Handout 2: Editing, Compiling, Running, and Testing Java Programs

This handout describes how to perform common Java development with Eclipse.

Starting Eclipse

You should have already performed the initial setup described in **Setup handout**.

If Eclipse shows the welcome screen, containing only the text "Welcome to the Eclipse IDE for Java Developers" on a pretty background, switch to the code editor by going to Window > Open Perspective > Other... and selecting `Java (default)`.

Eclipse will start up, display a splash screen, and then show a workplace selection dialog asking you which workspace folder to use for this session.

Opening Files; Managing Multiple Files

Switch to the "Java" perspective in Eclipse if you're not already in it (**Window » Open Perspective » Other... » Java**).

You can open multiple files in Eclipse by double-clicking each of them from the Project Explorer pane. You can navigate through different files using the tabs on top of the editor pane.

Creating New Files

New Java files

To create a new Java source file (with the `.java` extension), select from the top menu **File » New » Class**. A window will pop up, asking you details about the class. Leave the source folder as it is, and select a package (e.g. `hw0`). Choose a name for your class (e.g. `MyClass`) Type this name in the "Name" field and click **Finish**.

(If you want your new class to be executable, it will need a `main` method. Eclipse can generate that automatically for you if you check the appropriate checkbox in the **New Java Class** screen.)

New Text files

There is a similar procedure for creating new non-Java files such as text files. Select **File » New » File** or **File » New » Untitled Text File**. In the resulting dialog box, choose the parent directory of your new file and type the desired filename. If you want to create a new directory, you can do so by appending the directory name in front of your desired filename. For example, if you want to create a file `problem0.txt` in the directory `hw1/answers`, but the `answers` directory does not yet exist, you can choose `hw1` as the parent directory, and then type `answers/problem0.txt` as the file name, and Eclipse will create the new directory and file for you.

Editing Java Source Files

Here are some useful actions that you can perform when editing Java code:

- Autocompletion
- Organizing Imports
- Viewing Documentation

Autocompletion

Autocompletion is the ability of an editor to guess what you are typing after you type out only part of a word. Using autocompletion will reduce the amount of typing that you have to do as well as the number of spelling mistakes, thereby increasing your efficiency.

Eclipse continuously parses your Java files as you are editing, so it is aware of the names of variables, methods, etc... that you have declared thus far.

CTRL+Space can be used to autocomplete most things inside the Eclipse Java editor. For example, if you have declared a variable named `spanishGreeting` in the current class, and have typed the letters `spanishGree` in a subsequent line, Eclipse can infer that you mean to type `spanishGreeting`. To use this feature, press **CTRL+Space** while your cursor is at the right of the incomplete name. You should see `spanishGree` expand to `spanishGreeting`.

Eclipse can also help you autocomplete method names. Suppose you have a variable `myList` of type `List`, and you want to call the method `clear`. Begin typing "`myList.`" — at this point, a pop-up dialog will display a list of available methods for the type `List`, and you can select the appropriate method with the arrow keys. You can force the popup to appear with **CTRL+Space**.

Organizing Imports

You can press **CTRL+SHIFT+o** to *organize* your imports in a Java file. Eclipse will remove extraneous `import` statements and try to infer correct ones for types that you refer to in your code but have not yet been imported. (If the name of the class that needs to be imported is ambiguous – for example, there is a `java.util.List` as well as a `java.awt.List` – then Eclipse will prompt you to choose which one to import.)

Viewing Documentation

Although you can directly browse the Java API and other documentation at the Oracle website, it is often useful to be able to cross-reference parts of your code with the appropriate documentation from within your editor.

Note that you need to generate the api docs locally before you can view docs for classes in the assignment.

In Eclipse, to view the documentation of a class that is referred to in your code, place your cursor over the class's name, and press **SHIFT+F2**. A web browser window will be opened to the class's documentation page. If the class is provided by Java, the documentation page will be on Oracle's website. (This may not work for all versions of Eclipse.)

For your own classes, you will need to tell Eclipse where to find their documentation. To do so, right click on the project name (e.g. `csci2600`) in the Package Explorer pane and click **"Properties"**. Select **"Javadoc Location"** in the left pane. Select the location, e.g. **"file:*YourWorkspace*/csci2600/doc/"**. (Note: the **"file:"**

portion is important, since the location is expected to be recognizable by a web browser.) After setting the Javadoc location path, click **OK**.

Compiling Java Source Files

You must compile your source code before running it. The `javac` compiler is used to transform Java programs into bytecode form, contained in a *class file*. Class files are recognized by their `.class` extension. The bytecode in class files can be executed by the `java` interpreter.

Eclipse is set up by default to automatically recompile your code every time you save. Classes with compile errors are marked in the **Project/Package Explorer** with red cross marks. The compile errors, as well as compile warnings, also appear in the **Problems** view (usually situated at the bottom panel of Eclipse).

If your file is saved and Eclipse says that it does not compile but you believe that it should, make sure that all of the files on which your file depends are saved and compiled. If that does not work, try using **Project » Clean**.

Running Java Programs

In Eclipse, to run a program, right click on the Java source file containing the `main()` method and choose **Run As... » Java Application**.

There is also a button near the left-hand side of the Eclipse toolbar which will re-run the last application (or JUnit test, see below) that you ran.

Testing Java Programs with JUnit

JUnit is the testing framework that you will use for writing and running tests.

For more information, visit:

- http://junit.org, the official web site.
- JUnit Cookbook, a brief tutorial.
- JUnit API

JUnit is integrated with Eclipse, so you can run the test suite from within the IDE.

- First, select the test you want to run (e.g., `BallTest.java` in `hw0.test`) from the **Project/Package Explorer** (usually the leftmost pane). You can also select an entire directory (e.g., `hw0.test`) to run all test in it.
- From the Eclipse menu at the top of the screen, select **Run » Run As » JUnit Test**. You can also right-click on the selection and select **Run As » JUnit Test**.
- The JUnit GUI should pop up and run all the tests. You can double-click on failed tests to jump to the code for that test. When you're done inspecting the JUnit results, close the JUnit pane to go back to the Project/Package Explorer.

Most likely, you'll have to explicitly add the JUnit 4 library. Go to **Project » Properties » Java Build Path » Libraries » Add Library... » JUnit** then click Next, select JUnit 4, then click Finish.

Parts of this handout are copied from University of Wanshington's\ Software Design and Implementation course