The World of Eclipse and Its Debugger: An Introduction
An Event, a Sensation, that Took Place on Thursday, September 23, 2010 at 7:30 PM
Courtesy of the COS 109/126/217/226 Lab TAs, Who Are Awesome

Downloading and installing Eclipse
Download Eclipse from http://www.eclipse.org/downloads/. You will want the "Eclipse IDE for Java Developers" download. Eclipse comes as a .ZIP file, not an installer, so you will have to extract the files and set up desktop shortcuts yourself.

Unzip the file to Applications/ if you are an OS X user or C:\Program Files\ if you are a Windows user. Start Eclipse by running either Eclipse/Eclipse.app (Mac users) or Eclipse/eclipse.exe (Windows users).

Setting up your workspace
When you run Eclipse, it will prompt you for a workspace. Your workspace is just a folder to contain all of your projects for the semester. We recommend locations such as "C:\Users\you\COS226" (Windows) or "/Users/you/COS226" (OS X) or "/home/you/COS226" (*nix).

We recommend you use a different workspace for each of your courses.

Once you tell Eclipse the location of your workspace, it will load and then take you to a lovely Welcome page. Just close the tab that contains this page. You should now see a package explorer, a text editor, and many other panels. For now, close the Task List and the Outline panels to free up valuable screen space.

Your first project
Each Eclipse workspace can hold *projects*. Projects represent a collection of code and resource files that depend on each other. We recommend you create a different project for each of your assignments. To create a project, right click in the Package Explorer panel on the left side of your screen. Go to New -> Java Project. Enter a name for your project and select Finish. You're good to go!

At some point during the semester, you will need to use stdlib.jar and algs4.jar in your programs. Let's go over how to do that now. You should download these two files from http://www.cs.princeton.edu/algs4/algorithms/. Now go to your workspace preferences by clicking on Window -> Preferences (Windows) or Eclipse -> Preferences (Mac) and then Java -> Build Path -> User Libraries. Click on the New… button and enter a name like "COS 226 Library." Click OK and then click on the button Add JARs…. Navigate to where you downloaded the two lovely JARs and select them. Click "OK" until you are back in Eclipse proper. You now have a library for your 226 dependencies!

(You may notice that there are two different places to set preferences in Eclipse. The preferences you just changed were *workspace preferences*. They apply to all the projects in a workspace. There are also *project preferences* that you can access by right-clicking the Package Explorer. These preferences only apply to an individual project. Since we want external

libraries like stdlib.jar and algs4.jar to be available to all projects, we set them in workspace preferences.)

Now you have to add this library to your project. Right click on your project and go to Properties -> Java Build Path. Navigate to the Libraries tab and click the button Add Library. Select User Library and then check the library you just created. Click on OK until you are back in Eclipse proper.

You also want to create new Java classes where you can program and do your assignments. How do you do this? Right click on the project that you created earlier, go to New…, and then select Java class. Enter a name for the class and Finish. You just created a Java class!

Compiling

How do you compile (or "build") in Eclipse? You don't! – Eclipse does it for you. As you type or when you save, Eclipse automatically compiles your code.. Eclipse uses the compilation to jot down any warnings or errors it sees.

How do you see your compile warnings and errors? Two ways: Eclipse highlights them as you type. Also there is a tab called Errors that gives you a complete list of all of your errors.

You can find your compiled ".class" binary files in the folder

<your workspace location>/<your project name>/bin

For example, "C:\Users\you\COS226\percolation\bin" on Windows or "/Users/you/COS226/percolation/bin" on OS X.

Redirecting standard in and standard out

On COS 226 assignments, you will be asked to read in files from standard in or output files to standard out, usually with terminal commands like "java YourProgram < input.txt" or "java YourProgram > output.txt". The "<" and ">" signs indicate *standard in redirection* and *standard out redirection* respectively. Because Eclipse does not run your program in a terminal, you will need to find another way to redirect these two standard streams.

To redirect standard in, insert the following line of code as the first line in your "main" function:

```
System.setIn(new FileInputStream(filename));
```

"filename" is a String that has the path of the file from which you want to redirect standard in. But absolutely make sure that you take this line out of your code before you submit it!

To redirect standard out, go to the Run Configurations menu by clicking on the arrow next to the green run button on the top. Go to the tab Common, in the section Standard Input and Standard Output, check the box File and enter the location of the file to which you want to direct your output.

You can also always go to your terminal or command line and run your program as usual doing that. See above for where to find your .class files. You can "cd" to that directory and run "java YourProgram" as usual:

```
cd C:\Users\your\COS226\percolation\bin
java PercolationVisualizer < ..\data\input-25.txt
```

Remember, Eclipse periodically compiles in the background so you should remember to save your ".java" files before doing this in order to force Eclipse to compile a new version of your code.

Command-line arguments
Go to Run Configurations and go to the Arguments tab. In the box labeled "Program Arguments" enter your desired arguments separated by spaces.

Debugging using Eclipse
Eclipse can help you debug by letting you walk through your program. How do you begin? Open the debugging perspective by clicking on the icon in the top right corner with a plus sign on it and then selecting Debug.

The power of debugging comes from *breakpoints*. A breakpoint on a line in your code is an indication to the debugger that you want to stop and inspect a program before that line of code executes. From there, you can walk through your program, rewind it, change variables, and much more. You can create a breakpoint by double clicking on the left margin of your desired line. You know you created a breakpoint if a cute little blue circle shows up where you clicked. You can delete a breakpoint by double clicking on that cute little blue circle.

To begin your debugging fun, click on the bug icon or just press F11. Your program will pause at the line(s) at which you put a breakpoint. The eclipse debugger indicates which line of code it is *about to* execute (but not yet executed) by highlighting the line in a dark green color and putting an arrow next to the line.

Since your program has paused, you can view all of your variables and their current values. Do so by going to the Variables tab. You can also opt to watch variables by right clicking on a variable and clicking Watch. This will put the variable in the Expressions tab, which will allow you to conveniently collect the variables that are of interest to you.

You can also modify your variables during runtime! Go to the Variables tab and right click on a variable name. Select Edit Value and enter the new value that you want your variable to have.

You now will want to learn how to "un-pause" and proceed to more lines of code. How do you do this? There are buttons in the tab Debug that will make it happen for you. The ones of interest to you are the ones with the arrows on them. You can mouse over the button to see what each button does. Also, you can use keyboard shortcuts.

There are two basic movements that will be very useful for you:

- Step into (F5): if the current line is a function call, you will move to the first line of code inside that function
- Step over (F6): you will always move to the next logical line of code and not step into any functions

You can also tell you program to just keep executing as normal and not pause until it either ends or hits a breakpoint. This is called *resuming*. To do this, either click the green arrow button in the Debug tab or press F8. If you have taken COS 217 or used gdb, you may know these debugging movements respectively as "step", "next", and "continue".

If you are currently debugging inside of a function, there are two fun movements that could make your life easier:

- Step out of (F7): Continue running the program until you return from the current function. So it's like you're "stepping out of" the function that you're currently in.
- Drop to frame: This allows you to "rewind" your program! If you press this button, you will "rewind" to the beginning of the function that you are currently in, resetting any variables that may have changed.

One more thing to quickly mention – with Eclipse you can edit your code while you are actually debugging. If you save your edited code while debugging, Eclipse will automatically drop to frame and then let you debug the new code that you just wrote.

Using spaces instead of tabs
Go to workspace preferences by navigating to Window -> Preferences (Windows) or Eclipse -> Preferences (OS X). Go to General -> Text editors and check the box for Insert spaces for tabs. Then go to Java -> Code Style -> Formatter. Create a new Formatter policy based on the default by clicking on "New" and then "OK". Click on "Edit" to edit the new policy. Go to the Indentation tab, click the Edit button and make sure that the Tab Policy is Spaces Only. Okay out of everything!

Automatically formatting your code
Press Control + Shift + F (Windows) or Cmd + Shift + F (OS X).

Configuring the automatic formatter
Go to workspace preferences. Go to Java -> Code Style -> Formatter and have fun! There's a lot of stuff you can change.

Renaming a variable in your code
Right click on the variable and go to Refactor -> Rename… Then enter the new name, press enter, and Eclipse will automatically change every instance of the old name to the new name. As if by magic! *Dark magic.*

Showing line numbers
Right click on the blue margin to the left of your code and click Show Line Numbers.

Changing your font
Go to workspace preferences. General -> Appearance -> Colors and Fonts. Select Java and select "Java Editor Text Font" and click Edit.

Changing/Viewing Eclipse's Keyboard Shortcuts (which you should *really* use by the way)
Go to workspace preferences. General -> Keys. There are a lot of things you can change/view, enjoy!

Installing a new plugin
Go to Help -> Install New Software. Find the URL of your plugin (if you Bing/Google your plugin, its URL should be easy to find). Enter the URL into the box next to "Work with:" and then click "Add…". Enter a name and select OK! Your desired plugin should show up in the table below. Select which version you want and click Next through all of the dialogs. It will then start installing and you will live happily ever after with your new plugin.

FindBugs
This is an amazingly useful Eclipse plugin that finds common bugs in your code. But beware: it is no replacement for careful planning and debugging and will not find all of the bugs in your code. Your preceptors use FindBugs to find common mistakes in the code you submit, so running FindBugs is an easy way to not lose some points. The URL for the FindBugs plugin is http://findbugs.cs.umd.edu/eclipse.

Once this installs, you need to configure it. Download the official COS 226 FindBugs configuration file.[1] Now go to workspace preferences and then Java -> FindBugs. Go to the "Filter files" tab; next to the Exclude filter files box, click on the Add… button (this should be the middle Add… button). Select the file you just downloaded, press OK, and now you have FindBugs configured!

To run FindBugs on your code, right click on your project in the Package Explorer, and click on Find Bugs -> Find Bugs. There are two ways to see the bugs that were found. Firstly, each line that has a bug will have a tiny adorable bug icon in the left margin next to it. Secondly, if you click on the button with a plus sign on it in the top right corner and then select FindBugs, you will get a perspective called the "Bug Explorer" that will list all of your bugs.

Checkstyle
This is the same program that checks your style when you submit code to preceptors. With the plugin, you can install it as an Eclipse plugin and have it run as you program! The URL for Checkstyle is http://eclipse-cs.sf.net/update/. Make sure that expand the arrow for Checkstyle and ***only install version 4.4.3***. If you install version 5.1.1.* you will run into lots of problems! So only check version **4.4.3**!!!

Once it installs, you will need to configure it. Download the official COS 226 configuration file.[2] Now go to workspace preferences and click on Checkstyle. Click on the New… button. Make sure the Type is External Configuration File. Enter a name for your configuration (like

[1] http://www.cs.princeton.edu/courses/archive/spr08/cos226/checklist/findbugs-cos226.xml
[2] http://www.cs.princeton.edu/courses/archive/spr08/cos226/checklist/checkstyle-cos226.xml

"COS 226") and then find the location of the file you just downloaded. No description is necessary, just click OK. Now click on the new configuration that you just made and click the button "Set as Default". Click OK!

To run Checkstyle, right click on your project and go to either Checkstyle -> Check Code with Checkstyle or Checkstyle -> Active Checkstyle. (If you activate it, it will run automatically when you save as opposed to running when you tell it to.) You can see the style errors that Checkstyle finds by looking in the margins of your code. There will be an error symbol for all lines that Checkstyle has an issue with.

Thanks for reading this tutorial! Hope it helped!
And you can always find the wiki for Eclipse with tons more help and instructions on what you can do here:
http://help.eclipse.org