A Quick Start Guide to DrJava

A Quick Start Guide to DrJava

Table of Contents

Chapter 1. Introduction

DrJava is a programming environment for Java designed specifically for beginners, but it is also suitable for advanced program development. This document is a quick introduction to DrJava that will help you set it up properly on your computer and introduce you to some of its key features.

For more detailed and technical user documentation, see the User Documentation available from our website: http://www.drjava.org

Chapter 2. Getting Ready to Use DrJava

This chapter describes how to start using DrJava, including where to get the program and how to run it.

Downloading the JDK

To compile programs in DrJava, you must make sure you have a Java JDK (Java Development Kit) installed on your machine. All Java distributions available for downloading come in two forms: a JDK and a JRE (Java Runtime Environment). A JDK distribution consists of a JRE distribution (a Java Virtual Machine implementation) plus a collection of development tools including **javac**, a Java compiler, and **javadoc**, Java documentation generator. DrJava requires a JDK installation because it uses both **javac** and **javadoc** as plugin components. Without these components, DrJava can only edit Java programs; it cannot execute them or generate documentation for them. If you do not have a JDK installed already, you can download one for Windows, Linux, or Solaris computers directly from the Oracle webite. Apple Macintosh machines running Mac OS X already have a JDK installed. To download a JDK from the Oracle website, just follow these steps!

- Go to http://www.oracle.com/technetwork/java/javase/downloads/index.html,
- Do a search for "JDK"
- Select the most recent build of Java from the list of options you see. We recommend Java 6, assuming that you are using a current release of DrJava. Recent releases of DrJava are not compatible with versions of the JDK prior to Java 5 (previously called Java 1.5.0). Older versions of DrJava are compatible with Java 1.3 and Java 1.4.
- Click on the link to download the JDK.

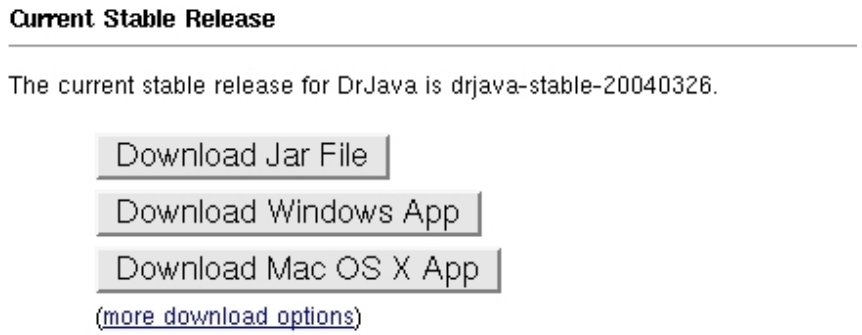
- Now, install the JDK, and you are ready to install and run DrJava!

Downloading DrJava

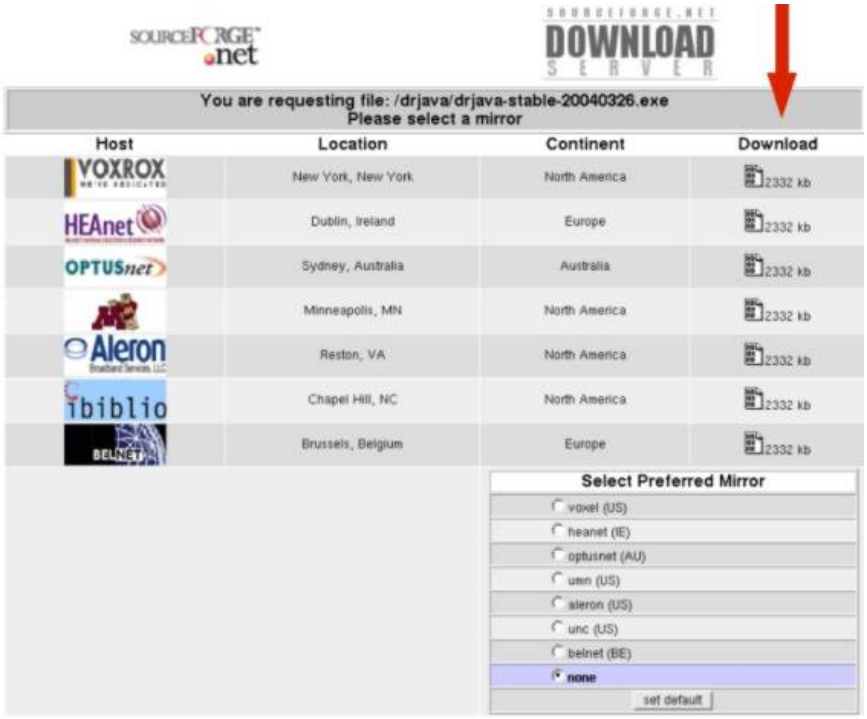
Follow these easy step by step instructions to download DrJava:

- Go to http://www.drjava.org. [http://www.drjava.org]

- Select which build of DrJava is right for you, and click on the appropriate button. We recommend using the most recent stable or beta releases since they support all of the Java 5 language extensions and Java 6 APIs, which make Java programming simpler. We also recommend that Windows users download the `.exe` file, that Mac OS X users use the `osx.tar.gz` version, and that other users download the `.jar` version. If you have problems with a beta release of DrJava, you can roll back to the most recent stable version. Experienced users may want to experiment with other versions of DrJava including the most recent Development Build jar instead. To do so, click on the "more download options" link.



- Click on one of the Download buttons to go to the download server. Choose a site to download the file from, and then click on the corresponding icon in the "Download" column on the far right to start the download.



- The download should begin automatically. When prompted, select where you wish to save the file to disk, and it will be downloaded to your computer.
- Congratulations! You have now downloaded DrJava! Now let's get DrJava running on your machine.

Chapter 3. Using DrJava, the Basics

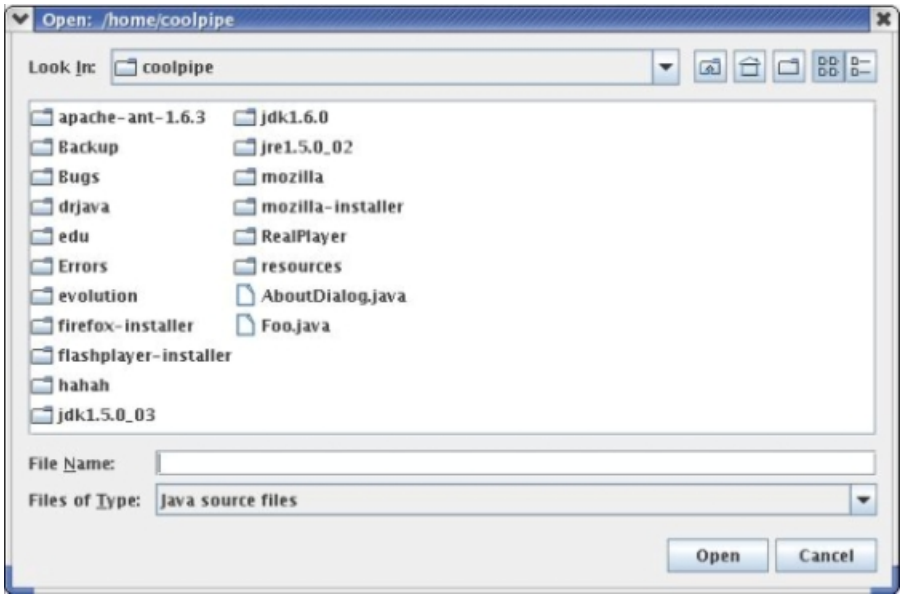
Okay, you've downloaded DrJava. Now, how do you use it? You're about to find out. This chapter will tell you how to run DrJava as well as how to do basic file editing and compilation. You'll also learn how to use the most powerful feature of DrJava: The Interactions Pane

Running DrJava

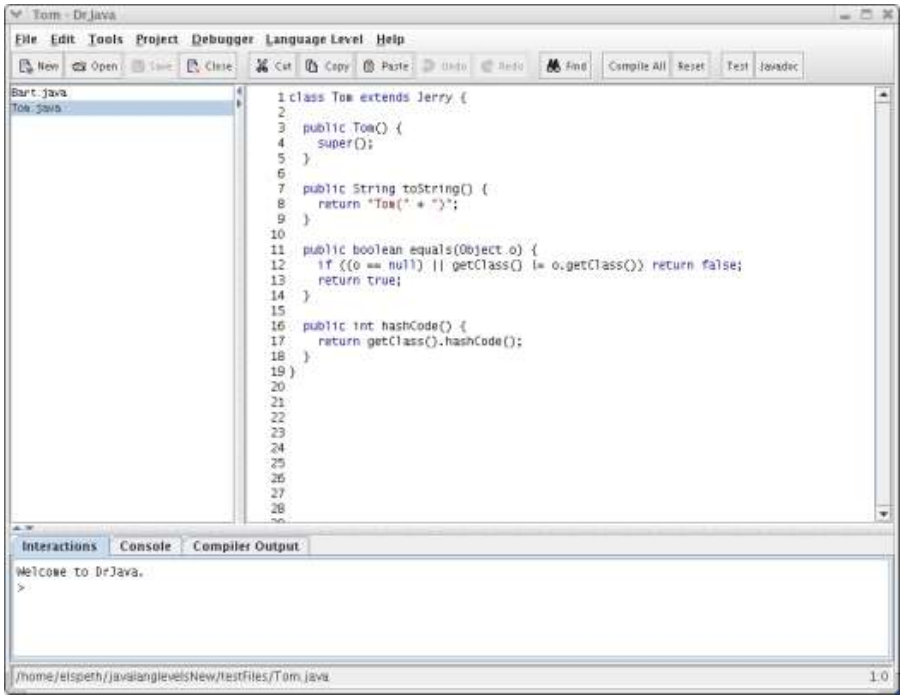
If you are running on Windows, all you need to do is double-click DrJava's `.exe` file, and DrJava will launch. On Mac OSX, unpack the `osx.tar.gz` (you can use StuffIt), and drag the file to Applications. On all other platforms, run DrJava from the command line by giving the command **java -jar nameOfJar.jar** (nameOfJar.jar should be the name of the DrJava jar you downloaded).

Opening and Creating Files

To create a new file, choose "File, New" from the File menu, or click on the "New" button on the toolbar. A new file is created for you, and you can begin typing in it. To open a file, choose "File, Open" from the File menu, or click on the "Open" button on the toolbar. A file chooser window will then be opened. The current directory is displayed at the top of the pop-up window. Navigate through this and select the file or files you wish to open.

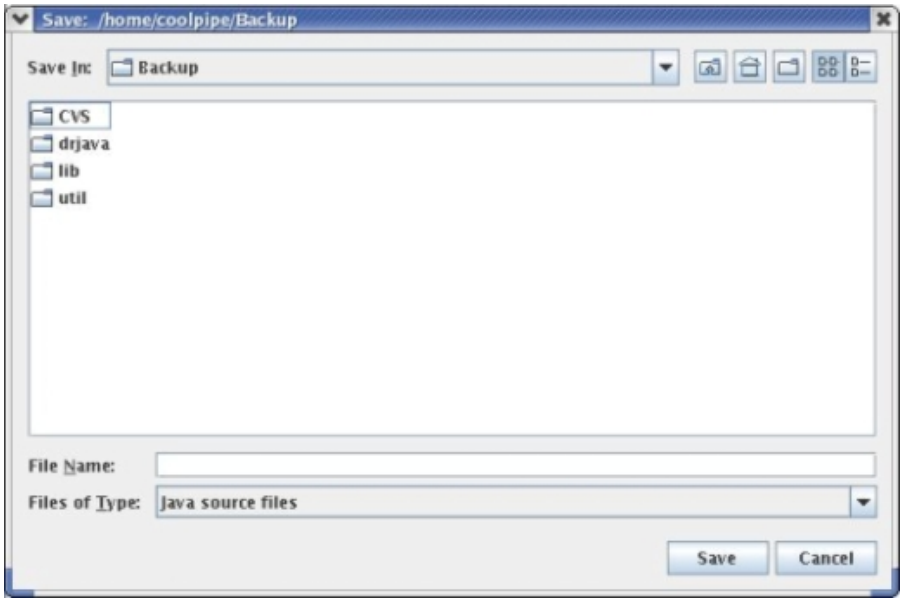


Once you have opened an existing file or created a new file, you will notice that the names of all open files are listed on the left of the screen, and the text of one file is displayed on the right. Select a file name on the left to have the file's text display on the right. The right pane is called the Definitions Pane. It is where you edit your files.



Saving Files

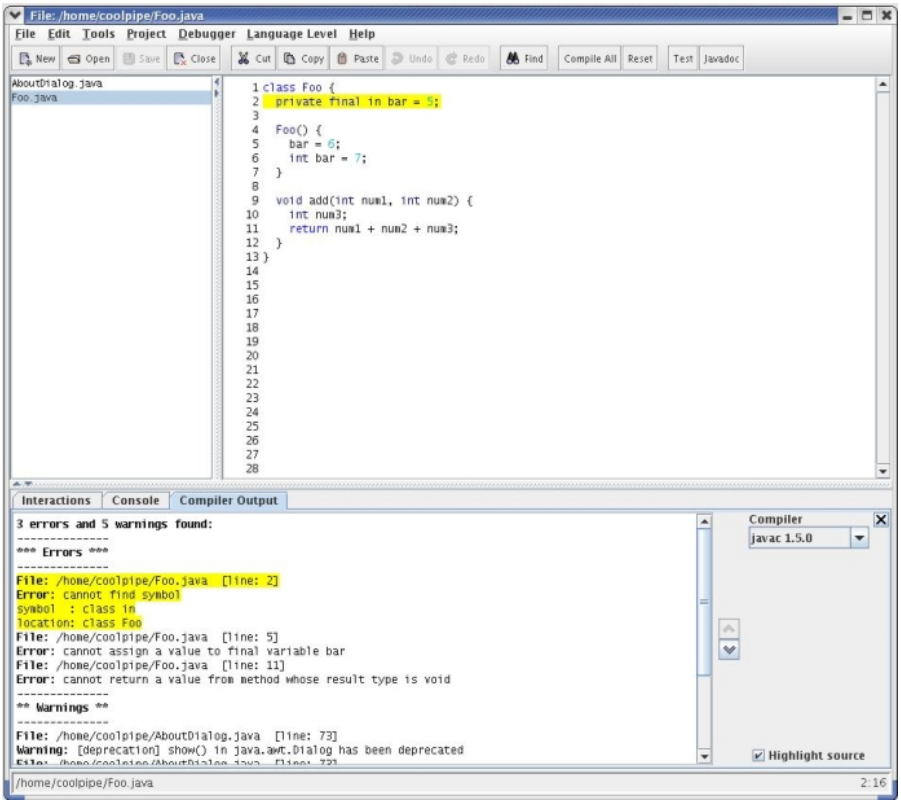
To save a file, either click on the "Save" button or choose "File, Save." If your file has been saved in the past, this will just overwrite the old version. If your file has never been saved, you will be prompted for where you wish to save the file. The current directory is displayed at the top of the pop-up window. You can also use "File, Save As" to save the file with a different name, and "File, Save All" to save all open files.



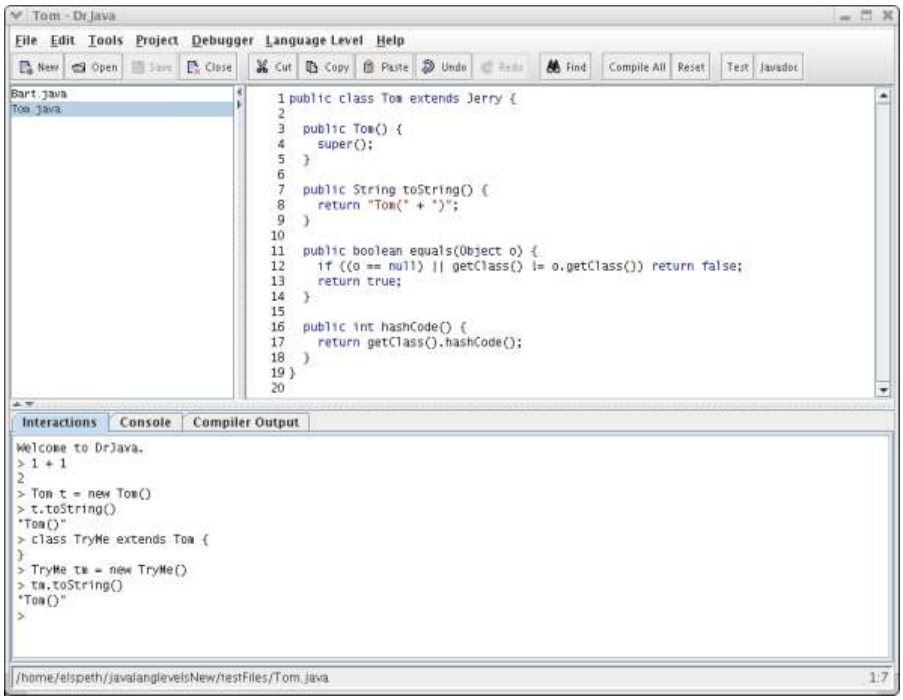
Compiling Files

To compile all open files, click on the "Compile" button. If you want to just compile a specific file, right click on its name on the left listing of files, and select Compile Current Document. Once the compile is

completed, the results are displayed on the Compiler Output tab at the bottom of the screen. If there are no errors, you will see "Last compilation completed successfully". If there are errors, they will be listed. If both errors and warnings are found, the number of each is listed, and errors will be listed before warnings. If you click on an error, it will highlight the place in your source code where the error is. Use the up and down arrow buttons to navigate between error messages.

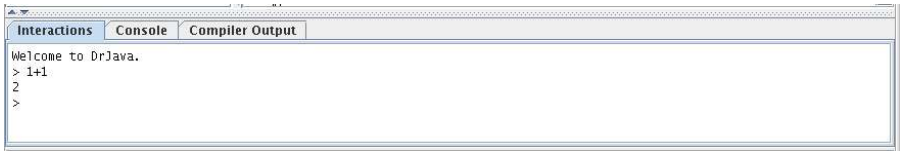


The Interactions Pane

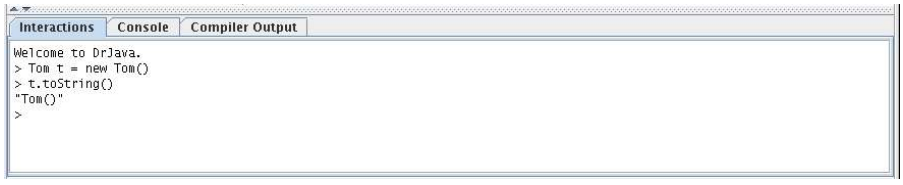


The Interactions Pane is one of the best and most distinctive features of DrJava. It offers both beginning students and more experienced programmers the ability to quickly try out code without having to write cumbersome main methods.

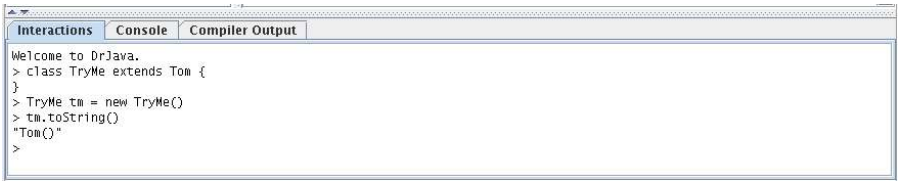
1. One way to use the Interactions Pane is to input basic Java expressions like 1+1. Those expressions will then be interpreted and evaluated, and the result will be displayed. This example shows how to do basic calculations in the Interactions Pane:



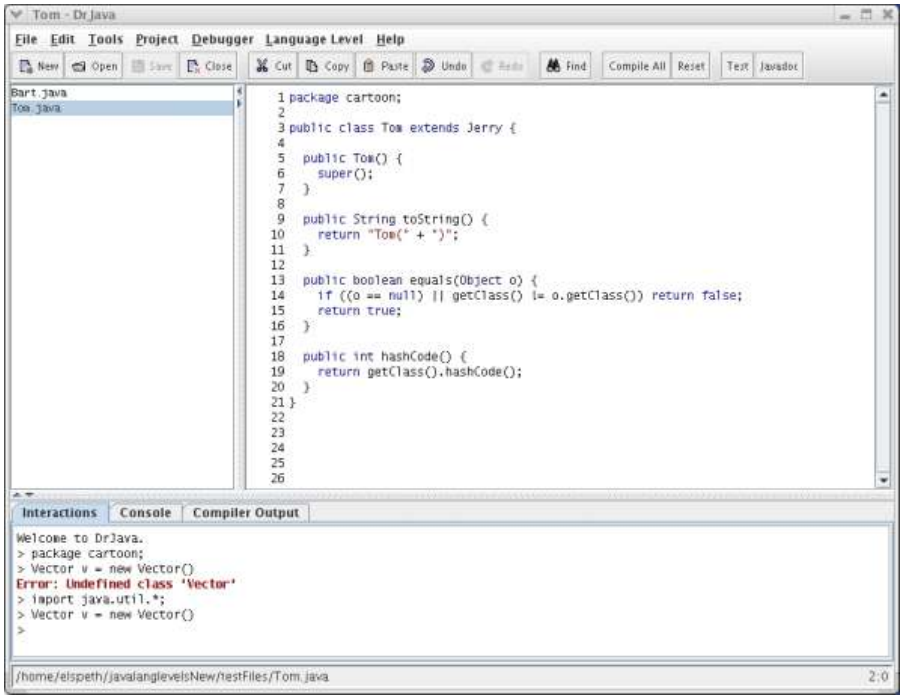
2. Another way to use the Interactions Pane is to instantiate classes and then call their methods--this allows you to quickly and easily verify the behavior of methods. This example shows how to instantiate a class and call one of its methods:



3. In addition, you can define classes and interfaces in the Interactions Pane, and then instantiate these classes and test them. This example shows how to create a class and call one of its methods:

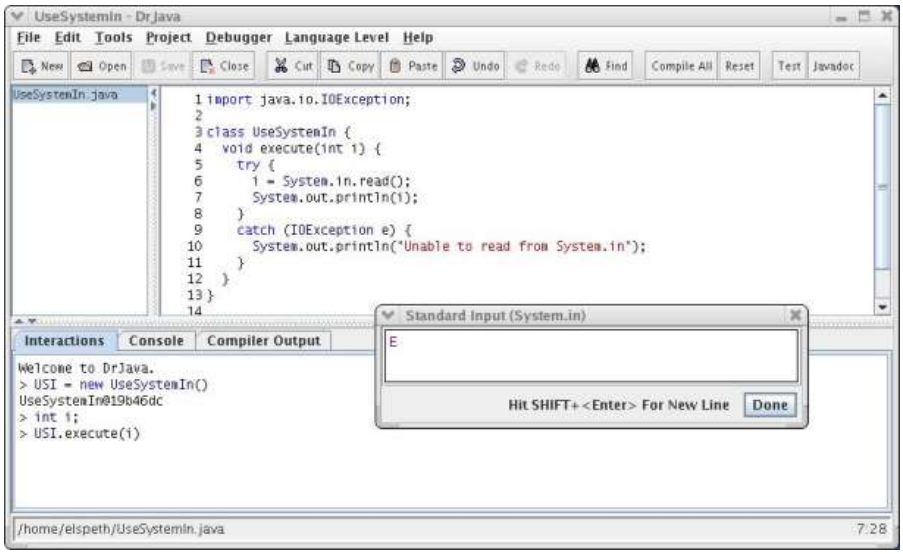


4. It is important to remember that if you are working with classes that are packaged in the Definitions Pane (where the file's text is), it is important to make the same package declaration in the Interactions Pane. Also, you need to import any other packages you wish to use, like you would in a normal `.java` file. In this example, you can see how the package is used, and also that until the `java.util` package is imported, `Vector` cannot be used:

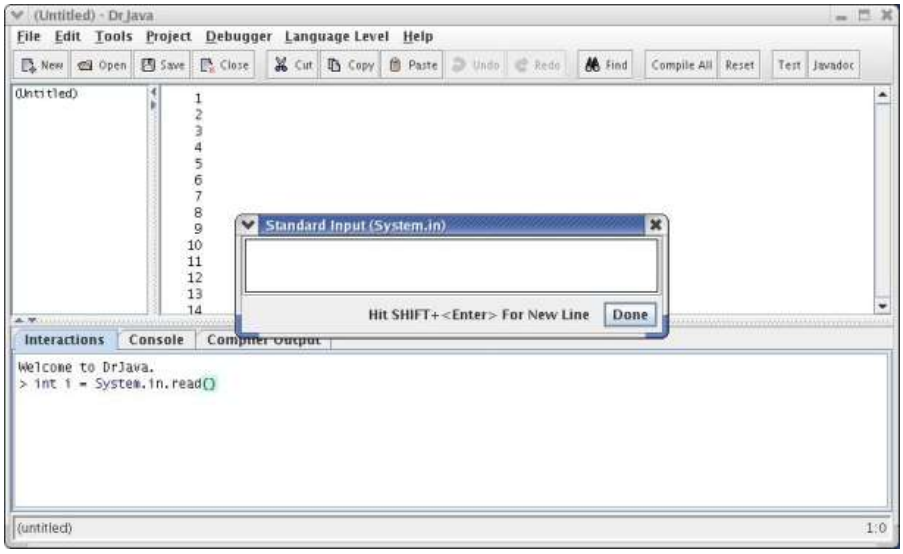


System.in

Many beginner programs rely on keyboard input read through `System.in`. DrJava provides full support for this. There are two ways to use `System.in` in DrJava. First, you can invoke a method in your code that relies on `System.in` in the Interactions Pane. When you do this, the System.in input box will appear. Type your input, and press Return.



Alternatively, you can use the `System.in.read()` method in the Interactions Pane directly. When the input box appears, type your text and then either press Return.



When the the input box appears in the Interactions Pane, you can choose to close the input stream by selecting the menu item "Tools, Interactions & Console, Close System.in", or by pressing the keyboard shortcut for it, which is `Ctrl-D` . The shortcut is labeled `Close System.in` in the Key Bindings section of the preferences.

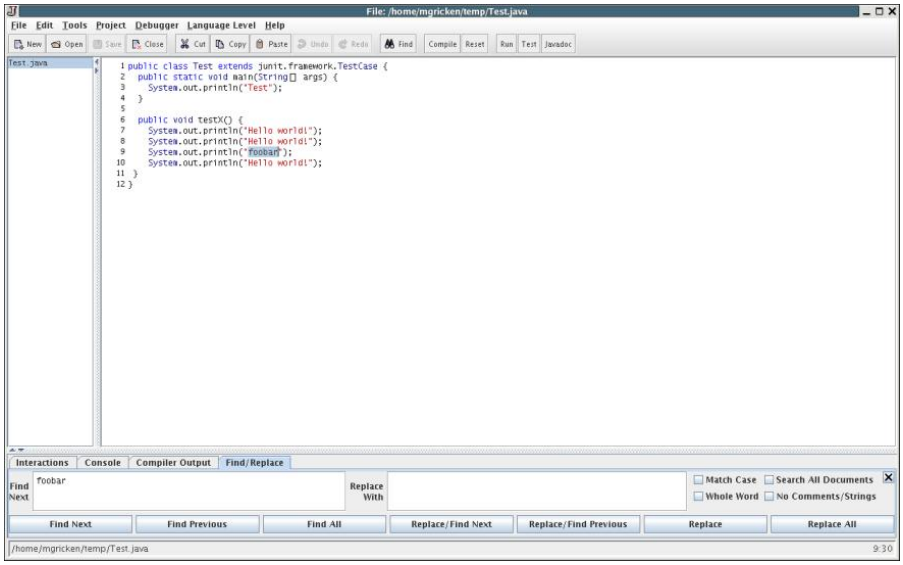
Here is an example of closing the input stream. The text in square brackets was entered by the user.

```
Welcome to DrJava.  Working directory is /Users/Shared/drjava
> System.in.read()
 [1]
49
> System.in.read()
10
> System.in.read() // press Ctrl-D now
 []
```

```
-1
>
```

The user first types '1' and then presses Return. This lets DrJava read a 49, which is the ASCII code for the character '1', and then 10, which is the ASCII code for the new line created by Return. In the second input box, the user pressed Ctrl-D immediately to close the input stream. This lets DrJava read -1, indicating of the end of the stream.

Find and Replace



The Find and Replace menu is a useful way to search for specific text in your files, and replace it, if you so choose. To launch the Find/Replace tab, either use the keyboard shortcut Ctrl-F or choose "Edit, Find and Replace" from the menu at the top of the DrJava window. Once you do this, the Find/Replace tab will appear at the bottom of the screen. Type the text you want to search for in the "Find Next" box, and if you want to replace it with anything, type that in the "Replace With" box. Then, chose which of the four options to the right of those text boxes that you want to use, and press one of the buttons at the bottom, or the Enter key. The direction of the search executed by pressing the Enter key depends on the direction of the last search. (The last direction is indicated by the label of the Find box: "Find Next" or "Find Prev").

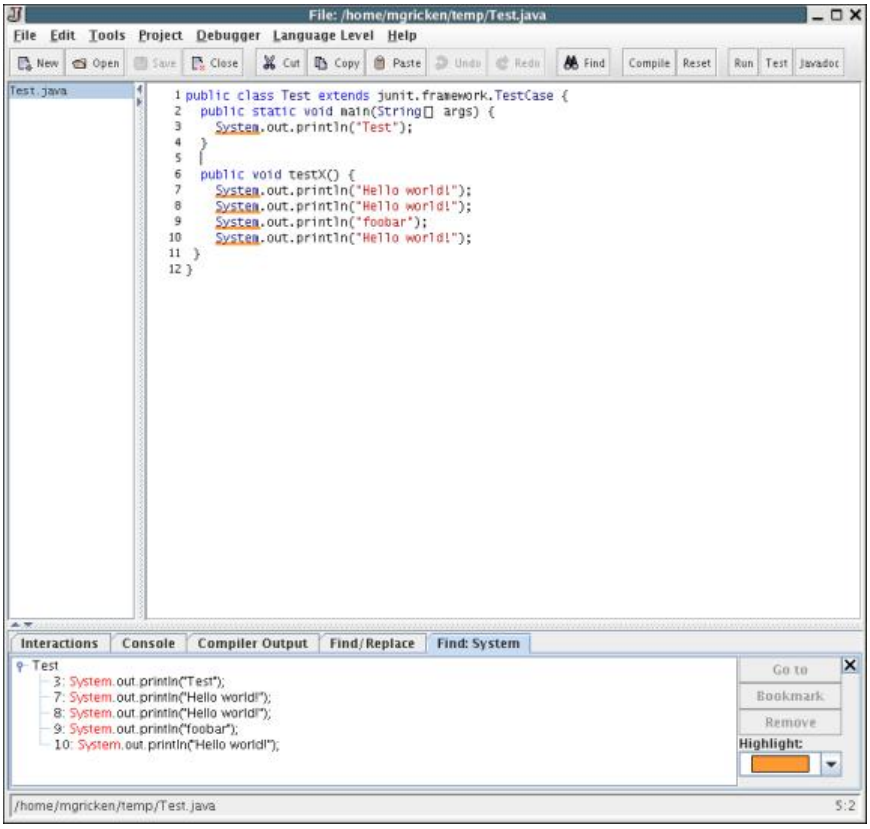
- **Match Case:** Makes the search becomes case sensitive.
- **Search All Documents:** Causes your search text to be looked for in all open files.
- **Whole Word:** Only finds instances of your search text that are preceeded and followed by either a space, a period, or an open or close parenthesis.
- **No Comments/String:** Ignores any instances found within comments or strings.
- **Search Selection Only:** Only used with Find/Replace All. Allows user to find/replace all instances of a word or phrase within a highlighted portion of the document.

To go from instance to instance of your search text, use the four buttons at the bottom.

- **Find Next:** Finds the next instance. Switches the label of the Find box to "Find Next", indicating the direction of the next search.

- **Find Previous:** Finds the previous instance. Switches the label of the Find box to "Find Prev", indicating the direction of the next search.

- **Find All:** Find all occurrences of the search string. This opens a new "Find: word" pane, where "word" is replaced with the search string. In this pane, view all found instances and jump to their locations in the text. You can have as many "Find All" panes as you like open concurrently. The "Find All" tool also automatically underlines the occurrences. In the screenshot below, the user searched for the word "System":



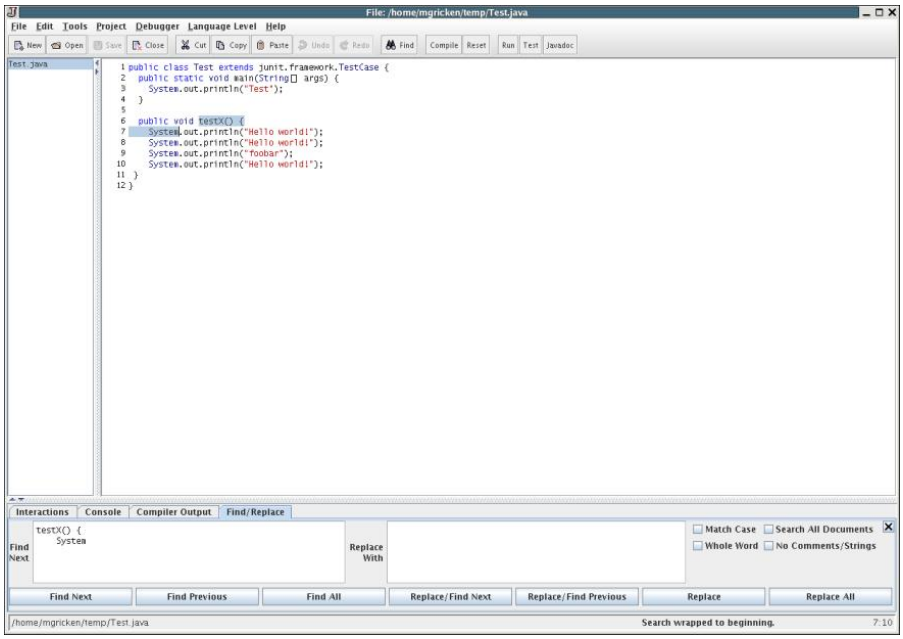
- **Replace/Find Next:** Replaces the current instance with your replace text and then finds the next instance. Switches the label of the Find box to "Find Next", indicating the direction of the next search

- **Replace/Find Previous:** Replaces the current instance with your replace text and then finds the previous instance. Switches the label of the Find box to "Find Previous", indicating the direction of the next search

- **Replace:** Replaces the current instance with your replace text.

- **Replace All:** Replaces all instances of the search text with the replace text automatically

A relatively new feature of Find and Replace is the ability to search across more than one line of text. The Find and Replace text boxes can accept more than one line of text. To use this feature you have two options. First, you can copy/paste directly into the Find and Replace boxes and search/replace as normal. Second, you can type text directly into the boxes, and when you want to create a new line press Ctrl-Enter (Note: Pressing only the Enter key when inside the Find box executes a Find. It DOES NOT create a new line. Both Enter and Ctrl-Enter create a new line inside the Replace box).

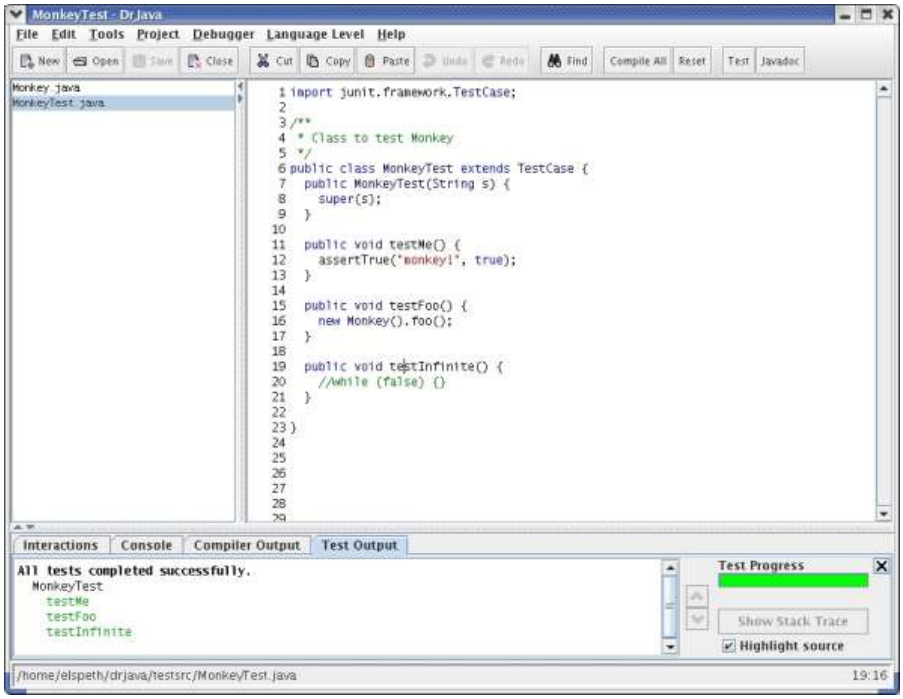

File: /home/mgricken/temp/Test.java
File Edit Tools Project Debugger Language Level Help
New Open Save Close Cut Copy Paste Undo Redo Find Compile Reset Run Test Javadoc
Test.java
1 public class Test extends junit.framework.TestCase {
2 public static void main(String[] args) {
3 System.out.println("Test");
4 }
5
6 public void testX() {
7 System.out.println("Hello world!");
8 System.out.println("Hello world!");
9 System.out.println("foobar");
10 System.out.println("Hello world!");
11 }
12 }
Interactions Console Compiler Output Find/Replace
Find Next
testX() {
System
Replace With
Match Case Search All Documents
Whole Word No Comments/Strings
Find Next Find Previous Find All Replace/Find Next Replace/Find Previous Replace Replace All
/home/mgricken/temp/Test.java
Search wrapped to beginning.
7:10

Chapter 4. Advanced Features

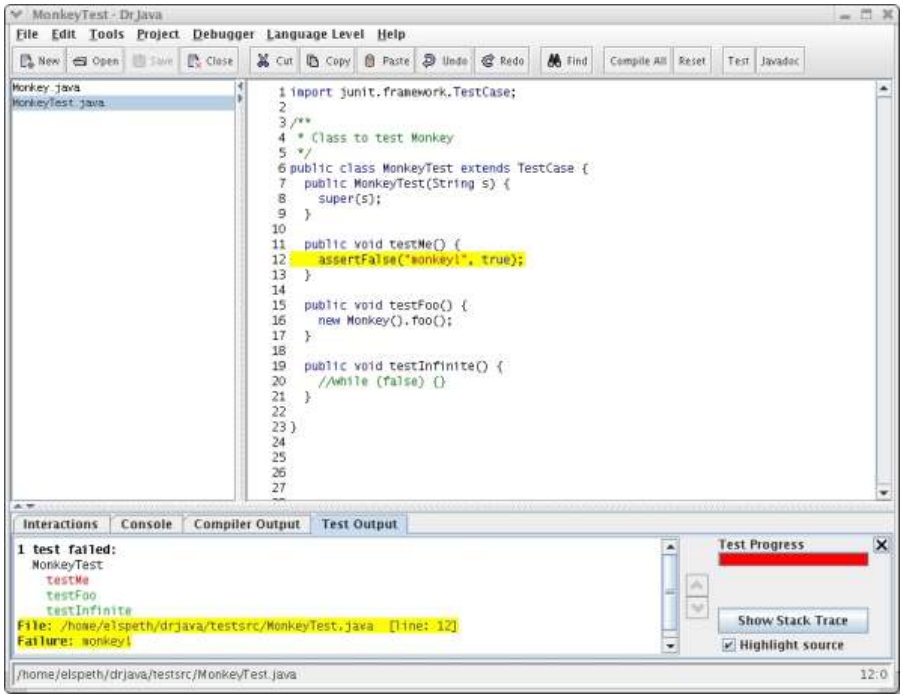
DrJava provides support for many advanced features.

JUnit Testing of Files

DrJava offers a JUnit test facility. For information about JUnit and JUnit tests, check out http://www.junit.org. [http://www.junit.org] Click the "Test" button to test all open JUnit tests, or select the file you want to test on the left menu, right click, and select "Test Current Document". The tests will be run, and the output will be displayed on the Test Output tab at the bottom of the screen. If all your tests passed, you will see a green bar.



If any tests failed, you will see a red bar. All failing tests will be listed in the test pane. Click on one, and its location in the source code will be highlighted.



Generating JavaDoc Documentation

DrJava offers the ability to generate JavaDoc documentation for the user's java files. For information about what JavaDoc is and how to format your comments to take advantage of it, see http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html.

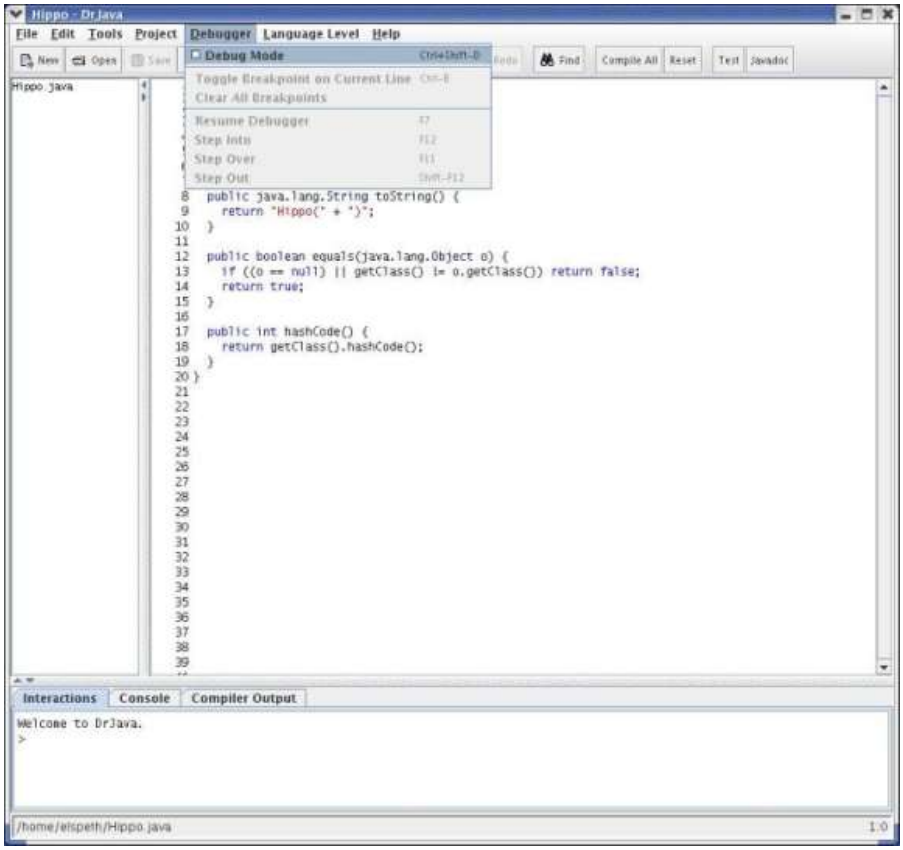
To generate JavaDoc for all files in the same directories as your open files, click the "JavaDoc" button on the toolbar. You will then be asked where you want to save the javadoc files. Select a location and click OK. Your JavaDoc will be generated for you.

To generate JavaDoc for one specific file, select it from the open list of files and right click and select "Preview JavaDoc" for current Document. You will be able to view the JavaDoc for that file in the JavaDoc viewer.

The Debugger

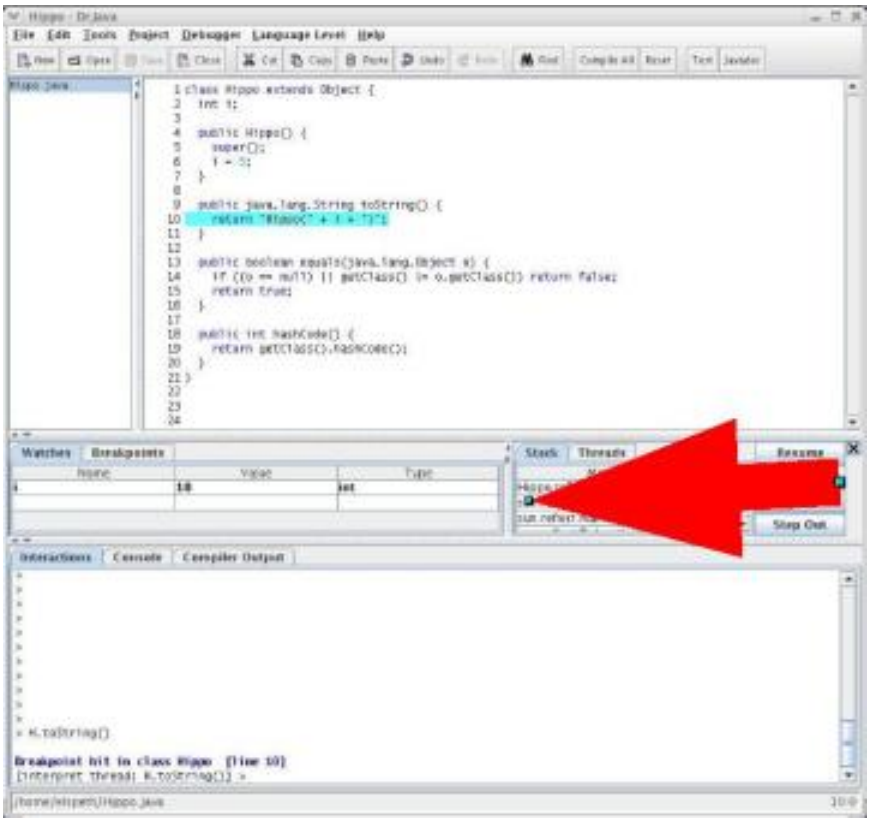
The debugger allows you to step through the execution of a program and stop at any point you choose so that you can track down bugs in your code. What makes our debugger an unusually powerful tool is that it allows you to modify the values of fields and variables *while debugging*! First, we will explain a little bit about how the debugger works in general, and then we will explain how you can use the Interactions Pane to modify field and variable values during runtime and thus more quickly zero in on problems in your code.

To enable the debugger, select "Debugger, Debug Mode". This will launch the debug panel in the lower part of DrJava, right above the Interactions Pane.

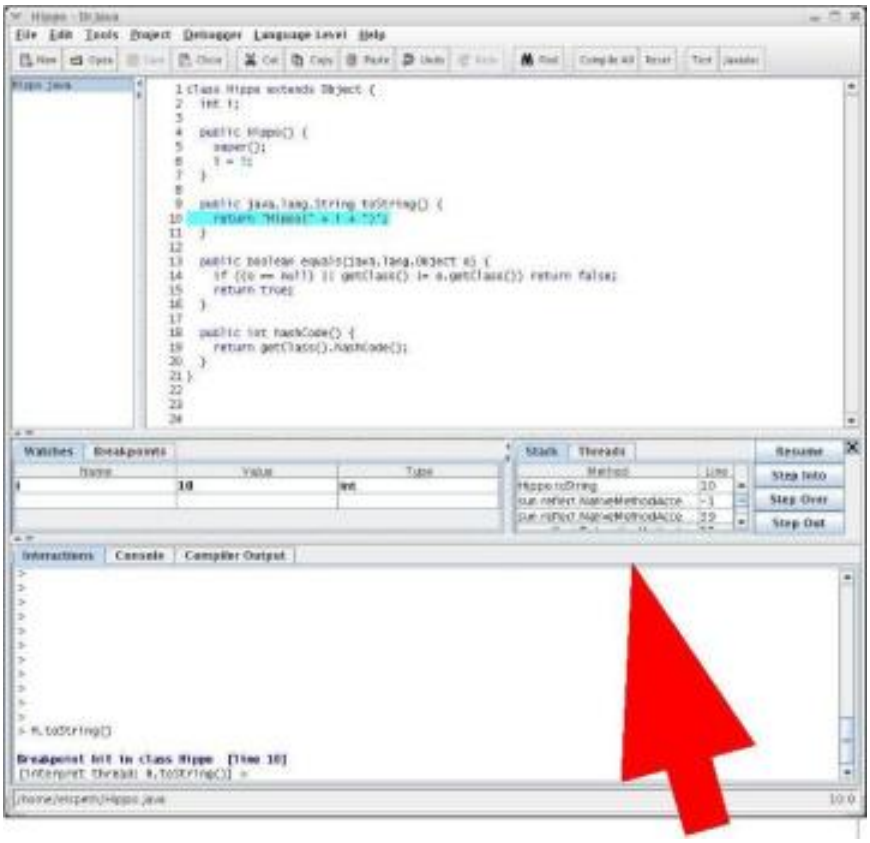


Debug Panel: The Debug Panel has three tabs to help you track control flow in your program. These tabs are labeled Watches, Stack, and Threads.

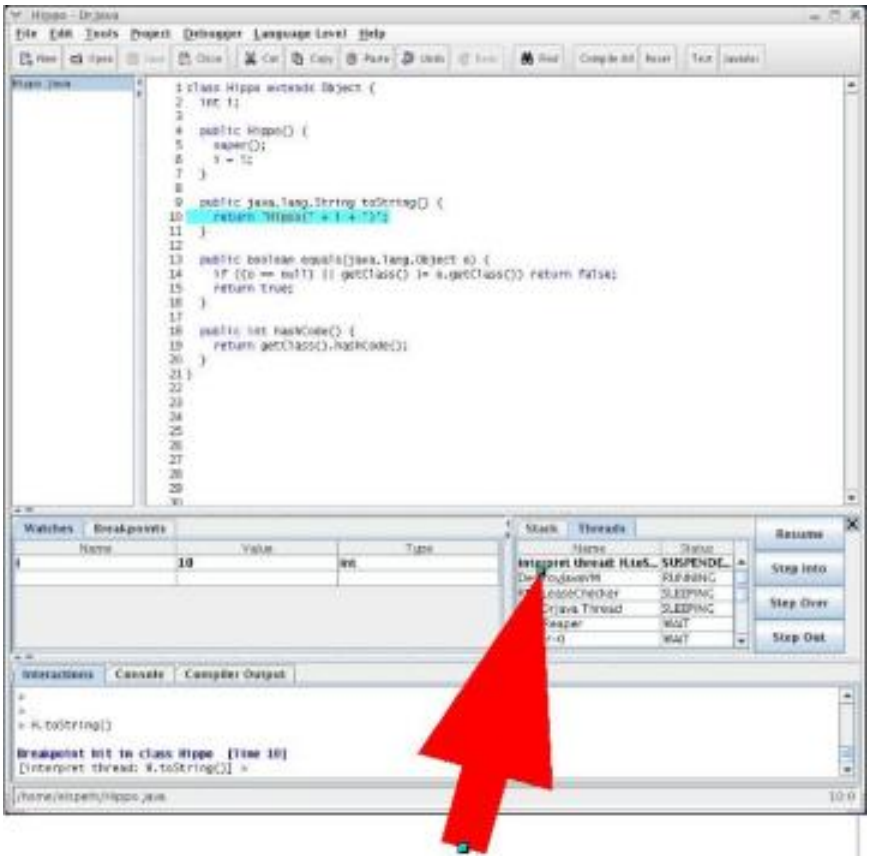
- **Watches:** This is a table that displays the instantaneous values of specified fields and variables given the current line that the debugger is on. To add a watch, click in the leftmost column on the first empty row. A cursor should appear and you can type in the name of the field or variable. You can also put in expressions that require evaluation such as x + 1 or array accesses.



- **Stack:** This a table that lists the current stackframes. Use this to determine the trail of methods that have been invoked to arrive at the current line in the execution. You can double-click on a particular stack frame to scroll to center it in the Definitions Pane.



- **Threads:** This is list of the threads that are running in the virtual machine. If there are multiple threads that are suspended, you can set the active thread (the thread whose execution you're examining) by double-clicking on the desired thread.



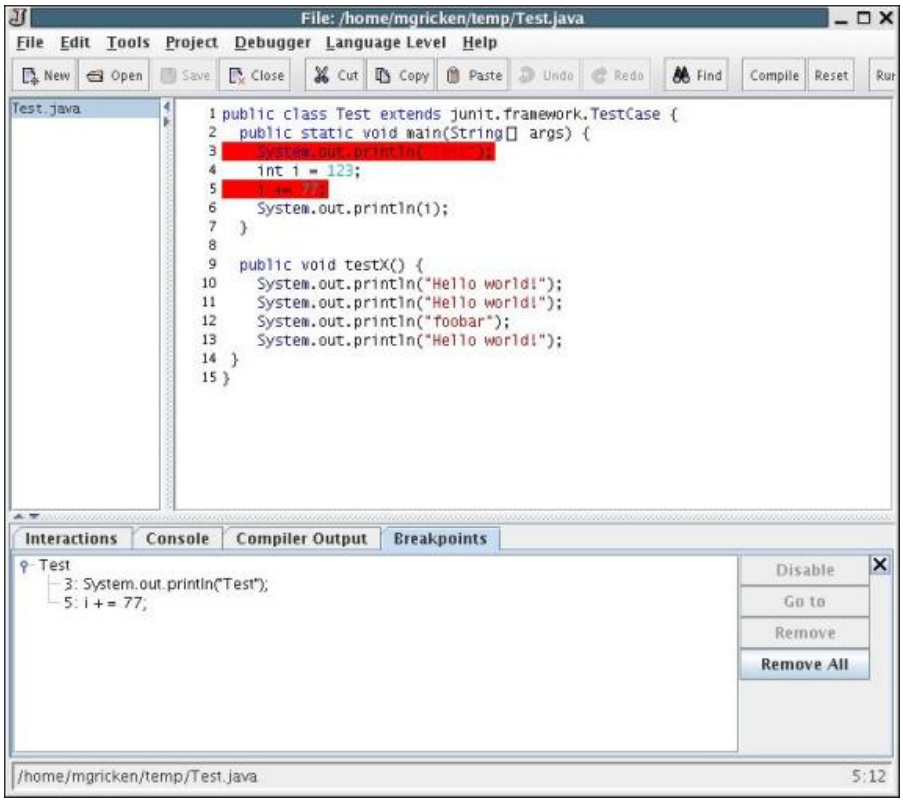
Breakpoints. Another important part of the debugger is the breakpoints panel. Even when the debugger is not enabled, you can set breakpoints using the Ctrl+B shortcut or by selecting the "Toggle Breakpoint" command from the Debugger menu or the context menu. This will highlight the current line in red, and the debugger will stop the running program when it reaches this line. To remove a breakpoint again, place the cursor on the same line and press Ctrl+B again.

To view all the breakpoints you have set and to navigate between them, you can open the Breakpoints Panel by pressing Ctrl+Shift+B or selecting the "Breakpoints" command from the Debugger menu. In this panel, all currently set breakpoints are listed, sorted by document and line number. To move the cursor to the location of a breakpoint, double-click on one of the entries in the Breakpoints panel, or select it and press the "Go to" button.

You can also select one or more breakpoints and use the "Enable" or "Disable" button to temporarily enable or disable the selected breakpoints. If a breakpoint is disabled, it remains set, but the program will not stop there. This is useful if you may need a breakpoint again later, but want it ignored right now.

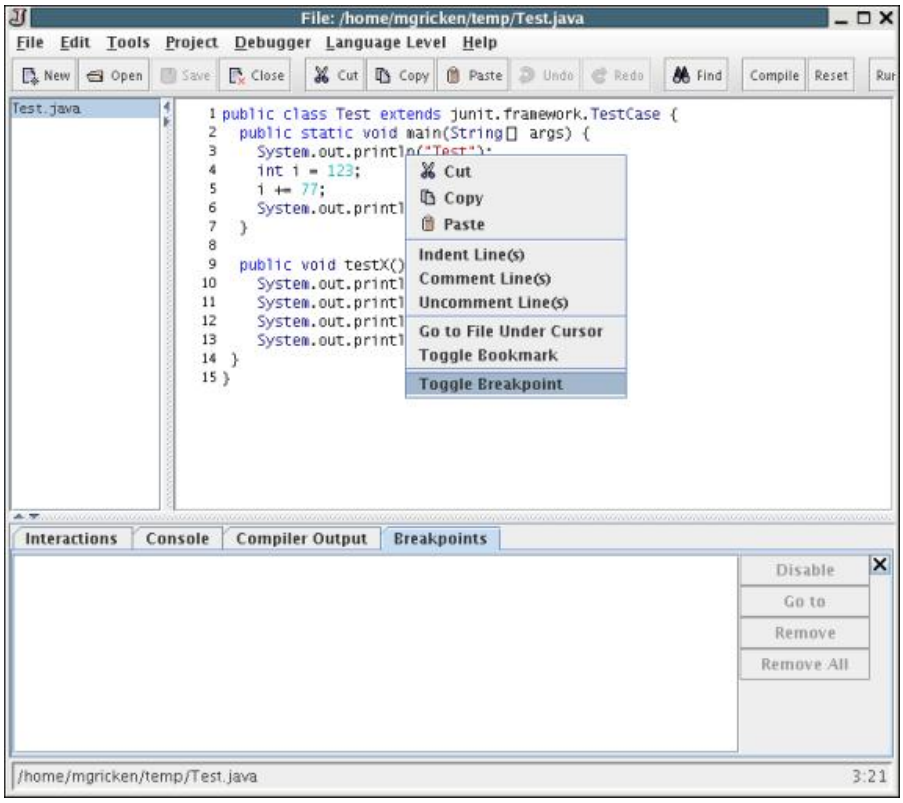
With the "Remove" button, you can remove one or more selected breakpoints. The "Remove All" button clears the entire list of breakpoints.

Breakpoints are considered part of a project and are therefore saved to and restored from a project file.

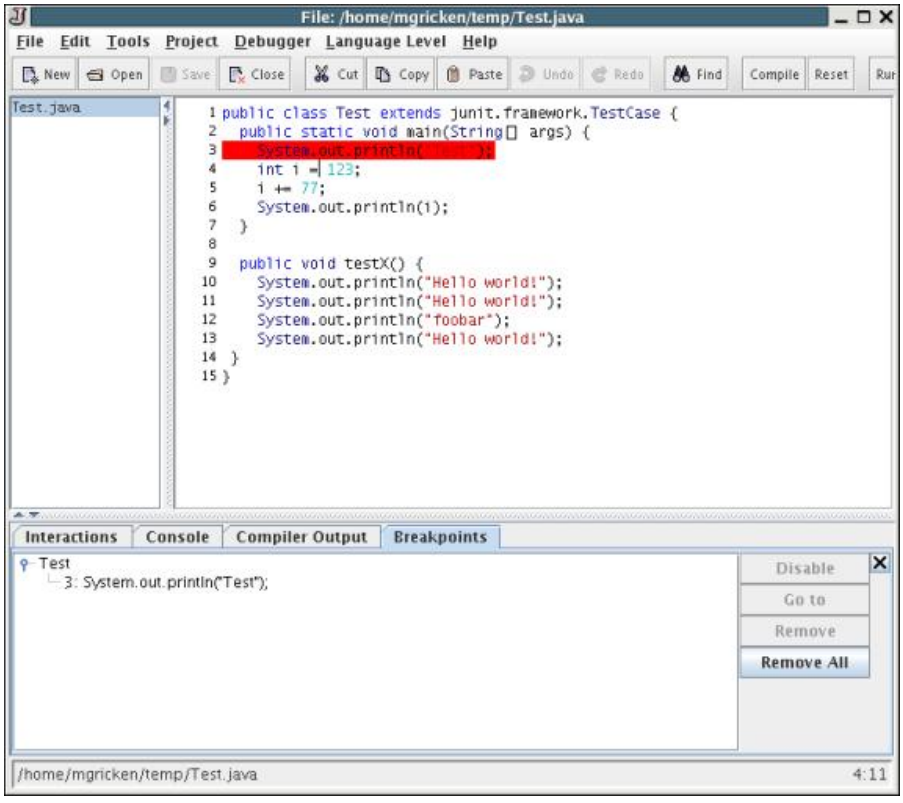


Using the Debugger

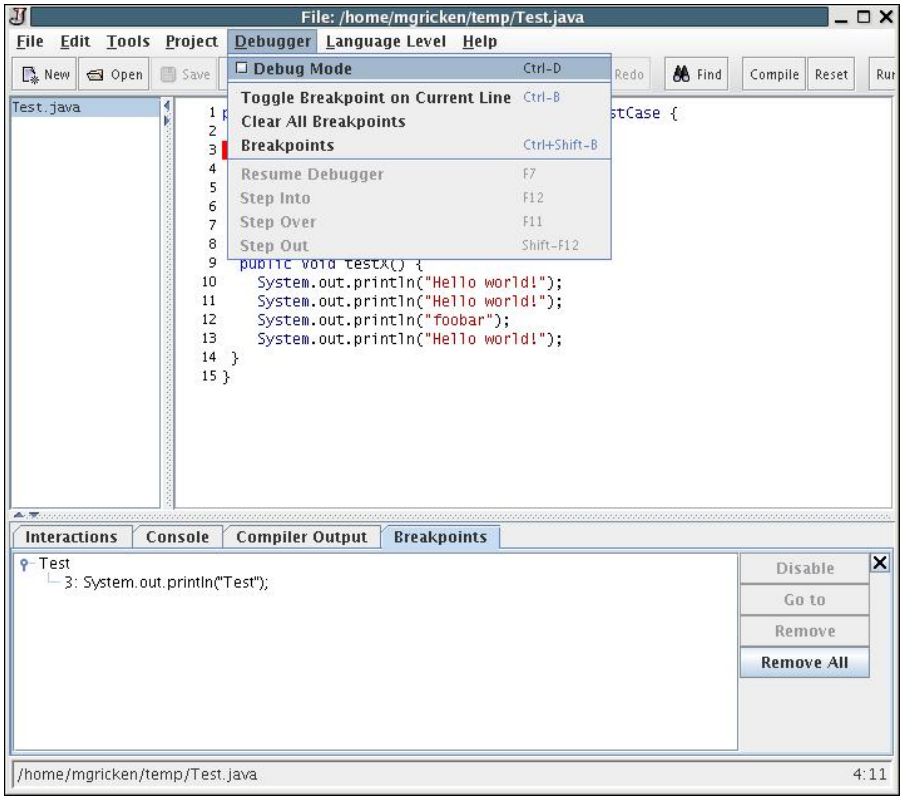
To use the debugger, you must first set a *breakpoint* on the line(s) of code on which you want execution to stop. To do this, right click on the line of code that you want the breakpoint on and select Toggle Breakpoint from the menu.



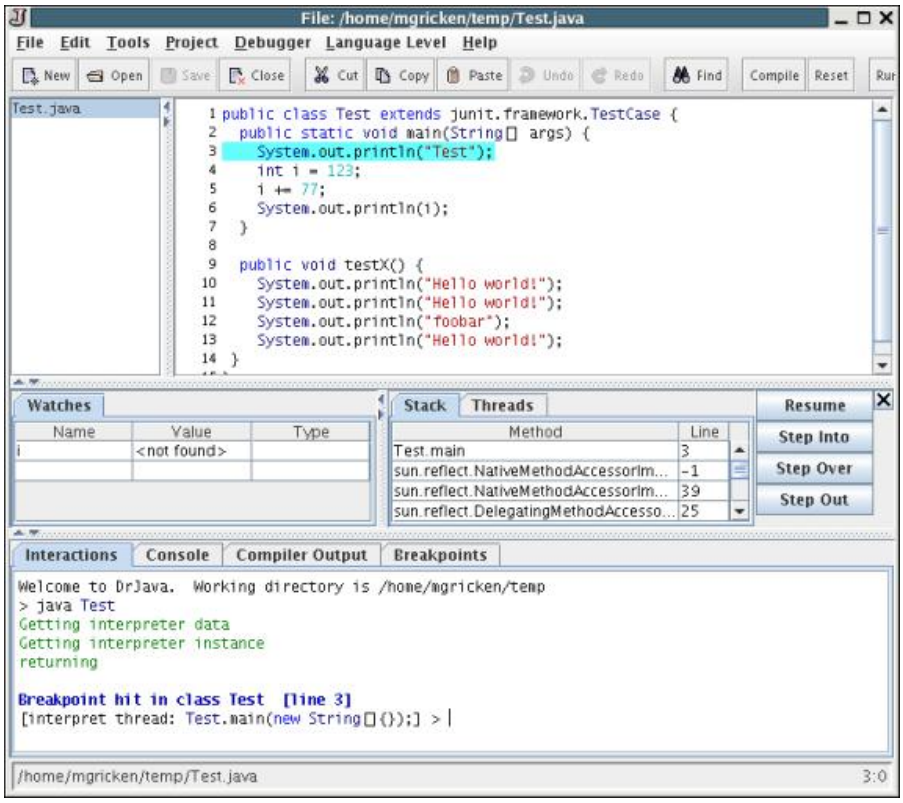
Once a breakpoint is set, it is highlighted in red, like this:



Once you have set the breakpoint(s), enable the debugger by pressing Ctrl+D or invoking the "Debug Mode" command of the Debugger menu.



Now enter a statement in the Interactions Pane that will run the method containing the breakpoint(s). Once execution reaches a breakpoint, it will halt, awaiting input from you.



At this point, the line that is about to be executed is highlighted in a special color (light blue by default), the "resume", "step into", "step over", and "step out" buttons are enabled, and a special prompt appears in the Interactions Pane.

- **The Resume Button:** runs the program until another breakpoint is reached, or the interaction has ended.
- **The Step Into Button:** steps through the execution of the method invocations in the current line.
- **The Step Over Button:** executes the current line and stops execution on the next line.
- **The Step Out Button:** executes the rest of the current method and stops execution at the line from which the current method was called.

At the prompt, you can still type arbitrary Java expressions like usual, but you have additional power--all the fields, variables, methods, and classes than can be seen from the current point in the execution can be viewed and modified. This gives you incredible power--you can actually modify the values of your fields and variables while debugging. All you have to do is type in an assignment statement.

Chapter 5. The Project Facility

Overview

The project facility is intended to provide a lightweight approach to managing a large number of files spread between different subfolders and packages.

The project facility makes it easy to work with larger projects, and allows you to save open java files into a project file and then reopen the project at a later time. This means that you don't have to waste time reopening files in multiple folders. The project facility was designed with two goals in mind:

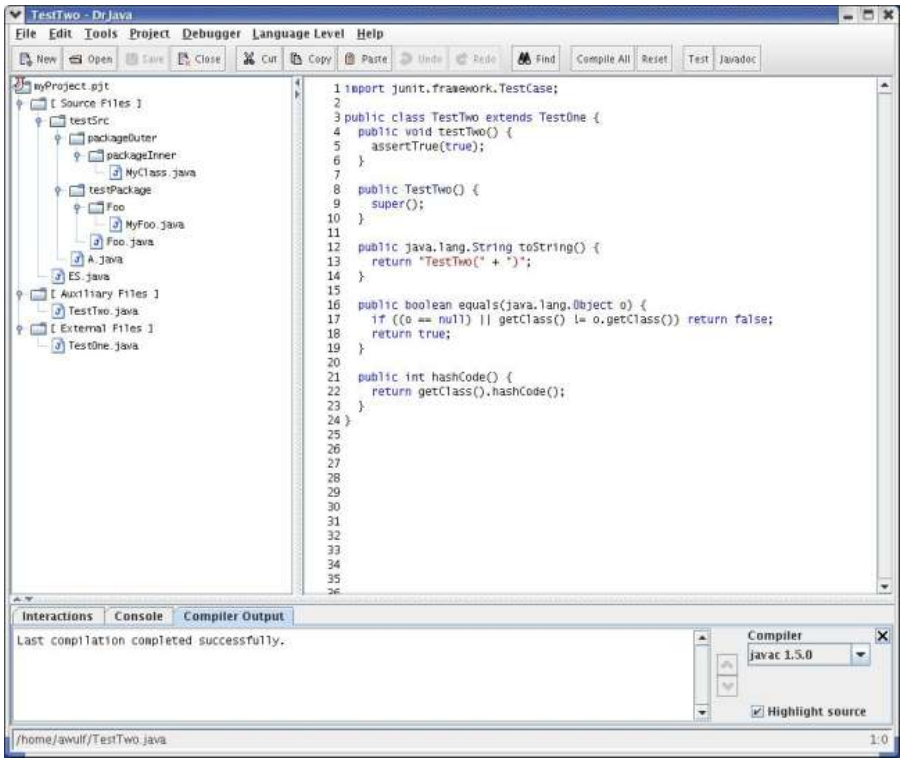
1. To allow sets of related files to be opened together.

2. To allow DrJava's tools and options to manage a set of related files as a single entity.

The fundamental difference between the DrJava project facility and the project facilities of most professional development environments is that DrJava does not distinguish between "Open" files and "Project" files--if a file is open and in the project source directory, it is part of the project.

Tree View

This project facility introduces a new view in DrJava called *Tree View*. When you are working with a project, the left hand pane displays the files in a tree view rather than the traditional list view. This means that you can see where your files are in relationship to each other. If you select a folder in the tree view, when you select "Open" or "Save", the default directory DrJava looks in will correspond to that folder.



There are three main categories of files in Tree View: Source, External, and Auxilary files.

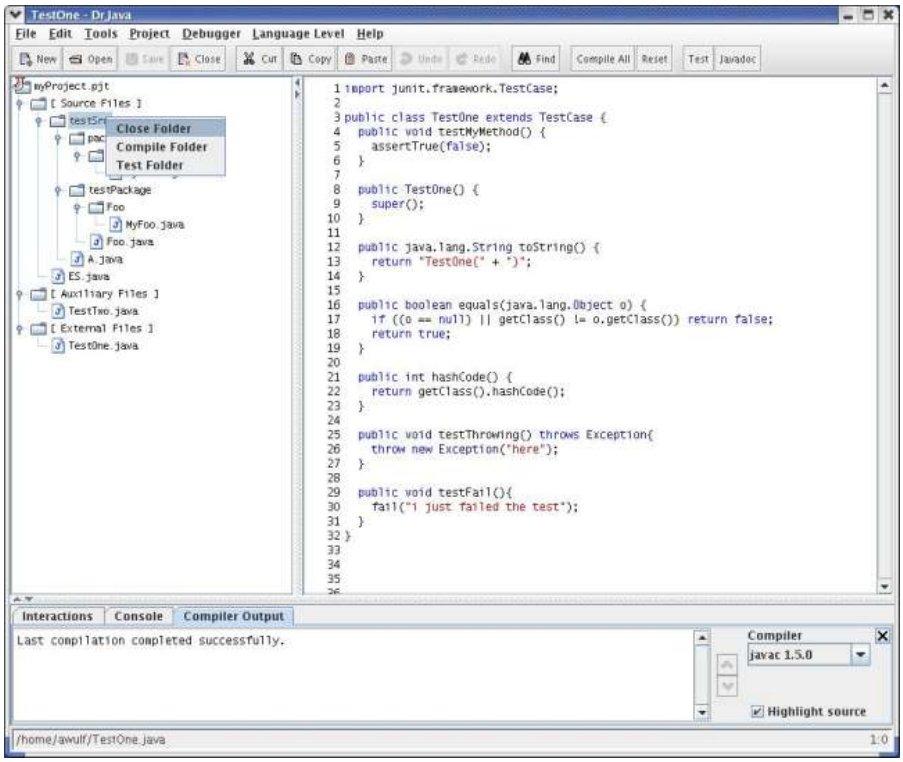
- **Source Files:** All files located in or below the project file's directory(project directory). Thus, the location of your project file determines what files are part of your project. It is important to put the project file in the root of your project hierarchy.

- **External Files:** Files that are not in the project directory. They will never be compiled or tested as part of your project. They are not saved in the project file.

- **Included External Files:** Files that are located outside of the project directory but that are still important to the project. They are compiled and tested with the project and are opened whenever the project is opened.

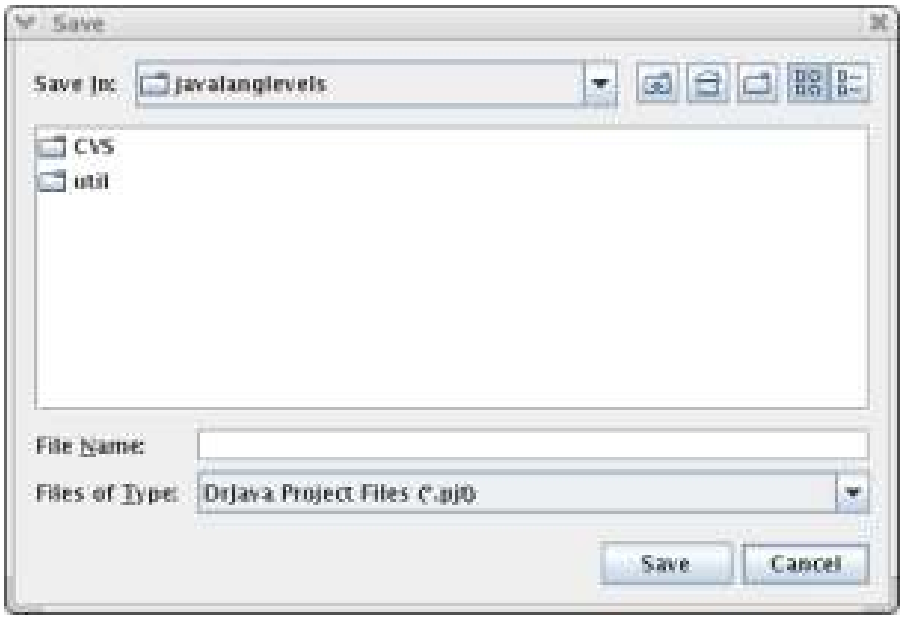
To move a file from the External Files branch to the Included External Files branch, right click on the file in the tree listing, and select the appropriate option at the bottom of the menu.

The behavior of some of the buttons also changes in Tree View. The "Compile All" button only compiles open project source and auxiliary files, instead of all open files. "Test All" will only tess open project source and auxiliary files. To compile or test the external files, you can right click on the folder and select "Compile Folder" or "Test Folder".

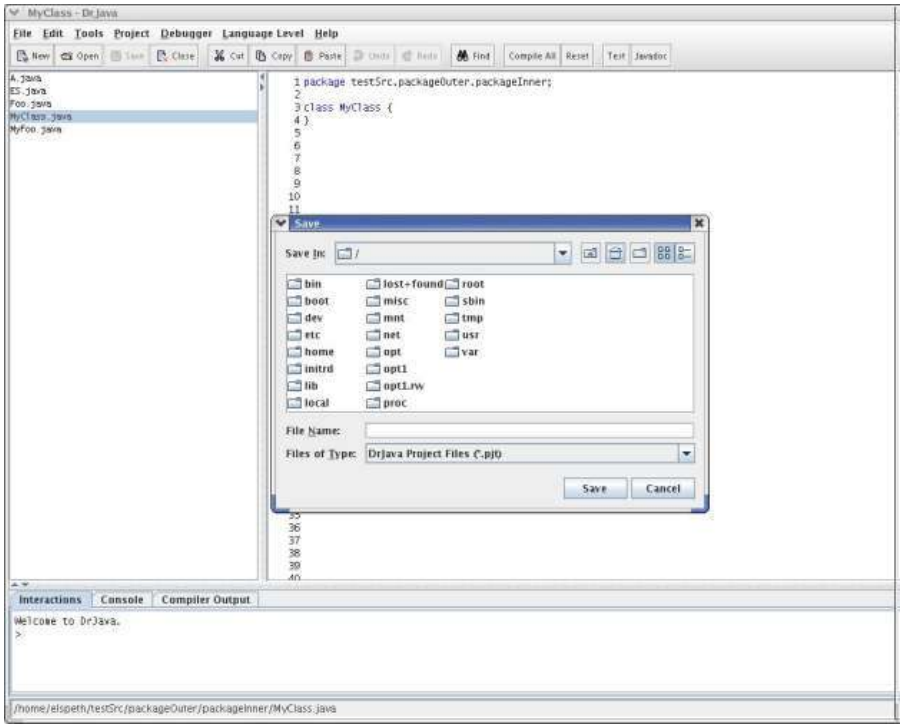


Creating and Saving a Project

When you want to create a new project, you have two options. The first is that you can chose "Project, New" from the top menu to create a brand new project. Once you have done this, a Save dialog box pops up. Specifiy where you want the project file to be saved, and you are ready to go. It is important to note that all project files must be in the directory or a subdirectory of where the project file is saved.



The other way to create a project is to open up the java files you want to include in the project, and then select "Project, Save As". This will create a new project consisting of those files.



To add new files to your project, just open them. They will automatically be sorted into the correct folder. When you save the project, they will be saved with it.

Saving Your Project

If you make any changes to the structure or state of your project (for instance, collapsing a folder tree or opening or closing files), you will want to save it so your preferences are maintained. To save a project,

select "Project, Save". Note that saving a project only saves the list of what files are in the project; it does not save the files themselves. Use "File, Save All" to save all the files in the project.

Saving a project will also save which document you are currently viewing so that it will be re-displayed the next time you open the project. It also saves the cursor location in each file as well as whether a folder in the project tree is collapsed or expanded.

Opening a Project

To open a project file, just select "Project, Open" and select the project file you wish to open.

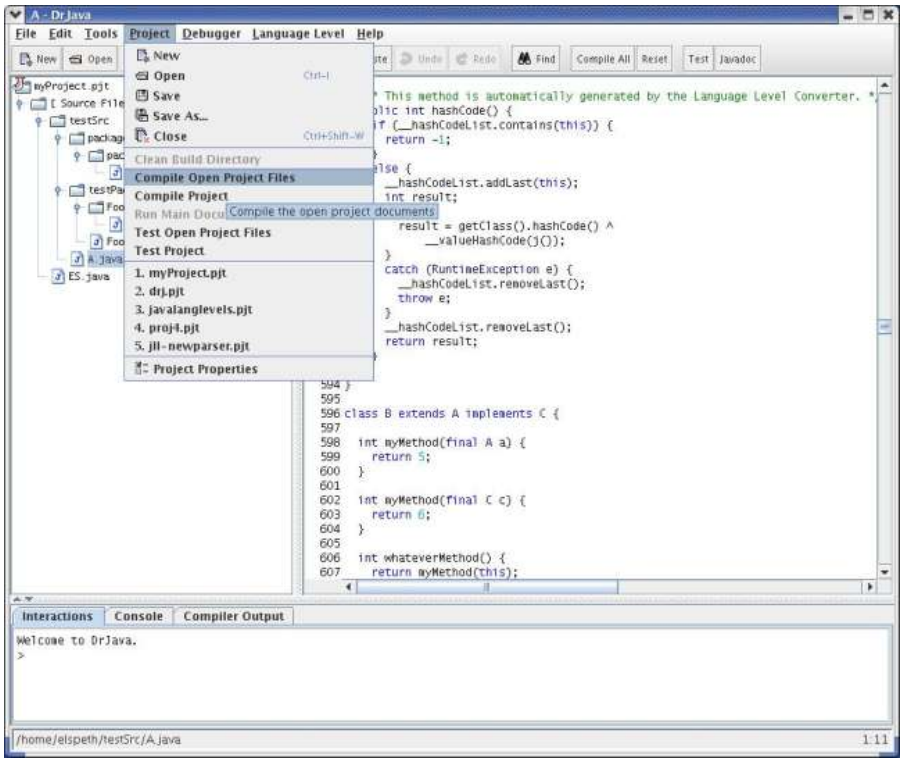
Compiling Your Project

When compiling a project, you must decide whether you want to compile only the open project files or all of the files in a project. Compiling all open files will only compile those files that are open in the project view (source files and auxiliary files). To do this, sellect "Project, Compile Open Project Files" or right click on the root of the project tree and select "Compile Open Project Files". You can also click on the "Compile All" button.

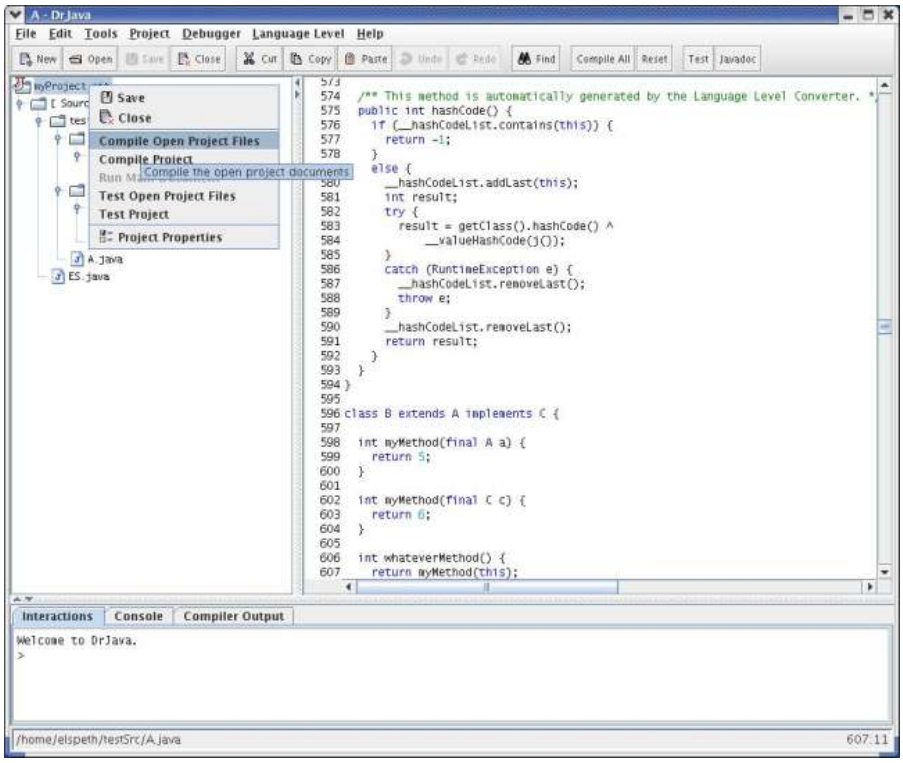
To compile all project files (source and auxiliary), even ones not currently open in DrJava, select "Project, Compile Project", or right click on the root of the project and click "Compile Project".

Note that all files in the External Files folder will not be compiled as part of the project--you must do this separately.

You can compile from the Project Menu:



Or from the Navigator Pane:

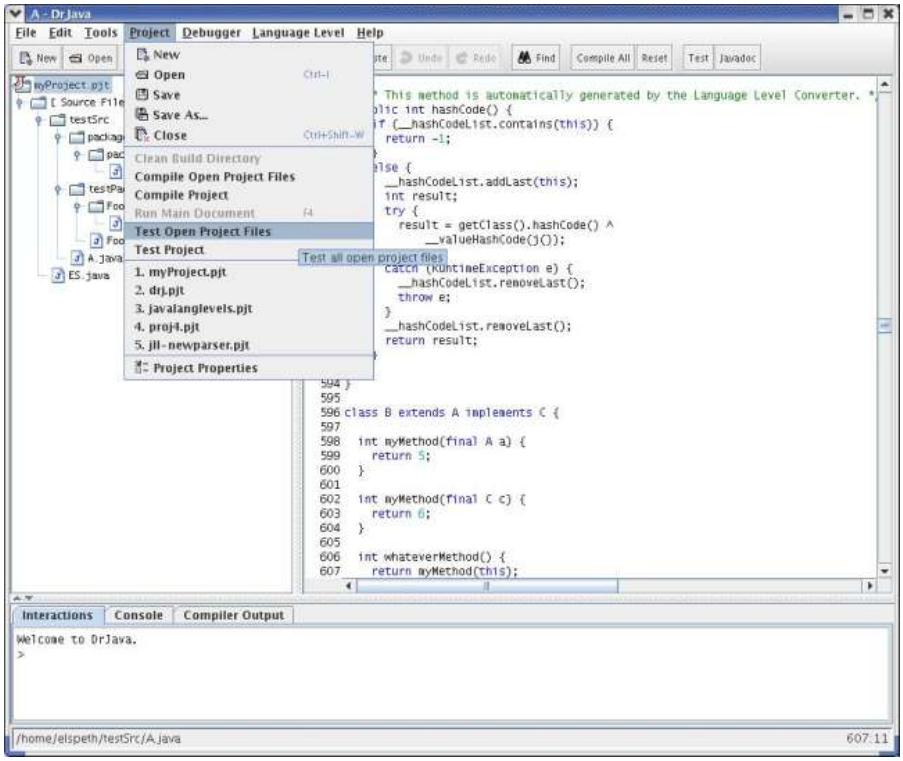


Testing Your Project

There are also two options for testing a project: test just the open project files or test the entire project. These have the same meaning that they do for compiling. To test only the open project files, select "Project, Test Open Project Files". This will test all JUnit test files currently open in the source files and auxiliary files branches.

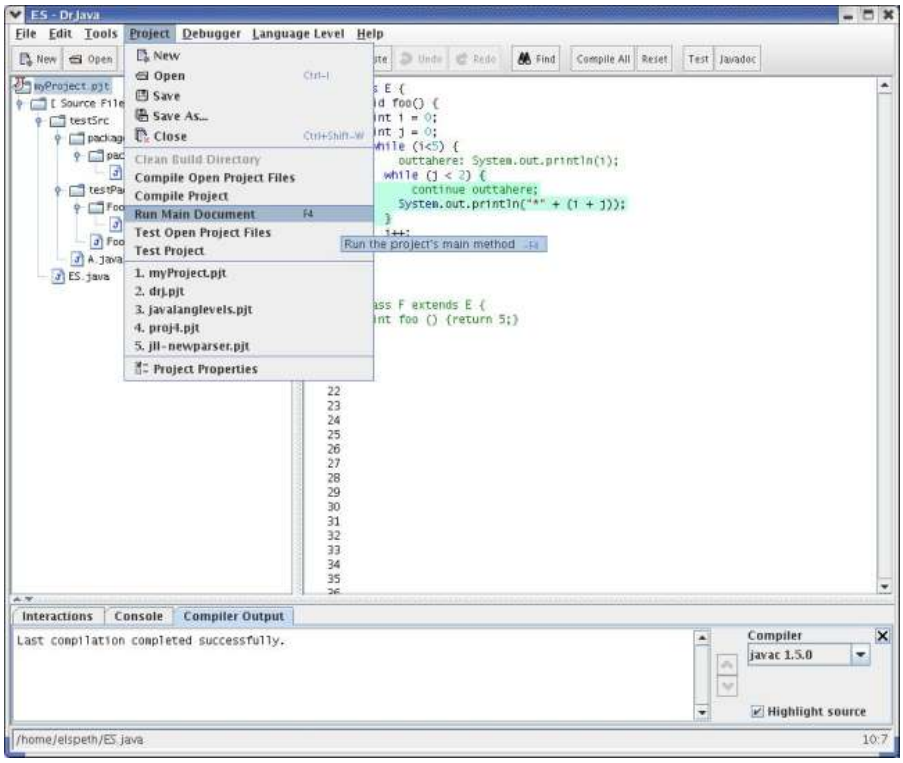
To test all project files, even those not open in DrJava, select "Project, TestProject". This will test all JUnit test files in the project directory and its subfolders as well as all of the auxiliary files.

External files will never be tested.



Running a Project

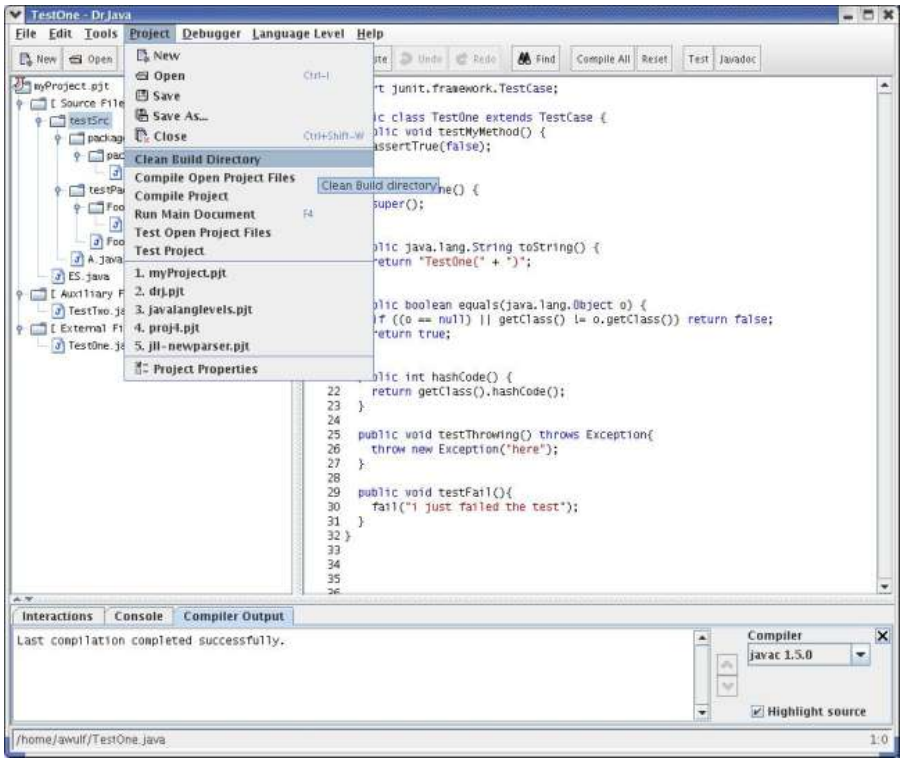
To run the main method of a project, select "Project, Run Main Class." This will load the class specified as the project's main class in Project Properties, and run its main method. If you have not specified the file to load in Project Properties, this option is grayed out



If you have configured a main class, then DrJava will also show a "Run Project" toolbar button. If you have not specified a class as main class, a "Run" button is shown instead that runs the main method of the currently shown document.

Clean Build Directory

This option is only enabled once you have specified a build directory in the Project Properties dialog box. Once you have specified a build directory, selecting this option will remove all `.class` files and empty directories in the build directory.



Project Properties

The Project Properties dialog box allows you to set two optional properties to better utilize the project facility. To access the Project Properties dialog, select "Project, Project Properties".

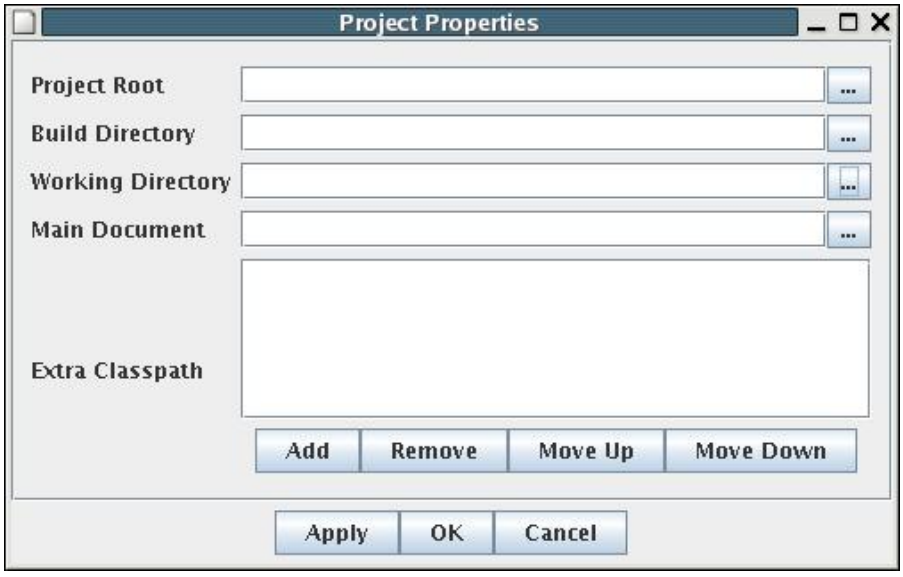
Project Root: This is the directory where the source files are located. It corresponds to the default package of your project. If no directory is entered, the directory containing the project file is assumed.

Build Directory: Set this if you want your compiled files to be compiled to a directory other than the source directory. This is often useful if your project is large or you want to be able to clean the build directory (see the option on the Project menu).

Working Directory: DrJava will use this directory as "current" directory for the Interactions pane and the user program.

Main Class: This allows you to specify the class that holds the main method for the project. When this is set, you can use "Project, Run Main Class" to launch the program.

Extra Classpath: In this section you can list directories or JAR files that should be included on the classpath while you are working with this project. The classes added this way are available both in the Interactions pane and when compiling.


Project Properties
Project Root
Build Directory
Working Directory
Main Document
Extra Classpath
...
Add
Remove
Move Up
Move Down
Apply
OK
Cancel

Chapter 6. The Preferences Menu

The Preferences menu in DrJava can be accessed by selecting "Edit, Preferences" on the main toolbar, or by the keyboard shortcut Ctrl-semi-colon. This menu gives you the ability to tweak your configuration of DrJava so that it better serves your needs. To switch between Preference Categories, click on the names in the left part of the Preferences window.

Resource Locations

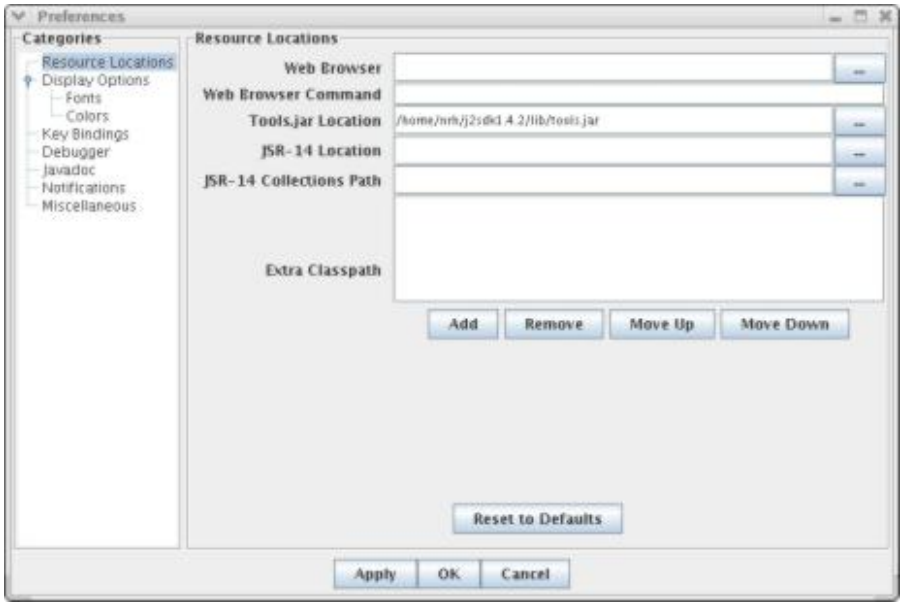
The Resource Locations menu allows you to specify the location of many important resources. All of these fields are optional and can be left blank if you want default values to take effect.

- **Web Browser and Web Browser Command:** Allow you to specify which web browser you want to use for viewing Javadoc and Help files. You can choose to specify the Web Browser program directly, state the command you use to launch your browser, or leave both blank if your system has a default browser.

- **Tools.jar Location:** The directory of your tools.jar file which has the compiler and debugger. This file is usually created during the installation of the Java JDK.

- **Display All Compiler Versions:** By default, DrJava only displays one compiler per major version, even if multiple updates are found (Example: You have JDK 6 Updates 10 and 14 installed; DrJava will only display JDK 6 Update 14). To display all compiler versions, mark this checkbox. Note: You have to restart DrJava when you change this setting.

- **Extra Classpath:** A way for you to specify extra directories you want the compiler to look in when it is trying to find class files. Use "Add" and "Remove" to control which directories are on the classpath, and "Move Up" and "Move Down" to order the directories in the order you want them to be looked through. If no directories are specified here, the compiler will look in the directories of the files you are compiling.



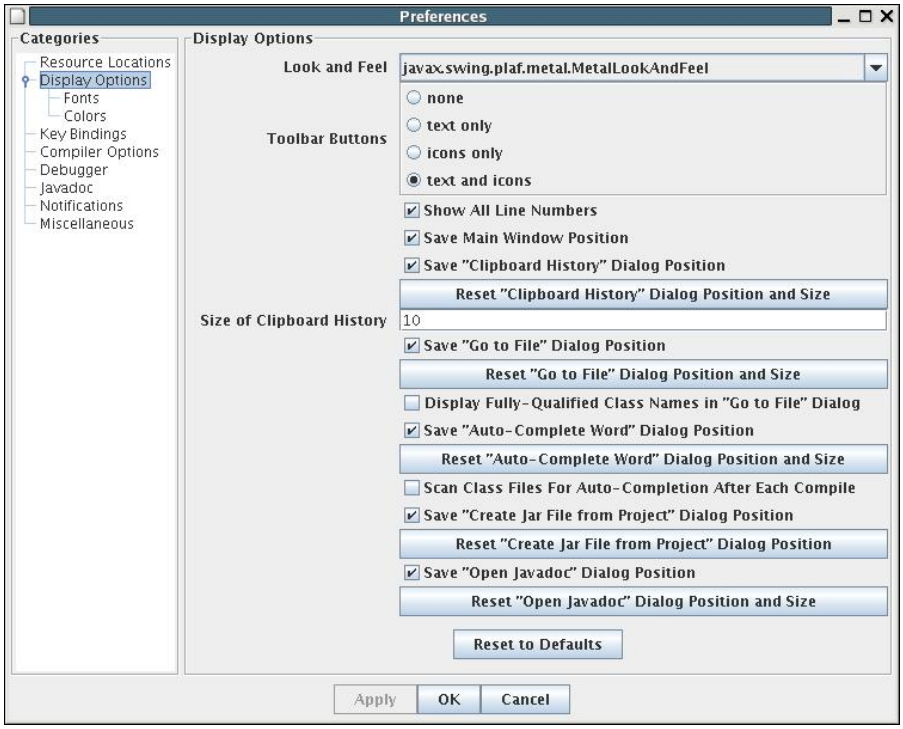
Older versions of DrJava also had the following options:

- **JSR-14 and JSR-14 Collections Path:** The JSR-14 jar is the location of the JSR14 compiler, while the JSR-14 Collections path is a collection of the generified collection classes, such as Vector<?>. If you are not using the JSR14 compiler, you can just leave these blank.

Display Options

The Display Options menu allows you to control how DrJava looks.

- **Look and Feel:** Specify what theme DrJava uses
- **Plastic Theme:** If Plastic is selected as Look and Feel, then the specific theme can be selected here
- **Toolbar Buttons:** Choose the configuration of text and graphics that you want on your toolbar.
- **Line Numbers:** Choose whether you want these shown on the left hand side of the definitions pane.
- **Show sample of source code when fast switching:** Choose whether a sample of the source code around the current caret position should be shown in the Fast Switch window.
- **Show Code Preview Popups:** Whether a sample of the source code around the document location should be shown in the Breakpoints, Bookmarks and Find Results panes.
- **Size of Clipboard History:** DrJava puts all text that you copy or cut out of text in a Ctrl+Shift+V, you can show the entries of the history and paste one of its items. This setting determines the size of the history.
- **Display Fully-Qualified Class Names in "Go to File" Dialog:** If this option is checked, the "Go to File" dialog displays both the simple and the fully-qualified class name (i.e. both MyClass and foo.bar.MyClass). This sometimes makes navigation easier, but it may increase the time it takes to display the "Go to File" dialog.
- **Scan Class Files For Auto-Completion After Each Compile:** DrJava can auto-complete the names of user classes. If this option is checked, DrJava will scan all class files that were created after each compile to obtain the names, even of inner classes. IF this option is not checked, DrJava can only auto-complete the names of the documents that are open. Selecting this option slows down compiles.
- **Consider Java API Classes for Auto Completion:** When this option is enabled, DrJava will include the names of the standard Java API classes in the list of names used for auto-completion.
- **Display Right Margin:** Enable this option to let DrJava display a vertical line representing the right margin of the document.
- **Right Margin Position:** This option controls the position of the right margin. By default, the right margin line is displayed after 120 columns, provided the "Display Right Margin" option above is enabled.



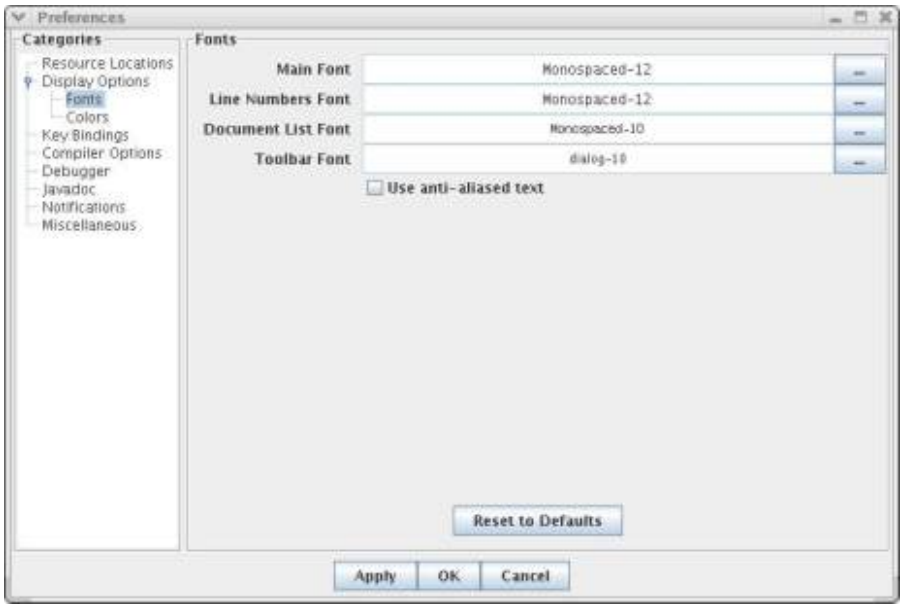
There are two submenus under Display Options, Fonts and Colors.

Fonts Options

The Fonts Options allows you to specify how the font should look. There are four font categories.

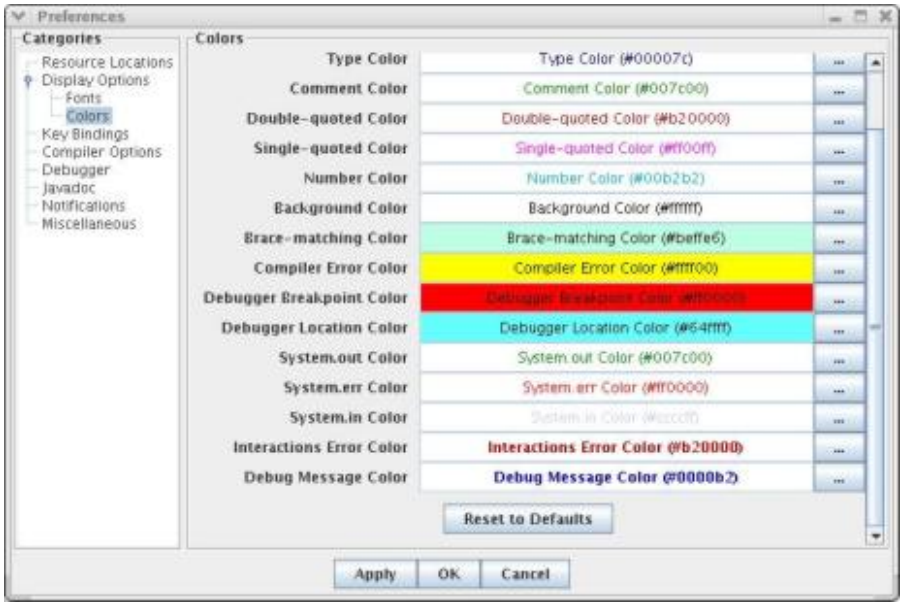
- **Main Font:** Specifies the font for the text in the definitions pane and the Interactions Pane.
- **Line Numbers Font:** Specifies the font for the line numbers in the definitions pane, if you enabled them.
- **Document List Font:** Specifies the font for the file names in the navigator pane on the left.
- **Toolbar Font:** Specifies the font used on the toolbar

You also have the option of using anti-aliased text. If this is selected, DrJava displays a smoothed version of the text. In some situations, this is more pleasant to view.



Color Options

The Colors Menu allows you to specify colors for a variety of types of text. Click on the "..." button to change the color for a type of text. The color is then demonstrated on the colors menu.



Window Positions

The Window Positions category allows you to control if DrJava saves the positions of its windows.

- **Save Window Size and Position:** Check this if you want the size and position of the main DrJava window to be saved on exit.
- **Save "xxx"Dialog Position:** Check this if you want the size and position of the "xxx" dialog to be saved on exit (different choices for "xxx").

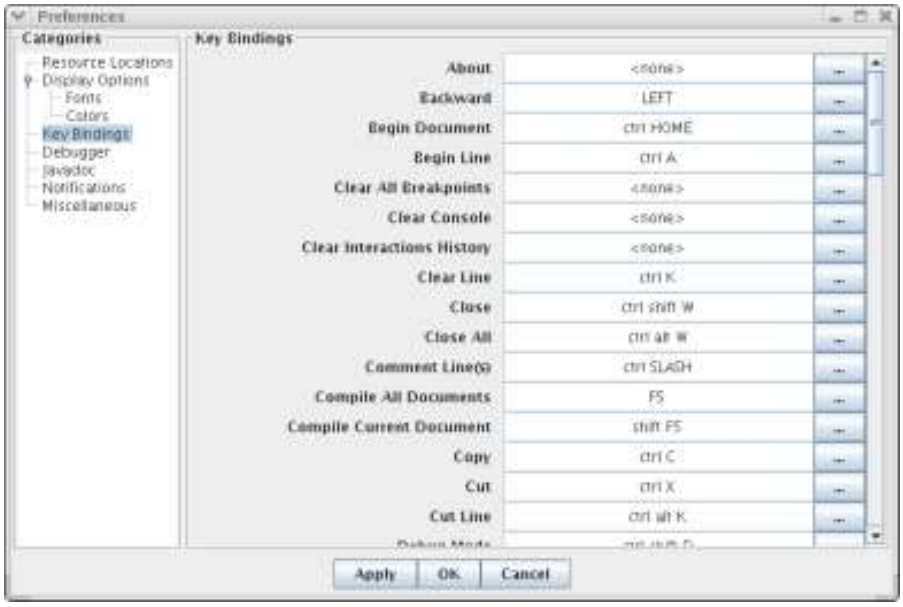
- **Reset "xxx" Dialog Position and Size:** Press this button to reset the size and position of the "xxx" dialog (different choices for "xxx"). This can be useful if the dialog was displayed outside the screen for some reason and is not accessible.

- **Detach Tabbed Panes:** By default, the tabbed panes are attached to the bottom of DrJava's main window. By selecting this option, DrJava displays the tabbed panes in their own separate window.

- **Detach Debugger:** By default, the debugger pane is displayed in DrJava's main window. By selecting this option, DrJava displays the debugger in its own separate window.

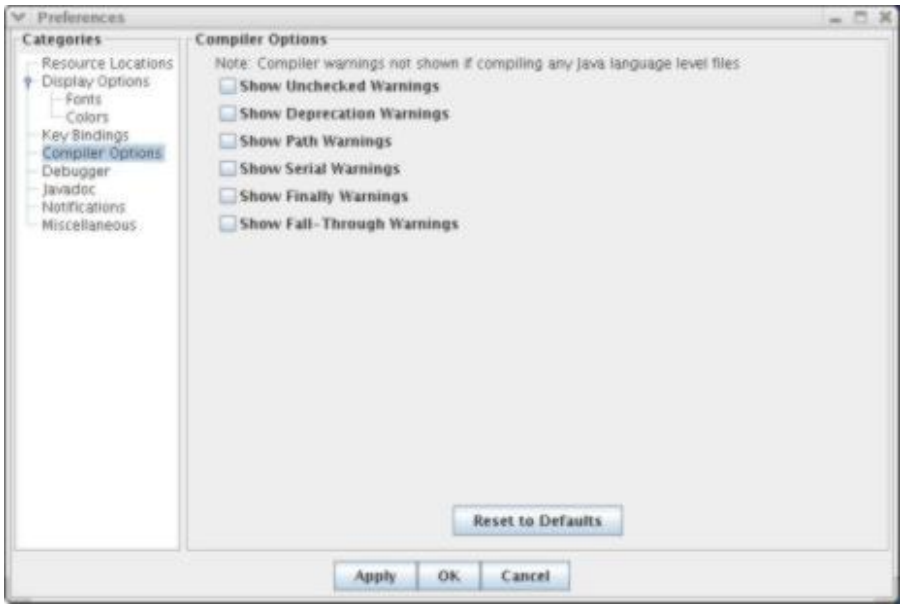
Key Bindings

The Key Bindings menu allows you to specify the keyboard shortcuts you use for everything from compiling a document to cutting text to resetting the Interactions Pane. The only restriction is that the shortcut you choose must be one basic entry--for example, you cannot use Ctrl-X, Ctrl-S to save a file, but you could use Ctrl-Alt-S. However, actions accept multiple key bindings so that a user may bind both Ctrl-X and Ctrl-S to the save file action and use either keystroke combination to save their file. Adding support for a wider range of keyboard shortcuts is a feature request currently under review.



Compiler Options

The Compiler Options menu allows you to set what warnings you will be shown in the compilation window on a normal compile. An explanation of each warning is given if you hover the mouse over it. We recommend that beginning programmers uncheck all of these boxes until they have a solid understanding of what the warnings mean.



Interactions Pane

The Interactions Pane category allows you to set options for the Interactions Pane.

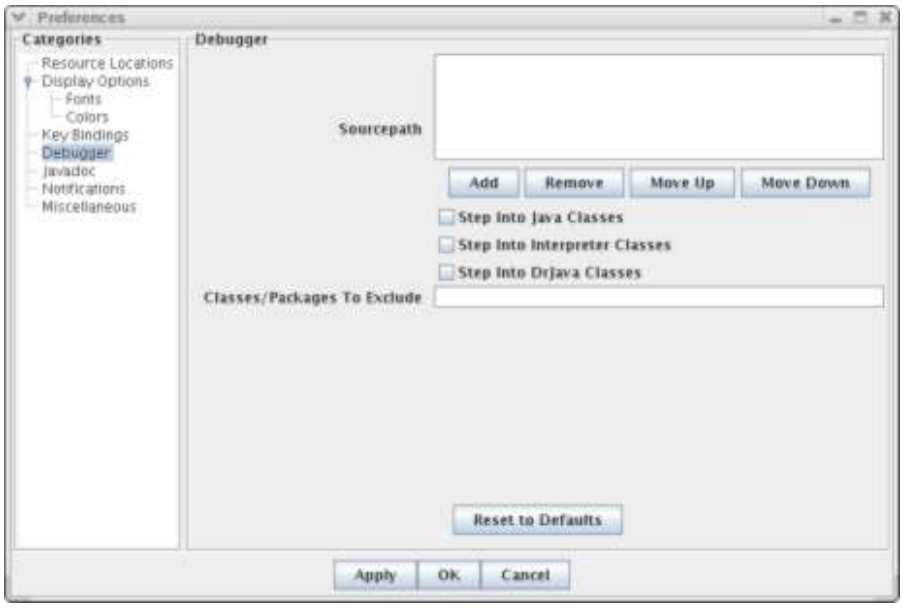
- **Size of Interactions History:** Specifies how many interactions will be stored in the history for you to scroll through using the up arrow.
- **Enable the "Auto Import" Dialog:** When this option is enabled, DrJava will display a dialog to automatically import classes when a class name is interpreted but not known. After the class has been selected, DrJava will execute the appropriate "import" statement and re-execute the line that caused the dialog to appear.
- **Classes to Auto-Import:** This option allows you to select classes and packages that should be imported automatically whenever the Interactions Pane is reset. List fully-qualified class names (e.g. java.util.ArrayList, or packages ending with a *, e.g. java.util.*).
- **Restore last working directory of the Interactions Pane on start up:** If this option is enabled, DrJava will restore the directory that was last used in the Interactions Pane. If it is disabled, DrJava will always use the value of the "user.home" property.
- **Smart Run Command:** If this option is enabled, DrJava will run applets and ACM Java Task Force Programs using the "Run" and "Run Project" buttons as well. These applets and ACM Java Task Force Programs do not need to have a main method to be run, as long as they are proper applets or ACM Java Task Force Programs.
- **Enforce access control:** This option controls the access control DrJava performs when class members are accessed. If the option is set to 'private and package only', then access control is used for all class members that are private or package private. If it is set to 'private only', then access control is used only for private members, and the other access levels can always be accessed. If it is set to 'disabled', all class members can be accessed, regardless of their access level. (Note: Currently, access control in DrJava's Interactions Pane has not been fully implemented; at most, access is checked for private and package private members; protected members can always be accessed.)
- **Require semicolon:** If this option is enabled, then DrJava will require a semicolon at the end of statements in the Interactions Pane.

- **Require variable type:** If this option is enabled, then DrJava will a variable type for variable declarations in the Interactions Pane (e.g. `int i = 5`). If it is disabled, DrJava will attempt to assign a variable type automatically (e.g. `i = 5` to declare an `int i`).

Debugger Options

The Debugger Menu gives you more control over the debugger.

- **The Sourcepath Box:** Allows you to specify any libraries that you want the debugger to look through. The order of libraries in the list specifies the order in which they are searched for matching .java files. Use "Add" and "Remove" to control what is on the Sourcepath, and "Move Up" and "Move Down" to control the order.
- **Stepped Into Checkboxes:** Allow you to specify what will be stepped into. If you tell the debugger to step into a file, it will open up the source and show you line by line what is being executed. You would only do this if you were curious about behavior that was expected to occur within the method being executed. Normally, you would not step into Java classes and Interpreter classes, because you assume their code is correct and are intstead interested in how your code interacts with them. You should never select Step Into DrJava classes unless you are debugging DrJava itself.
- **Classes/Packages to Exclude:** Allows you to specify other classes/packages to step over. This should be a list with fully qualified names. To exclude a whole package, add `packagename.*` to the list. You might use this box to exclude instructor provided libraries, for example `java.util.*`.
- **Auto-Import after Breakpoint/Step:** Automatically imports all classes and packages again that had been imported when the program was last suspended, i.e. before the breakpoint was hit or before the last step was taken.
- **Auto-Step Rate in ms:** The delay interval at which the automatic trace steps into each line of code in the program.



Javadoc Options

The Javadoc menu lets you specify everything you need to generate perfect Javadoc for your source files. For information on Javadoc in general, see the Javadoc section above.

- **Access Level:** Allows you to set the level of what will be shown in your documentation. Choose the most restrictive access you want displayed, and everything less restrictive will also be included.

- **Java Version for Javadoc Links and URL:** Set the version to whatever version of java your code relies on, and maek sure the corresponding URL is set appropriately. Because most Javadoc documentation references library files in the Java API, it is important to specify which version of the API to link to.

  This option also controls which URL is used to access the Javadoc pages for the Java API.

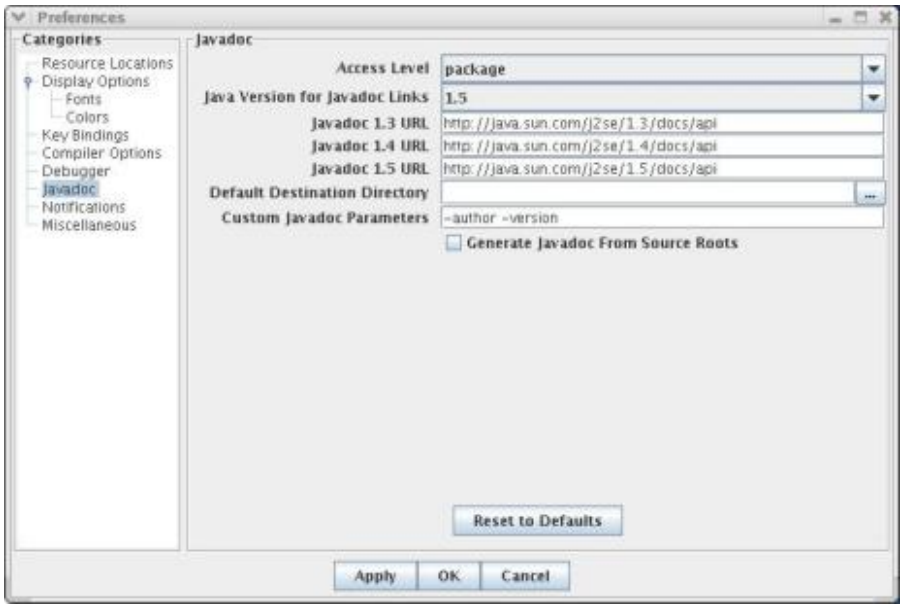
- **JUnit 3.8.2 URL.** The URL to use when generating links to JUnit 3.8.2 library classes or opening the Javadoc pages for the Java API.

- **Additional Javadoc URLs.** A list of URLs used to open Javadoc pages for user-specified libraries. Please enter the URL to the directory that contains the `allclasses-frame.html` file. For example, to open DrJava's Javadoc, enter `http://drjava.org/javadoc/drjava`.

- **Default Desination Directory:** Specifies where the generated documentation will be saved. Click on the "..." button to specify it.

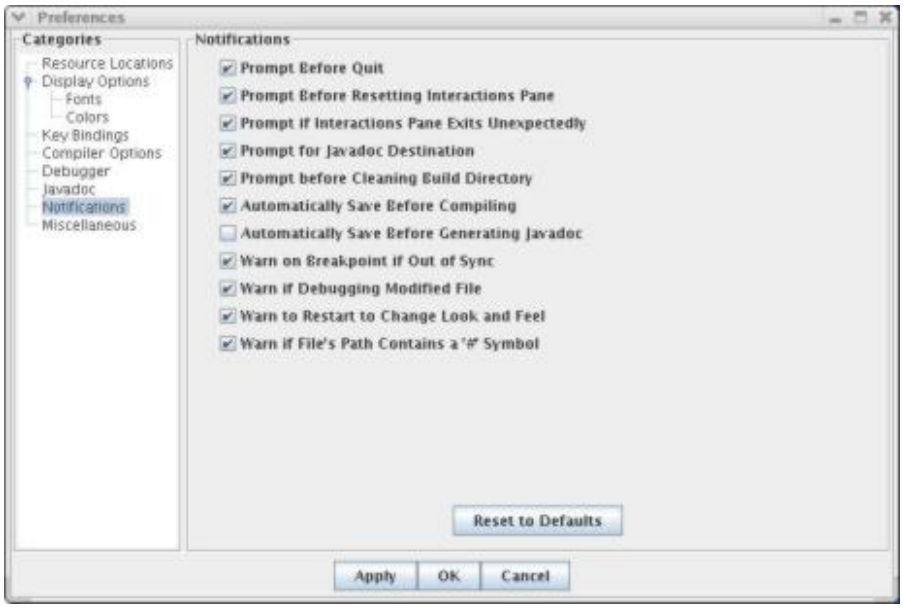
- **Custom Javadoc Parameters:** Allows the expert user to pass more complicated flags to Javadoc.

- **Generate Javadoc from Source Roots.** Controls whether all packages in a file's directory have Javadoc run over them, or whether just the immediate package and subpackage do.



Notifications Options

The Notifications category lets you say whether or not you want notifications on certain actions. This is the place to come if you accidentally turn off a warning and want to turn it back on.

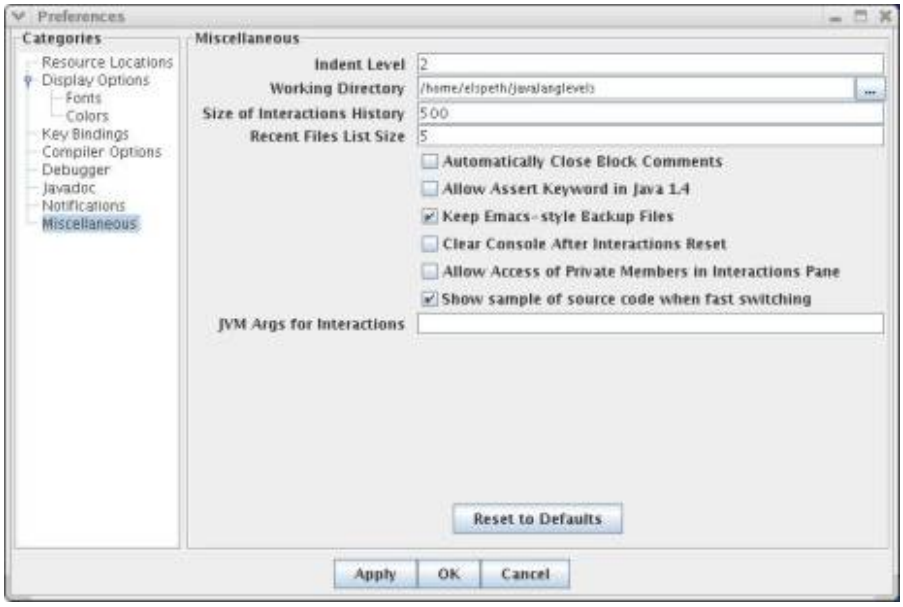


Miscellaneous Options

The Miscellaneous category contains all of the options that did not fit neatly into one of the above categories.

- **Indent Level:** Specifies the number of spaces that make up a "tab" in DrJava. This is used for auto-indenting.
- **Recent Files List Size:** Specifies how many of your most recently open files are displayed when you open the "File" menu on the toolbar.
- **Maximum Size of Browser History:** Specifies how many source code locations are stored in the browser history. This history can be used to go back and forth, like in a web browser.
- **The Checkboxes:** Specify a variety of options that are personal preference:
  - **Automatically Close Block Comments:** Puts a "*/" after every "/**" or "/*" you type, so you don't have to remember to close the comment.
  - **Allow Assert Keyword in Java 1.4:** Passes a flag to the 1.4 compiler that specifies the assert keyword is okay.
  - **Keep Emacs-style Backup Files:** Tells DrJava to automatically make backups of files such that file file.java would be backed up as file.java~. This is a nice option, because it provides an additional level of protection against lost code.
  - **Clear Console After Interactions Reset:** Makes the Interactions Pane clear when it is reset. After a reset, any previous objects built in the Interactions Pane have been lost and must be recreated. Clearing the console helps you remember that you need to recreate everything.
  - **Require test classes in projects to end in "Test":** Whether to require that JUnit test classes in projects end in "Test". If this is enabled, classes that do not end in "Test" will not be considered JUnit tests when in project mode.

- **Put the focus in the definitions pane after find/replace:** If this option is checked, the focus will be put in the definitions pane after using "Find". If this is not checked, then the focus will remain in the Find/Replace tab.
- **Forcefully Quit DrJava:** On some systems (namely tablet PCs), DrJava does not shut down properly when quit. Select this option to remedy this problem.
- **Enable Remote Control:** Java's "remote control" allows other applications to control certain aspects of DrJava, for example what file is displayed. This feature is also necessary if you want to double-click on a .java file to open the file in an existing instance of DrJava.
- **Remote Control Port:** Selects the port that Drjava uses for its remote control.
- **Follow File Delay:** Specifies the number of milliseconds that have to pass before DrJava will update a "Follow File" window again.
- **Maximum Lines in "Follow File" Window:** Specifies the number of of lines that a "Follow File" window may contain. If a file has more than this many lines, only the end of the file will be displayed.



File Types

Configurable options for file types. Note that the options here are only available on Windows, and only if the .exe file is used.

.drjava files are DrJava project files. .djapp files are DrJava add-ons. .java files are Java source files.

- **Associate .drjava and .djapp Files with DrJava.** Set the file type associations so that double-clicking on .drjava and .djapp files in the Windows Explorer will open them in DrJava.
- **Remove .drjava and .djapp File Associations.** Remove the association of .drjava and .djapp files with DrJava.
- **Associate .java Files with DrJava.** Set the file type associations so that double-clicking on .java files in the Windows Explorer will open them in DrJava.

- **Remove .java File Associations.** Remove the association of .java files with DrJava.
- **Automatically assign .java, .drjava and .djapp Files to DrJava.** Specifies whether Drjava should automatically associate .java, .drjava and .djapp files with DrJava so those files are opened with DrJava when double-clicked. When set to the default, "ask me at startup", DrJava will check at startup if those associations exist, and if not, present a dialog allowing the user to set the file associations. Setting this option to "always" will set the file associations automatically at startup without user interaction. Setting it to "never" will not change file associations nor ask the user.

JVM Options

The JVM category contains options for the two Java Virtual Machines (JVMs) that DrJava uses.

- **Maximum Heap Size for Main JVM in MB.** Specifies how many megabytes of memory Java should use for the Main JVM (the main part of DrJava including the editor and compiler). The "default" setting leaves this up to Java. If you experience "Out of memory" errors, set this to a value that is larger than 64 MB (default in Java 5) but smaller than the amount of physical memory you have. If it is still too small, choose the next bigger setting. Note: You have to restart DrJava for changes to become effective.
- **JVM Args for Main JVM.** Specifies the JVM arguments that should be used for DrJava's Main JVM, other than the maximum heap size (-Xmx), which is controlled using the option above. Note: You have to restart DrJava for changes to become effective.
- **Maximum Heap Size for Interactions JVM in MB.** Specifies how many megabytes of memory Java should use for the Interactions JVM (used to interpret code in the Interactions Pane and to run programs developed in DrJava). The "default" setting leaves this up to Java. If you experience "Out of memory" errors, set this to a value that is larger than 64 MB (default in Java 5) but smaller than the amount of physical memory you have. If it is still too small, choose the next bigger setting. Note: You have to reset the Interactions Pane for changes to become effective.
- **JVM Args for Interactions JVM.** Specifies the JVM arguments that should be used for DrJava's Interactions JVM, other than the maximum heap size (-Xmx), which is controlled using the option above. Note: You have to reset the Interactions Pane for changes to become effective.