Eclipse Notes

Sesh Venugopal

Using Eclipse to write Java programs

What is Eclipse?

Eclipse is an Integrated Development Environment (IDE) widely used by Java programmers. It helps with all the logistical and administrative tasks that go into developing software, therefore leaving you, the programmer, free to focus on the design of your program's logic, i.e. the data structures and algorithms. For instance, it compiles as you type--if there is a synatx error in the line you just typed, a red X shows up next to that line, along with an error message if you mouse over the X. So you can immediately fix the error before you move on. If you type an entire program and there are no red X's anywhere, your program is already compiled and you can run it.

I find that my programming time is much more effective and productive when I use Eclipse. Most of all, it is a lot more fun to program in Eclipse than to use jEdit or any other smart editor. Because Eclipse is widely used, there's a lot of third-party support in the form of "plugins". As you program more and more, you will start using these plugins for your work...

Before you install Eclipse, you want to install Java on your computer. (This also allows you to write Java programs outside of Eclipse if you choose to do so.)

Downloading and Installing Java

Install Java 1.8 on your Windows computer

> Go to the Java downloads page, and click the Java Platform (JDK) 8u101/8u102 download box (left box). This will bring up a page, Java SE Development Kit 8 Downloads, with a list of downloadables for various platforms. Pick the Windowsx86 download, or, if you have a 64-bit machine, the Windows x64 download. Launching the exe file after you have downloaded it will install Java in your folder of choice. (The default installation will put the JDK under c:\Program Files.)
>
> If you already have an earlier update version of Java, as long as it's Java 1.5 or later, that's fine, you don't NEED to install the latest version.

Installing Java 1.8 on your Mac

> Macs comes with Java pre-installed, at least version 1.6. You can find it the directory

/System/Library/Frameworks/JavaVM.framework/Versions/Current

Typing

```
> java -version
```

in a shell will tell you what version it is. If you have version 1.5 or later, it will do for this course. But if you want to get the latest version (1.8), you can go to the same downloads page as described above for the Windows install, and download/install the Mac OS X version.

Downloading and Installing Eclipse

You can download and install Eclipse on your Windows, Mac, or Linux machine. Go to http://www.eclipse.org/downloads/ and download the **Eclipse Installer** for your Windows, Mac, or Linux machine, then follow the instructions to install Eclipse. (This is the latest release of Eclipse, code named **Neon**. Make sure you select the correct version - 32-bit or 64-bit - for your machine.)

At installation time, when asked, choose the **Eclipse IDE for Java Developers** package.

There are plenty of tutorials which are available from right inside the Eclipse program to help you write and run a Java program, but some of the basic tasks you will do for your programs are described in this document.

Views and Perspectives

To get started with Eclipse, a quick introduction to *views* and *perspectives* would be useful.

- In Eclipse, go to Help -> Help Contents -> Workbench User Guide -> Getting Started -> Basic Tutorial -> Editors and Views -> Views and read the entire section to learn about views.
- Go to Help -> Help Contents -> Workbench User Guide -> Getting Started -> Basic Tutorial -> Perspectives. Read the main Perspectives section, and the New Perspectives subsection.

Workspaces

When you start Eclipse, it will ask you for a workspace, which is a folder on your machine under which all your work will be stored. You may want to create a separate workspace folder for each class. See The Eclipse Workspace section below for more details.

Writing a Java Program in Eclipse

To write a Java program in Eclipse you will need to do the following:

- Set up a new project
- Develop one or more classes in the project

Set up a new project

When you want to build a Java application, you need to set up a new Java project. Different applications will be in different projects.

Suppose you want to build a new Java application called Maze. Here's how you would set up a project for this application:

- If you are not already in the Java perspective for some reason (the Java perspective is the default), put yourself in it by doing Window -> Open Perspective -> Other -> Java (default)
- In the package explorer view area (white space) right click, then New -> Java Project (or you can do File -> New -> Java Project)

- In the New Java Project dialog that pops up:
  - Enter a name for the project (Maze, in this example)
  - In the **Contents** section, select **Create new project in workspace**
  - In the **JRE** section, select the radio button for **Use default JRE (Currently 'jre6')**
  - In the **Project layout** section, select the radio button for **Create separate folders for sources and class files**
  - In the **Working sets** section, leave the box unchecked
  - Click **Finish**

  In the package explorer view, you should see a folder with the project name (Maze, in this example). Directly under it, you should see a folder called **src** - this folder will contain all the Java classes you write for this project.

Develop one or more classes in the project

Once you have a project set up (e.g. Maze), you can start building your Java classes.

Every class you build must be in some package. Eclipse will allow you to build classes that are not in any named package, but you will see a warning that this is not good. So, as a matter of good programming practice, always create a package (or packages) in which you can build your classes.

That being said, introductory programming courses may steer clear of packages for simplicity's sake, in which case all program files will go directy under the project's **src** folder.

For the sample Maze project, assume that you want to build a class called Amazing in a package called route. (Packages are conventionally named in lower case, classes alway have a capitalized first letter.) Here are the steps to start building your class:

- Right click on **src**, then New -> Package (or, click on the New Package wizard icon - yellow square - in the task bar)
- In the **Java Package** dialog that pops up, enter the name of the package (e.g. route), then click **Finish**

  In the package explorer view, you should see a folder under **src** with the name of the package (e.g. **route**)
- Right click on the package folder (e.g. route), then New -> Class (or, select the package folder in the Package Explorer, then click on the New Class wizard - green circle with a C - in the task bar)
- In the **Java Class** dialog that pops up, enter the name of the class (e.g. Amazing), leave the rest of the fields as they are, and click **Finish**

  In the package explorer view, you should see the java file for the class under the package folder. Also, Eclipse will switch focus to the new class file in the editor. You should see a small starting template of the code in the editor, including a package line at the top, and the class outline below. You are all set to build your class.

Using a Different Version of Java

If you have several versions of Java installed on your computer, and you want to pick a specific version of Java for a project, here's what you need to do:

- Go to Windows -> Preferences (on the Mac, it's Eclipse -> Preferences)
- In the **Preferences** dialog that pops up, expand the **Java** item in the left frame
- Click on the **Compiler** item (do not expand it)

- In the **Compiler** frame on the right, check if the **Compiler compliance level** is set to 1.8. If not, select 1.8 from the drop down list. (If your Eclipse version doesn't support 1.8, then use 1.7 or 1.6 or 1.5.)

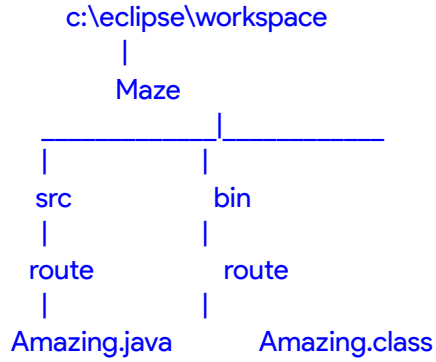
- In the left frame, click on the **Installed JREs** item (do not expand it)
  - In the **Installed JREs** frame on the right, if you see a listing for jre8, which is checked, you are done. Otherwise, click on the **Add...** button to the right of the list. Then select **Standard VM**, then **Next**, then **Directory...** to select the home directory of JRE 1.8 (it should be directly under the folder where you installed the JDK 1.8), then **Finish**.
    On the Mac, it should be in /System/Library/Frameworks/JavaVM.framework/Versions/1.6/Home/
  - In the list of installed JREs, you should now see the JRE 6 you picked above. Make sure the checkbox next to it is checked, then click **OK**.

  For more details, refer to Help -> Help Contents -> Java development user Guide -> Getting Started -> Basic Tutorial -> Preparing Eclipse, and read the section Verifying JRE installation and classpath variables.

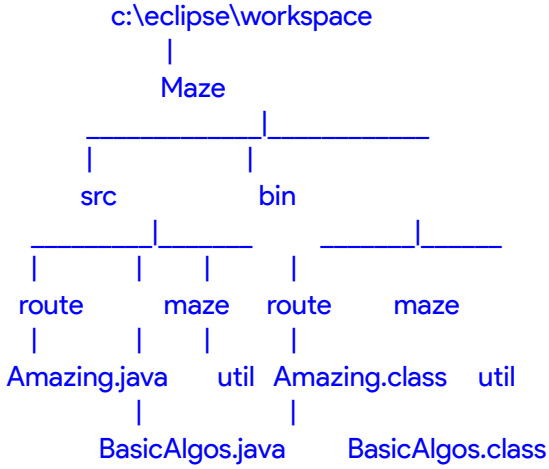
The Eclipse Workspace

Every time you run Eclipse, it stores all your work in a workspace on your computer. This is a folder you can specify when you start up Eclipse. Say you specify the folder **c:\eclipse\workspace** as the workspace.

The projects, packages and classes you build are arranged in a hierarchy under the workspace folder. The hierarchy for the example discussed above would look like this:



Say you add another package called maze.util, and under it a class called BasicAlgos. Then the workspace would look like this:



Notice how the package name maze.util gets converted into a hierarchy of folder maze with a folder util under it: for every dot in the package name, a new level is created in the workspace folder hierarchy.

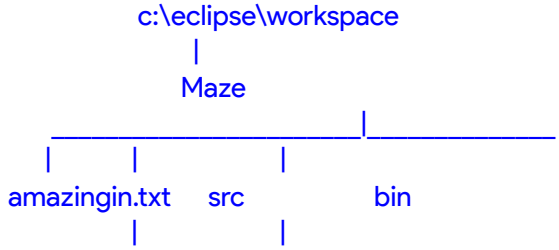
Running a Java Application

Say Amazing is an application class that has a main method.

The very first time you run an application you need to do it the long way. In the package explorer view, right click on the file that has the main method (e.g. Amazing.java), then select menu item **Run As**, and click on **Java Application**. Eclipse will open a console view (below the editor, by default) for terminal-based input and output, and run the program.

Subsequently, you can run the program by pulling down the green arrow drop down list in the task bar at the top, and selecting your application by name (e.g. Amazing)

If you are going to ask for input files in the program, these files must be placed directly below the project folder. So, for example, if the Amazing program takes input from a file called amazingin.txt, then the file is placed like this, directly under the Maxe folder:



Setting Up Program Arguments

If you run a program that takes arguments, you need to set up the arguments for in the program properties.

Say that Amazing needs an argument which is the name of an input file from which some data will be read. (So if you were to run this program outside Eclipse, on the command line, you may type > java Amazing infile.dat)

This is how you would set up the argument in Eclipse:

- In the package explorer view, right click on the file that has the main method (e.g. Amazing.java), then select menu item **Run As**, and click on **Java Application**. The program will run, but will give you an error since no argument was passed (assuming that the program is indeed accessing the argument). This step is just to "prime" Eclipse so that it sets up a run configuration for this program - see the next steps.
- Now repeat the above by selecting **Run As**, but this time click on **Run Configurations...**
- In the **Run Configurations** dialog that pops up, you should see an entry with the name of the main class (e.g. Amazing) in the left frame under **Java Application** - click on this if it is not already selected.
- In the right frame, you should see the configuration set up for the program (e.g. Amazing), showing the settings under the **Main** tab. Click on the **Arguments** tab.
- In the text area titled **Program arguments** immediately below the tab bar, enter the name(s) of the argument(s), one per line (e.g. input file infile.dat), and click the **Apply** button (toward the bottom).

IMPORTANT: If one or more of your program arguments are file names, you need to place the files **directly under the project folder** in Eclipse. So, if your file is stored on somewhere on your computer, you should drag and drop it into the project folder. (In the example above, you would need to drag and drop the file infile.dat into the folder Maze.)

You can now run the program using the green arrow shortcut as described in the previous section.

Zipping up a Project

If you want to zip an entire project so it can be imported in another Eclipse install in its entirety, here's what you do:

- Go to the project in the package explorer view, and right click on the project folder, then select **Export...**
- In the **Select** dialog that pops up, expand the **General** item in the list of choices, then select **Archive File** and click **Next**
- In the **Archive file** dialog that pops up, in the left text area you will see the project name checked (if you expand this, you will see **bin** and **src** checked as well), and in the right text area you will see **.classpath** and **.project** checked. In the "To archive file" text field below, choose the zip file you want for your output, then click **Finish**

Zipping up source files

- Go to the project in the package explorer view, and right click on the **src** folder, and select **Export...**
- In the **Select** dialog that pops up, expand the **General** item in the list of choices, then select **Archive File** and click **Next**
- In the **Archive file** dialog that pops up, in the left text area you will see the project name with **bin** and **src** items, with the **src** checkbox selected. Do the following:
  - Expand the **src** folder - this will show all the package folders
  - *Uncheck* the **src** folder
  - *Check* all the package folders
- In the "To archive file" text field, type the complete path name of the zip file, or click on **Browse...** to select the directory you want the zip file to be stored in, and type the name of the zip file.
- Check the "Create only selected directories" radio button.
- Click **Finish**

The zip file should have all the source files from all the packages in your project, placed in the respective directories.

Importing a Zipped Project Into Eclipse

- Go to the **File** menu and select **Import...**.
- In the **Select** dialog that pops up, expand the **General** item, and choose **Existing Projects into Workspace**, then click **Next**
- In the **Import Projects** dialog that pops up, click the radio button for **Select archive file** and select **Browse..** to find and select the project .zip file on your computer
- Back in the **Import Projects** dialog click **Finish** and you will see the project created in your workspace.

Adding an External Jar File to a Project

Suppose you want to add a jar file called ext.jar to your project.

- Right click on the project name, and select **Properties**.
- In the dialog that opens up, click on **Java Build Path** in the left frame.
- In the right frame, click on the **Libraries** tab.
- Click on the **Add External Jars...** button on the right.

- Browse to the **ext.jar** file and click **OK**.
- You will see ext.jar appear in the **JARs and class folders list on the build path** text area. Click OK.

You will also see ext.jar under **Referenced Libraries** in your project in the Java package explorer perspective.