer online reference. Another problem with books is that they do not contain your own programs (!). Sun has a very nice API generation via javadoc and it is usable but not practical and fast as it should be: typically you want a short and quick answer concerning the argument of some function, or how a function is named. Typically you would like only the public methods, sorted alphabetically to find them quickly and easily. The javadoc API does not provide this. There is an alphabetical index of all the compiled java sources but that's not practical and the methods/fields/constructors of the classes are not presented in an efficient way, which is one has to click on a name to see the signature.

The weakness of javadoc is that one has to recompile the documentation each time one changes something in the source code whereas an on-the-fly tool would not bother about this. Using a component which has no javadoc API could be possible with a such tool. It is also not possible to browse all your installed classes at once, or browse in a package directed way. Furthermore information in the API is limitted by the javadoc tag in the source code which would be nice to look at (even if no source code nor API is available!).

Another motivation is the speed of access to information, one should type a class name, maybe incomplete, press enter and get a listing of availble constructors, fields and methods filtered, sorted and clickable to get access to the source code of that particular constructor/method. This should be fast: 3 seconds to write the query and get the result at most. Sorted methods are crucial to find for example a `void setSomething(what I want)` or a `something getIt()` method type. Access to the fields should give all the constants without lot of decorations that confuse the reader in the mess of information of the javadoc API.

We can now summarize what we want from such a tool.

1.2 Requirements/Features

The tool should be:

- user friendly: fast and easy to use; should not take the whole screen and be just large enough as an assistant.
- efficient: display as litle information as needed, and display only relevant information.
- powerful: offer filtering, decompiling, disassembing and easy browsing capabilities at the source level or the API level.
- on-the-fly: access all the installed classes of a Java Virtual Machine and access all the available sources on a computer in compressed format (src.zip distributed with JDK) or not (your own development directory).

- useful: bring relevant information, answer questions of a developper when using a component, explore a component, a package in a hierarchical way.
- platform independent: run under Windows and Linux basically (Unix platforms follow) although irritating differences concerning the paths characters and other stuff.

Expected functionalities of the tools are:

- browse classes, packages and java source code (available or not).
- browse class API, class hierarchy.
- query class name, automatic completion, automatic search.
- cross references and browsing capabilities: when finding an interface implemented by a queried class, access to it should be as easy as one click; similar capabilities concerning the source code.
- searching and extracting capabilities on the source code.
- practical cut-and-paste from extracted source code, which is of the main interest when implementing an interface.

Story of the tool The tool was called Assistant and written in Java. The first version emerged with JDK 1.0 where reflection was poor. Version 2 was greatly improved and offered all the basic wanted features but showed itself limited though a number of enhancements, among them dynamic package discovery. The latest revision was 2.4 and is no longer distributed. Version 3 is a complete rewrite with a totally new architecture: performance and usability improvements, better maintainability were a concern. The project was then renamed: the distribution is called Java Assistant (current 1.6) ; the main tools are Assistant (current 3.4) and PackageViewer (current 1.0.2).

Java Assistant 1.6 meets all the desired requirements. A TODO list emerged from its usage and contains desired improvements that were forgotten or difficult to guess when implementing the tool and bugs to fix. A distribution is available for free (see the license in section 2.1

Java Packages Used Reflection is heavily used (the whole `java.lang.reflect` package), the GUI is based on AWT components and AWT lightweight components, Swing is used for the HTML java source browser, `java.io` is very used for all the stream redirection and file searching and `java.util.zip` is intensively used for jar/zip archive exploration (class and source search).

1.3 Design

The design emphasizes a modular architecture with a core which does the all the querying and searching independent from the GUI which uses the core. The GUI has components which are as much as possible independent from each other and thus reusable. The distribution contains java files which are not directly related to Assistant (and not used by it), these files are useful components.

1.3.1 Architecture

The architecture is organized into the following packages:

- `adavid.awt`: the graphical components based on `java.awt` with java lightweight components. This package contains alternative enhanced components (F*ocusable*`Label, NiceButton, NiceCheckbox, NiceLabel, BorderedPanel`, beta-research components (`MCList,MCItem`, tree components (AWT based, lightweight `TNode, TNodeRenderer, TListener, ImageTNodeRenderer, DefaultTNodeRenderer`), and static libraries-component-managers like (`ActionManager, AutoFocuser, DialogFactory, PopupManager`). The idea is to take advantage of the speed of the AWT components over Swing, that's why the tree was developped.
- `adavid.io`: this package contains string related stream facilities to redirect ouputs mainly.

- `adavid.reflect`: this package is the core of the Assistant. Key classes are `ClassReflect, JavaFinder, PackageFinder, PackageResource` to explore the classes and packages and `Decompiler` which is a general interface to use third party decompilers.
- `adavid.swing`: this package contains mainly reusable components, only `JViewer`, which is the Swing-HTML java viewer is relevant for Assistant.
- `adavid.util`: this package contains executable programs, qualified as utilities and components that are useful similarly to `java.util`. Assistant and PackageViewer are there. Sorting related classes and other useful components are there like `Bundle, ResourceLoader, JResource` in charge of automatic initialization and internationalization.

1.3.2 The Core

The core is based on reflection, directory and zip/jar exploration. One has to distinguish between purely reflection functionalities which are to access the API from the classes and the exploration to answer to the queries (completion, search, match, package browsing).

Reflection `adavid.reflect.ClassReflect` completes the `java.lang.Class` class and provides high level more powerful methods to gain more information on the classes. It provides built-in filters which command the behaviour of the methods.

`adavid.reflect.PackageFinder` is in charge of finding all the installed packages. I distinguish between possible classes and real classes for efficiency: when a package is open and searched (via `adavid.reflect.PackageResource`), classes are really checked. To validate a package, at least one class has to be found and validated. `adavid.reflect.PackageFinder` explores the directories and the zip/jar files based on the `classpath` information retrieved from the virtual machine. Partial results like the `classpath` content and the package list are serialized and saved (`$HOME/.PackageFinder.info`) to avoid recomputing them. However the `classpath` is checked to know if an update is necessary or not at each start.

Exploration `adavid.reflect.PackageResource` is in charge of finding the related information of a package, like all its classes, the class extensions of a class inside this package. A package resource is related to an entry in the `classpath` and a package definition.

`adavid.reflect.JavaFinder` is in charge of finding the java source code and performing search in the source. Decompilation is NOT here. It can return `.java` or `.jj` source files. It contains the HTML filter/generator to feed to the HTML viewer: this is related to source code manipulation.

Customization Access to third party decompilers and disassemblers is done via an interface. The linking and is dynamic and it uses reflection: someone implements this interface and declares it in a configuration file (see the customization section 1.3.4) and the Assistant uses it, that's all! The interface is `adavid.reflect.Decompiler`.

Internationalization and automatic initialization is addressed via the class `adavid.util.Bundle`. Some extensions of this class are provided by `adavid.util.ResourceLoader` and `adavid.util.JResource` for Swing related resource.

Other classes are used (`Sortable*`) for sorting and retrieving information easily on methods and constructors.

To really see how the core works, use the Assistant on it! You will see how useful on-the-fly source browsing can be.

1.3.3 The GUI

The GUI is based on reusing the components from the core and the AWT/Swing components, so it is pretty straight forward, just setup the components and listen to the events. Some more work is done in the package viewer that has to build the different trees. The GUI application is in the `util` package since they are the utilities themselves, specialized GUI in a way, not general reusable

GUI components as in the `awt` and `swing` packages. One common points is that the components listen to their own events as much as possible to avoid splitting the event handling through many objects. The main parts are:

- `adavid.util.Assistant`: the main window, where a text field allows to enter class/interface names. Upon success one sees a panel where one can choose to view constructors, fields or methods. All is gathered in `adavid.util.Assistant` since it is an independent tool based on all the other components. This controls an option dialog which tunes the filters and the java source options.
- `adavid.util.PackageViewer`: the package browser window, uses heavily the tree component and communicates with the Assistant to send queries on selected classes. It finds the class hiearchy per packages and builds the package hierarchy. The package viewer register all the packages and is invoked to initialize them if match is needed: then all the packages are queried.
- `adavid.awt.DialogFactory`: provides all the dialog windows, to ask for simple confirmation, and the simple (and fast) source viewer that has built-in search capabilities.
- `adavid.swing.JViewer`: provides the swing-HTML browser/viewer. It listens to its event and can query the core to get source extracts when clicking on links.

1.3.4 Customization

This is based on reflection. The idea is: it is not practical to keep a precious file to modify and control the configuration, one can lose it. Instead, `adavid.util.JDefaultResource` *generates* a configuration file associated to a class (using reflection) by looking at accessible fields (public, or via a special class). Access can be granted if the class extends a special class `adavid.util.DataResource` which has protected method which can be overriden (by themselves: `super.callMe(args)`) and accessed via package rights by `adavid.util.Bundle`. This is advanced method right usage. It is possible to implements a special interface (`adavid.util.DataResourceAccessible`), which is equivalent though poses security access problems because it opens a bit too much compared to the extension solution. When initializing a class, calls to `adavid.util.Bundle` may save a lot of trouble of initializing components (awt), setting listeners... and it sets up strings as well, which can be *overriden* in this special generated file. More information is given in the generated file as comments. See `adavid.reflect.Bundle, adavid.reflect.ResourceLoader, adavid.reflect.JResource` APIs for more information.

This allows to solve internationalization issues by just changing the content of strings and customization issues by redeclaring new colors, titles, options for the components. A special naming convention is adopted to make this work, see the related classes. Extension to swing components becomes specially powerful if we consider the number of customization options that we have.

Dynamic Linking To Disassemblers/Decompilers

Decompilation settings is handled by the customization capabilities by just redefining a string which contains the list of decompiler interfaces. Then reflection is used to invoke the class from an empty constructor.

1.4 Implementation

It is straight forward from the component based design with the well defined dependencies. Javadoc tags were included, though not everywhere, but this is in the TODO list. Some political choices were: as many static methods were used to allow easy reuse of classes (without instanciating) and methods were declared as much public as possible. Self explanatory naming was used as much as possible with as little inline comments as possible (when needed), for readability (by Assistant!).

Chapter 2

Distribution

A distribution free of charge is available at: *www.docs.uu.se/~adavid/java/javaindex.html* The package is distributed as a single zip archive (`adavid_pack.zip`) which contains all.

2.1 License

The license is LGPL:

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 a program 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.

Our method of protecting your rights has two steps: (1) copyright the library, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the library.

Also, for each distributor's protection, we want to make certain that everyone understands that there is no warranty for this free library. If the library is modified by someone else and passed on, we want its recipients to know that what they have is not the original version, 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 companies distributing free software will individually obtain patent licenses, thus in effect transforming the program into proprietary software. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License, which was designed for utility programs. This license, the GNU Library General Public License, applies to certain designated libraries. This license is quite different from the ordinary one; be sure to read it in full, and don't assume that anything in it is the same as in the ordinary license.

The reason we have a separate public license for some libraries is that they blur the distinction we usually make between modifying or adding to a program and simply using it. Linking a program with a library, without changing the library, is in some sense simply using the library, and is analogous to running a utility program or application program. However, in a textual and legal sense, the linked executable is a combined work, a derivative of the original library, and the ordinary General Public License treats it as such.

Because of this blurred distinction, using the ordinary General Public License for libraries did not effectively promote software sharing, because most developers did not use the libraries. We concluded that weaker conditions might promote sharing better.

However, unrestricted linking of non-free programs would deprive the users of those programs of all benefit from the free status of the libraries themselves. This Library General Public License is intended to permit developers of non-free programs to use free libraries, while preserving your freedom as a user of such programs to change the free libraries that are incorporated in them. (We have not seen how to achieve this as regards changes in header files, but we have achieved it as regards changes in the actual functions of the Library.) The hope is that this will lead to faster development of free libraries.

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, while the latter only works together with the library.

Note that it is possible for a library to be covered by the ordinary General Public License rather than by this special one.

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

0. This License Agreement applies to any software library which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Library 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 compile 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) 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) 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.

c) 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.

d) 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 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.

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 to 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 Library 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

2.2 Files

The `adavid_pack.zip` archive contains:

- `VERSION_1.6`: the file version tag containing the change log, the todo list.
- `INSTALL`: installation manual with a mini how-to.
- `jassistant.bat`: windoze script file, one should edit it to setup special classpaths.
- `Jassistant.pif`: the windoze property file for the script.
- `readme.txt`: a quickhelp and license file.
- `adavid.jar`: the class archive, do not unzip it.
- `adavid.zip`: the source archive, do not unzip it, use Assistant to browse it.
- `adavid_html.zip`: the API generated by javadoc (JDK 1.2.2 under windoze), unzip it if you want to browse it!
- `install`: the linux install script.
- `tosun`: the Sun/Solaris translation script which runs install.
- `Assistant.ps.gz,Assistant.pdf`: this document in postscript compressed and portable document formats.

2.2.1 Executables

The package has different tools/utilities that can be run independently. By running them without argument one gets an help on them. The help is concise and sufficient.

- `adavid.util.JDefaultResource`: generated default configuration files from a class.
- `adavid.util.JMakefile`: generates `Makefile` files for a java project, enters recursively directories. Different configuration options are available.
- `adavid.util.StatSource`: gives statistics on sources.
- `adavid.util.PackageViewer`: the Assistant package viewer, can be started separately.
- `adavid.util.Assistant`: Assistant, see the `readme.txt` file or the `About` dialog for the available options.
- `adavid.reflect.ClassReflect`: test reflection, get information, run it without arguments to get its usage.
- `adavid.reflect.JavaFinder`: test reflection, get information on source code, run it without arguments to get its usage.
- `adavid.reflect.PackageFinder`: test reflection, get information, run it without arguments to get its usage.

2.3 Versioning

The distribution has a double versioning scheme: the package version (current 1.6) and the file version per class. The changes are related in the `VERSION_xx` file which is the tag file for the distribution. Each file has a javadoc `version` tag, as well as `author` tag (set to myself!).

Distributions are automatically generated from the source and html directories.

2.4 Tested Systems

Tested systems are Windows 9x/NT: JDK 1.2.2, Linux RH 5.2/6.0: JDK 1.1.7, Sun/Solaris: JDK 1.2. Kaffe has been tested under Linux but it works poorly. The JIT tya works very well.

The Windows virtual machine implementation is much more performant, it is noticeable. The implementation of the swing-html java browser as well as the html-filter/generator is more a prototype to test the concept. It has serious performance concerns that are on the TODO list.

One important thing to know is that Assistant pushes the virtual machine to its limits and it is possible to end up with segmentation fault or a core dump from the virtual machine! This

is due to a very heavy use of reflection and very high number of loaded classes into the virtual machine which seems not to be able to unload them when not needed.

2.5 TODO

As for every project, here is the TODO list:

- solve the problems between ScrollPane and TNode this seems to work perfectly under Windoze, the motif awt implementation seems not to be that good.
- improve the serie TextArea stream/writer, that's not wonderful.
- review the sort in the package viewer.
- review the class search, .mocha is tried and should not be tried.
- reimplement in a more efficient way the java viewer, now: java=>html->swing.parse->document->display future: java=>document->display.
- history of the most requested classes to avoid retyping them
- 100% Swing GUI.
- improve the javadoc tags, and add little more comments.

Chapter 3

User's Manual

3.1 Basics

Assistant is aimed at helping java application development. Features are:

- browse classes, packages and java source code (available or not).
- browse class API, class hierarchy.
- query class name, automatic completion, automatic search.
- cross references and browsing capabilities: when finding an interface implemented by a queried class, access to it should be as easy as one click; similar capabilities concerning the source code.
- searching and extracting capabilities on the source code.
- practical cut-and-paste from extracted source code, which is of the main interest when implementing an interface.

One can use it as a tool or one can use its components to build other programs. In this view the distribution provides other classes not directly related to Assistant.

The API is javadoc produced and the source can be browsed with Assistant itself (which is much better).

Note that the cursor changes: hand means that something is clickable. Furthermore the focus follows the mouse when something clickable is entered.

3.2 Assistant

A typical commmand under Linux to start Assistant is: `java -ms128m -mx128m -Djava.source.path=$JAVASRC adavid.util.Assistant -redirect -confirm &> /dev/null` which is produced by the install script with the variables set correctly. `$JAVASRC` gives the path of the java sources. It is a normal path format with an addition: `somewhere.zip[inside_zip]` will specify to look at the directory `inside_zip` inside the `somewhere.zip` file. The option `-redirect` tells to redirect outputs to internal dialogs and the option `-confirm` tells to ask for confirmation when exiting.

The main window is depicted in figure 3.1

Possible operations are:

- enter a class/interface in the *"class/interface"* text field, and then press `[ENTER]` to start the query.
- press the *Packages* button to open the package viewer.
- press the *About* button which opens a dialog window with the content of the `readme.txt` (that can be translated and reset using the customization capabilities).

When a query is launched, it may work and then further operations are available, or it fails and then the text is selected and the user has to try another one. An automatic match may be

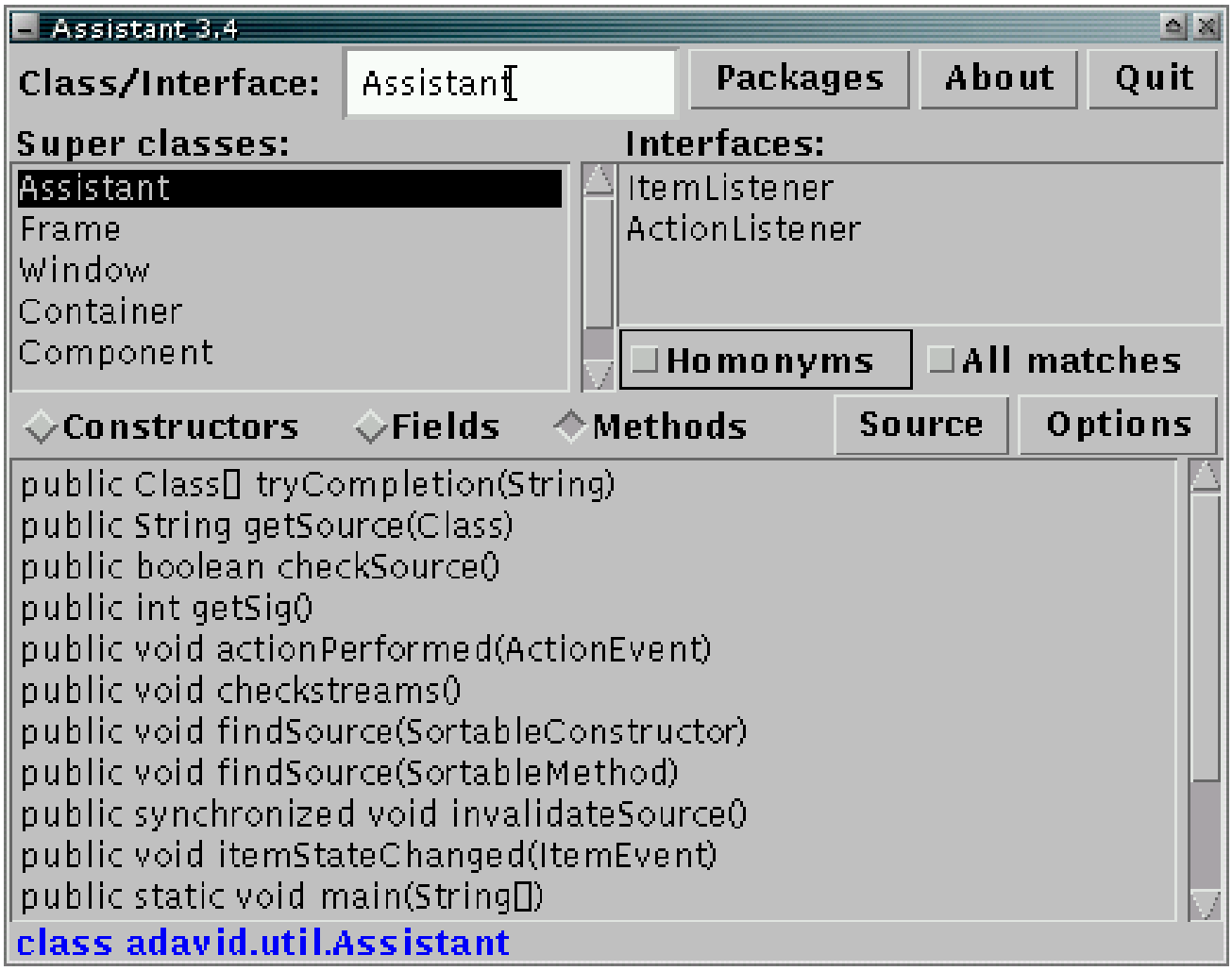

Figure 3.1: Assistant: the main window.

necessary, then the package viewer is invoked since it contains information to search via all the packages.

The opened panel contains a super class list on the left, which gives the class hierarchy (extension hiearchy), a list of implemented interfaces on the right and match options: *homonyms* and *all matches*. There is a list which can be switch with checkboxes to view either constructors, fields or methods.

Further operation are:

- check the *homonyms* checkbox: Assistant tries to find all classes with a given name in all the packages, the panel is updated and the list on the upper left corner gives now all the found classes.
- check the *all matches* checkbox: Assistant tries to match the given name with all the classes in all the packages, and sorts the list by match, the panel is updtaed and the list on the upper left corner gives now all the found classes.
- choose to view constructors/fields/methods
- press the *source* button to retrieve the source code: preference order is configured in the options dialog. The order is java - jj - uncompile - disassemble. The user chooses which first, and in case of failure, Assistant tries the others. In case of total failure, the button is

disabled.
- press the *options* button to open the options dialog depicted in figure 3.2.
- select a method/constructor to get the source of that particular method/constructor.

3.3 Options

The options dialog is depicted in figure 3.2.

Name - Type This dialog commands the behaviour of the Assistant. Names options are available to display names with their full definition or a shortcut, e.g. `java.lang.String` or `String`. This is configurable for the names of the fields/methods or the types (results/arguments).

Access scope This commands the filter to let show only specified scope methods/fields/constructors.

Modifiers This filters the methods/fields/constructors by modifiers. *any* disables the subfilter, *present* requires the presence of the modifier and *absent* requires the absence of the modifier.

Source This commands the order to try to find the source code of a class. Order is java - jj - uncompile - disassemble. The user chooses which first and Assistant tries in this order. "jj" refers to the javacc parser generator source code.

There is a checkbox called *HTML* which enables the HTML source code viewer mode instead of plain text (well it is HTML-like, HTML is not used any more). Another option is "Single Window" to use only one window for the viewer and the arrows "¡¡" and "¿¿" allow to navigate though the history.

Decompiler A choice allows the user to choose among the installed decompilers/disassemblers to use. A button *Options* allows to configure that particular decompiler/disassembler. See the *Decompiler* interface to implement this functionality. A default "gate" is given to use the *mocha* decompiler per default. The gate implements *Decompiler* and is called `mocha.MochaGate`. It is provided as an example for those who want to integrate other decompilers/disassemblers. The gate itself is based on reflection since mocha is obfuscated with invalid names.

Sorting Commands the sorting capabilities: one can enable or not sorting and set the sorting from the modifiers, or not (in that case only the names).

Update packages This button rereads information and rebuild the `.PackageFinder.info` file. It could be useful if a new package is installed and the classpath has not changed. Otherwise Assistant can be started with the *-update* flag.

3.3.1 Source Viewer

The raw text source viewer is depicted in figure 3.3. It provides basic search capabilities in the source code with a find button and a text field to enter the text to find. Note that pressing `[ENTER]` in the text field activates the search. The search is done from the cursor position in the text area which shows the source.

3.3.2 Swing/HTML Source Viewer

This is basically the same viewer as the raw text viewer though with colored-syntax source code (simple, based only on keywords and fairly simple: see highlighted words in the comments). It is depicted in figure 3.4. The main difference is that one can click the anchors which are marks in the java source code, in javadoc format as `see somewhere# function`. If one enters such a

string in the text field, the same happens if one presses the *Open* button. The source viewer may record an history if Assistant is configured to use a single window for the source in HTML mode (though HTML is not used any more).

Figure 3.5 shows what happens if one activates the link in figure 3.4.

Figure 3.6 shows an example of colored disassembled code.

3.4 Package Viewer

It is depicted in figure 3.7. The main panels are: the package tree on the left and the corresponding class tree for a selected package on the right. To open a package, simple click on the blue triangle. Simple click on a package shows information about this package at the bottom of the window. Double click opens that package.

3.5 Package Viewer

Inside the class tree: simple icons mean normal class. Icons marked **i** denote interfaces and icons marked **A** denote abstract classes. These classes are clickable and then short information is given below (a small list which gives the parent class and the implemented interfaces) and a button is updated with their name. Double clicking the classes will send a query to Assistant. This is equivalent to press the button with the name of that class.

Inside the tree, pressing "*" on a focused package opens completely this subtree. The trees are navigable with the arrows and the enter keys.

3.6 Other Components

Other independent components are provided, in the `adavid.swing` and in the `adavid.awt` packages. Use the assistant to get more information on them. They are not related to Assistant.

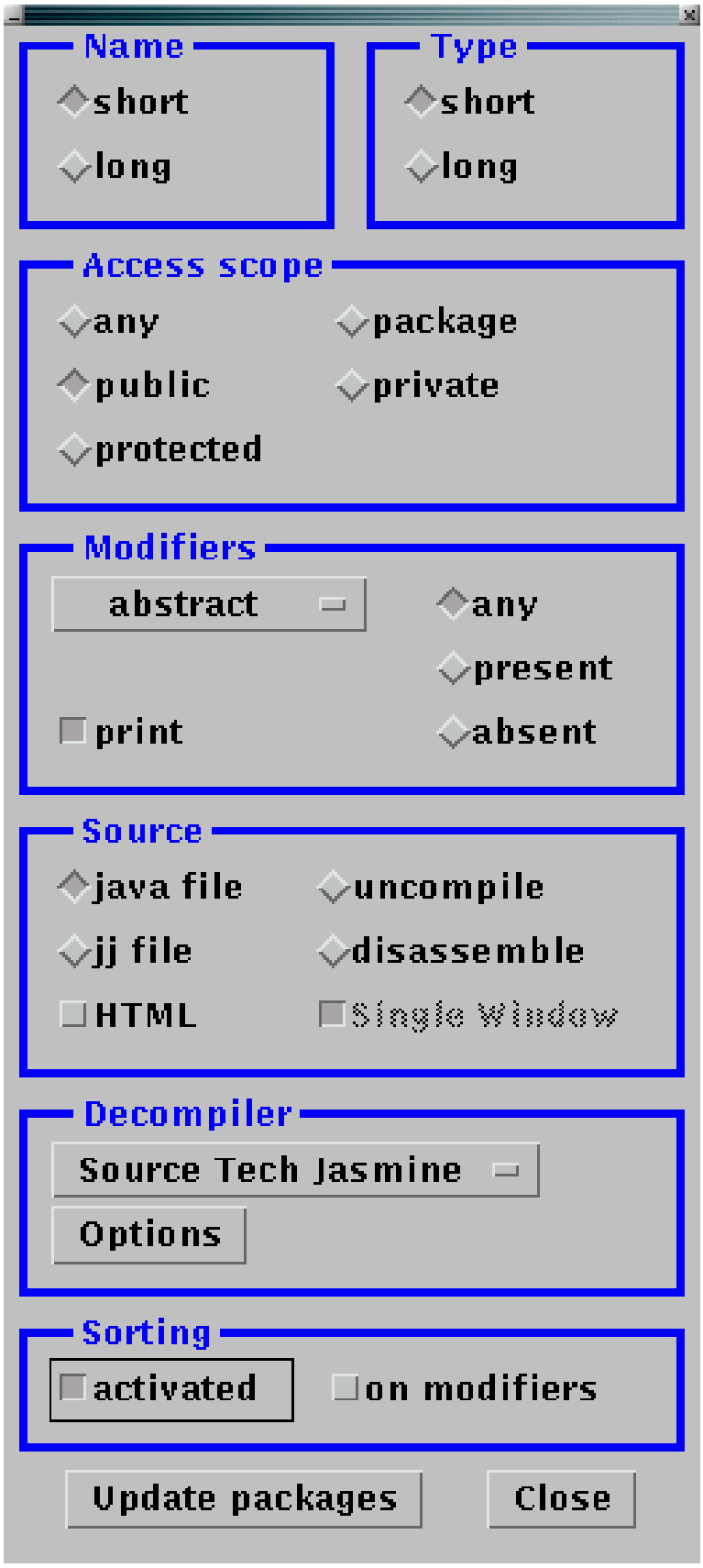

Figure 3.2: Assistant: options dialog window.

Figure 3.3: Assistant: raw source code.

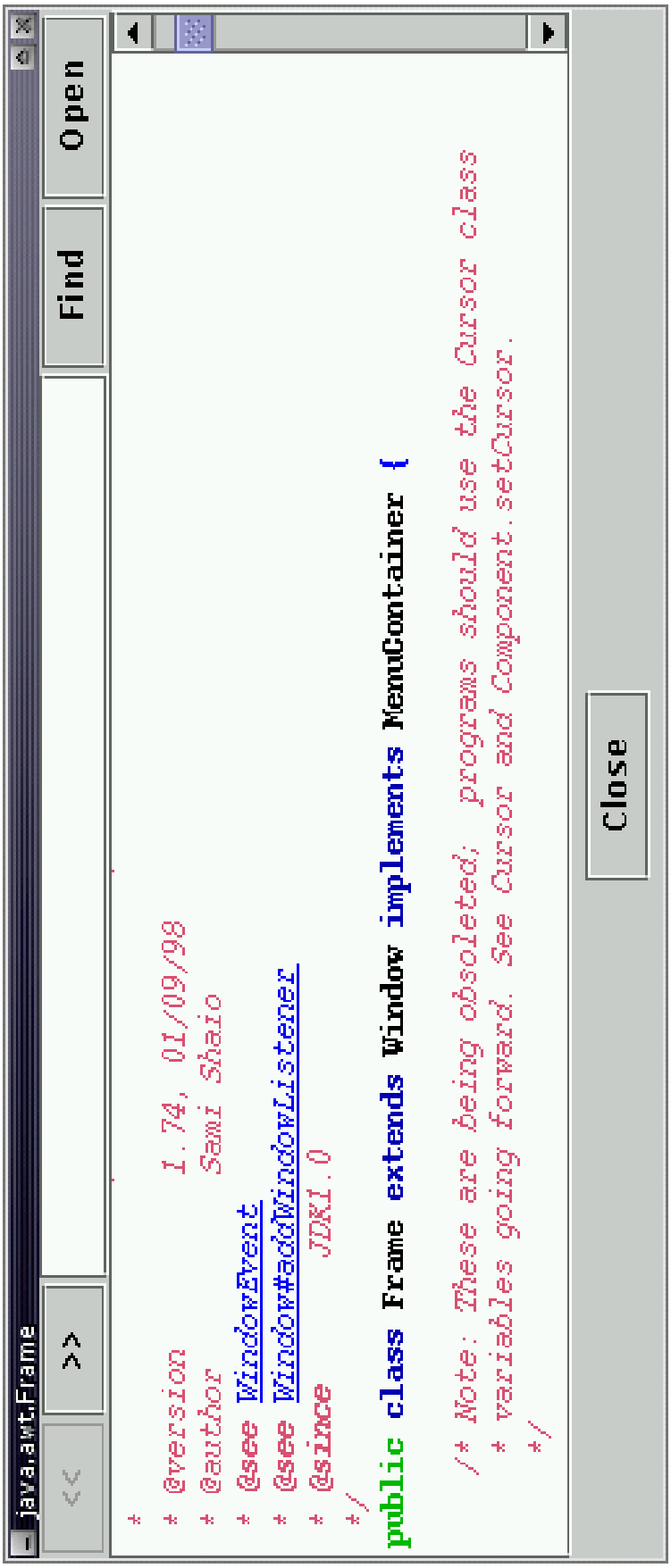

Figure 3.4: Assistant: html formatted source code.

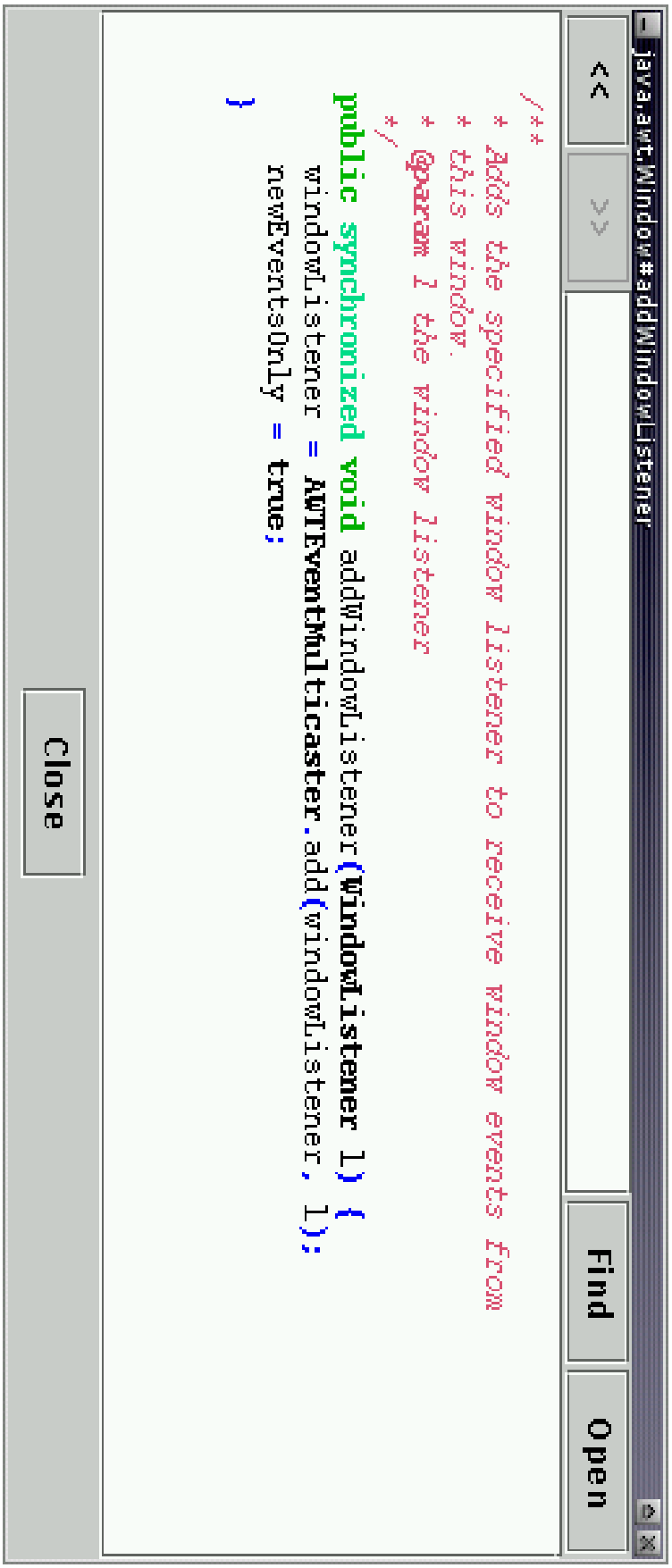

Figure 3.5: Assistant: html formatted source code, following the link in figure 3.4.

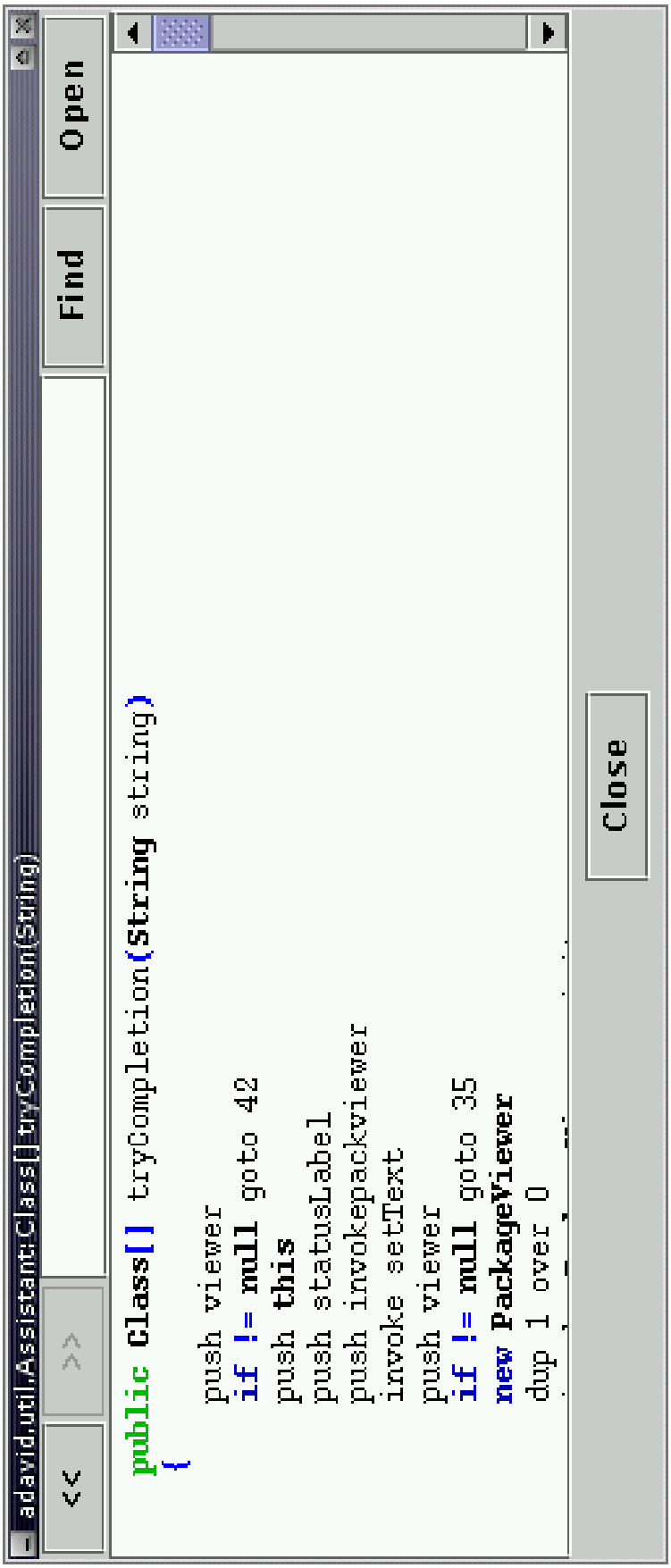

Figure 3.6: Assistant: disassembled function.

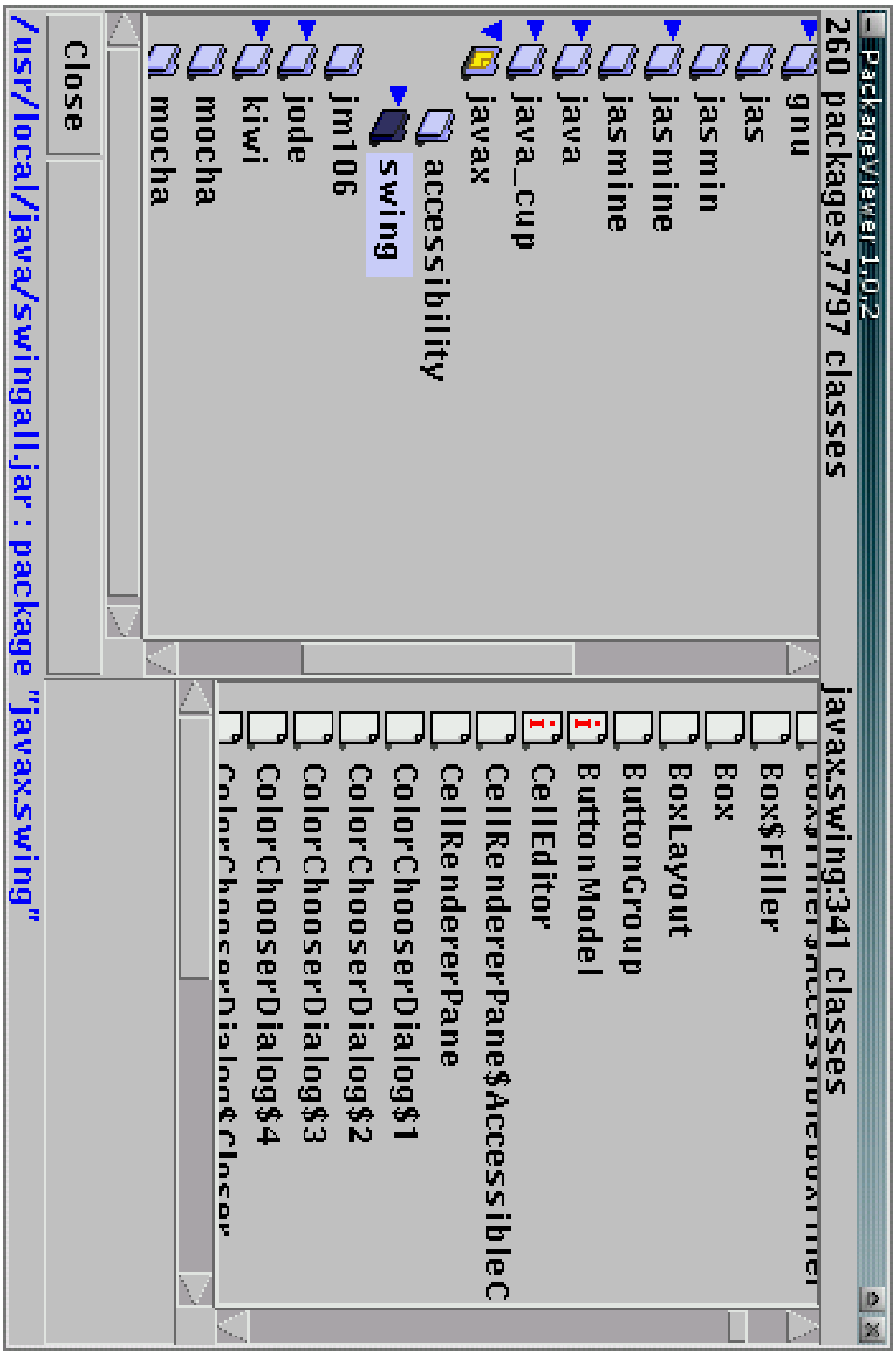

Figure 3.7: Package Viewer: the main window.