

CPSC 1180

Lab 1: Introduction to Eclipse

Objectives

1. To become familiar with the basic functionality of the eclipse IDE
2. To write and debug a simple program using eclipses integrated tools
3. Install eclipse, and other course essential software on a personal computer

Introduction

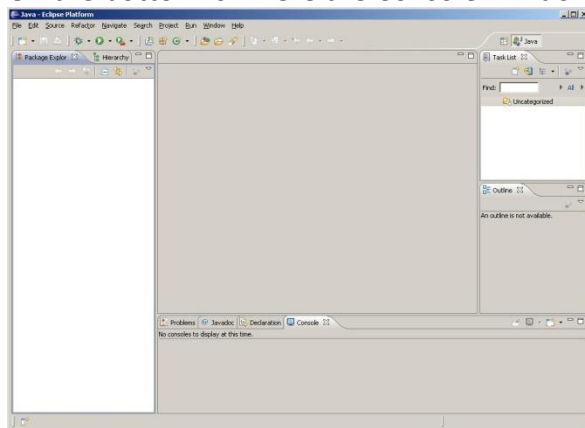
Welcome to CPSC 1180. If you have not received information about how to hand in labs and submissions standards, consult your lab T.A. The labs in this course will teach you how to write object oriented programs in a modern development IDE. Through object oriented techniques and planning you will learn to write programs dramatically more complex and sophisticated than you have in previous courses. In this lab you will be introduced to your development environment eclipse, as well as some of the tools eclipse offers, such as the debugger. Further information about eclipse and its extensions can be found at their website. If you are familiar with Eclipse or another IDE this lab may seem very simplistic, in that case think of this as more of a reference than a lab and follow along to refresh your memory.

Procedure: (part 1 - a simple program)

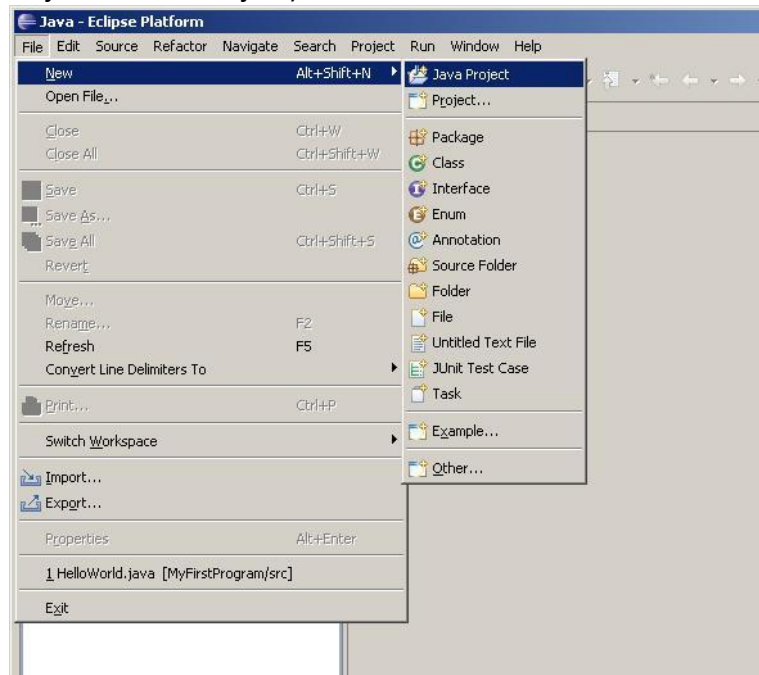
To begin launch eclipse by pressing Start → all programs → development software → eclipse for java. At which point you will be prompted to set up a work space folder. Your H: drive is suggested.

After loading you will be prompted with the welcome screen which should be closed. You will now see the eclipse workspace.

- a. To the left is the Project Explorer where you can view projects that you are working on.
- b. In the center is the space where your code will go.
- c. On the bottom is where the Console Window and Errors will appear.



At the top, click File->new->Java Project (or you can right click in the Project Explorer field and click New -> Project -> Java Project).



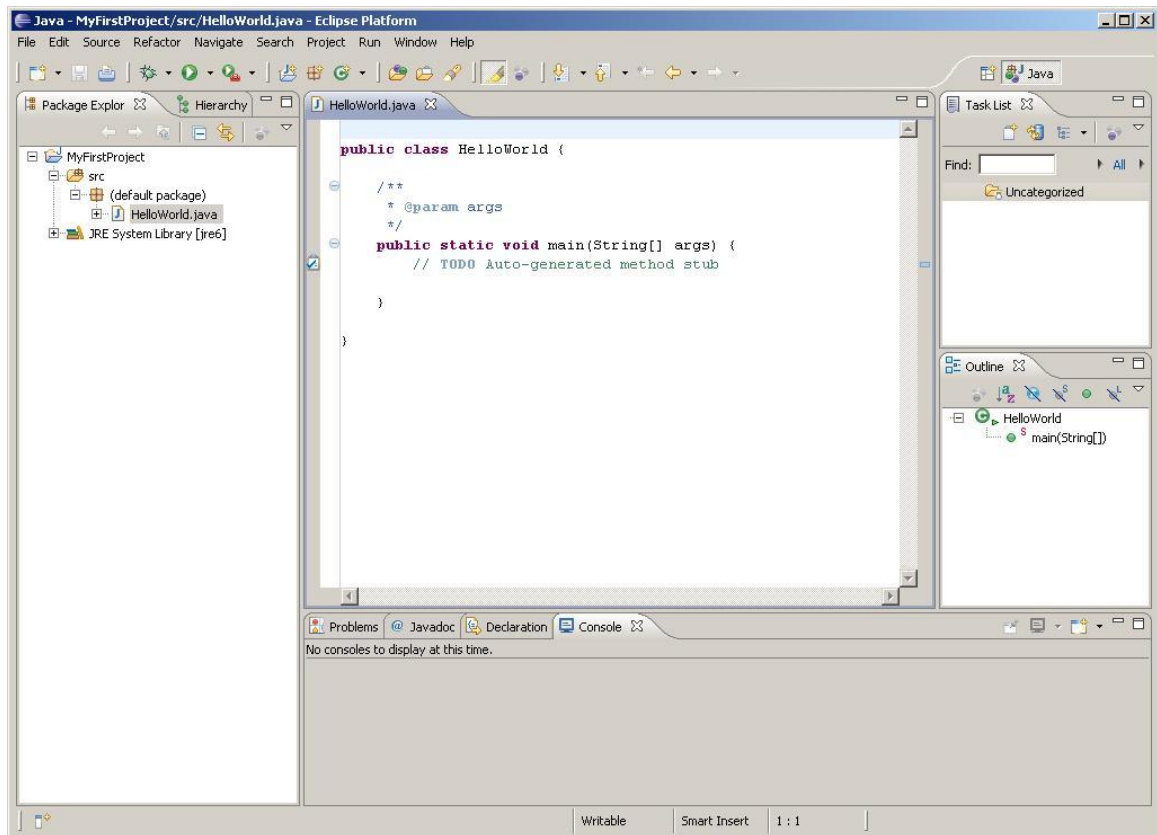
Give the project a name, such as assign1 or MyFirstProgram and then click Finish.

- a. This will create a project folder in your workspace folder.
- b. This will also place your project in your Project Explorer window.

Now that you have created your project folder you need to add a java class file. Just as before click File->new->class. Give your file a name, such as assign1 or HelloWorld. Then check 'public static void main(String[] args)' and click Finish.

One of the major advantages of an IDE such as Eclipse over simple text editors is that they reduce the amount of repetitive coding that needs to be written. Statements like public static void main(String[] args) prefix most java programs so the process of typing it has been streamlined.

Your HelloWorld.java file should appear in your Project Explorer and the code should show up as its own tab in your code window.



Below the TODO comment, type in:

```
System.out.println("Hello World!");
```

You may notice that sometimes while you are typing a drop-down box will come up with suggestions. You may use the mouse or arrow keys to find the method you need and then press Enter to place it into your code. This is often helpful to save typing and the method descriptions can sometimes help you figure out which one you need.

Notice that after the print line has been typed an * appears next to the file name in its tab. This means that the file has not been saved. Click File->Save to save the file.



Click Project->Build All. This will compile and build your project and will also report any errors that exist. Errors will appear in the Problems tab of the bottom window.

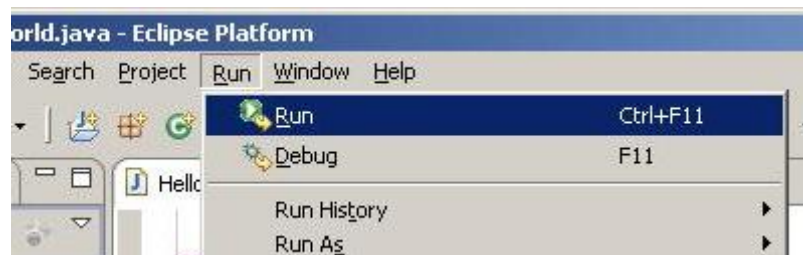
A red X will also appear next to the line of code that is incorrect (here a semi-colon is missing).

```
public static void main(String[] args) {  
    // TODO Auto-generated method stub  
    System.out.println("Hello World!")  
}
```

The X, as well as the highlighting is known as static analysis, many tool exists that will do more than simply underline incorrect syntax some can find memory leaks, as well as identify inefficient code. Customizing your development environment with a suite of static analysis tools will help you write better code.

The shortcut for build all is Ctrl + B, and the shortcut for run is Ctrl + F11. These are probably the most common shortcuts that you will use during the debugging and testing stage so become familiar with them. A list of essential Eclipse shortcuts can be found at <http://www.rossenstoyanchev.org/write/prog/eclipse/eclipse3.html>. And a complete list can be found at <http://dgc.ethz.ch/lectures/ss07/vs/material/EclipseShortcuts.pdf>. Familiarity with shortcuts will dramatically increase your speed and efficiency while coding in eclipse.

When no errors are reported, click Run->Run to run your program.



The program will run and display in the Console tab at the bottom of the window.



When you have finished coding, running, and testing your program, you should close your project. Right-click on the project's name in the Project Explorer and then click Close Project. Then you can exit the program by clicking the red X in the corner of the window.

Most of your labs will not be able to be finished during the assigned lab time. Because of this, you will need to bring your project folders home with you. For this purpose you can use a variety of methods.

- 1) A portable flash drive
- 2) File sharing websites such as dropbox or google drives
- 3) Or public coding repositories with version control such as Github

It is suggested that you use at least one of these methods to back up your code as well. Github is the most secure but also has the steepest learning curve. However, the functionality can be integrated directly into eclipse.

In order to save your project to another storage location

- a. Navigate to the workspace folder on the H: drive and locate your program.
- b. Right-click and copy the java project.
- c. Navigate to your alternative drive or storage location and paste the project there
- d. To keep track of your different assignments you should keep each of the different assignments in different folders within your CPSC1181 folder.
- e. ***Note*** You may also just back up the java files rather than the whole project folder.

If you have only backed up the java files, follow this procedure to open your saved project:

- a. Open Eclipse.
- b. Click File->New->Java Project.
- c. Give the project a name and click Finish.
- d. Expand the project in the Project Explorer to reveal the src folder.
- e. Navigate to your java file that you backed up on your alternative drive.
- f. Drag that file into the src folder in the Project Explorer.
- g. Double-click your java file to open it.
- h. Edit, save, build and run your program as needed.
- i. ***Note*** This makes a copy of your file and saves it in the workspace; the original will not be modified, so remember to back up the file to your alternative drive when you are finished working on it again.

(Part 2 – debugging a program)

Debuggers are perhaps the most important tool associated with IDE's, in previous courses you may have tried to test your program by displaying large amounts of output so that you could analyze the execution of your program after it completed. A debugger allows you to control the speed of the execution while monitoring the values of variables and the state of your program. Debuggers make it very easy to isolate and identify errors in your programs.

Create a new class like before and write the following code to swap the value of two variables

```
public class SwapVals {  
    public static void main(String[] args) {  
        int swapa = 1;  
        int swapb = 2;  
        System.out.print(swapa+" ");  
        System.out.println(swapb);  
        //swap values  
        swapa = swapb;  
        swapb = swapa;  
        System.out.print(swapa+ " ");  
        System.out.println(swapb);  
    }  
}
```

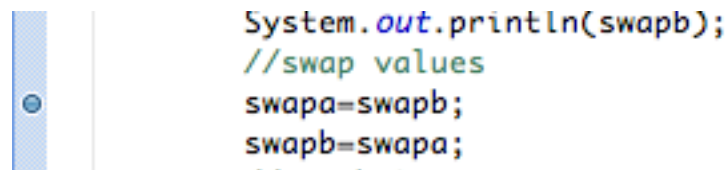
Save, build, and run the code. The output of the code looks like:

```
1 2  
2 2
```

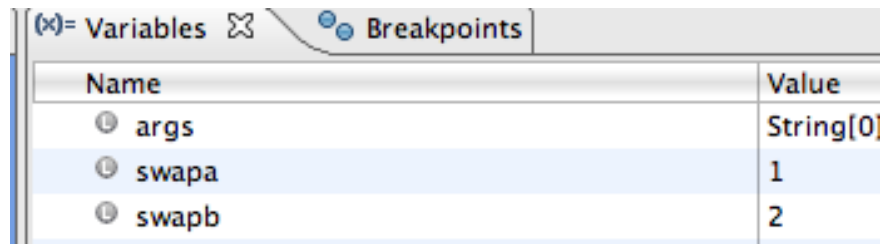
This run output is not what we would expect, by looking at the code we can see that the values of swapa and swapb are supposed to be switched, but something is missing, because when the code is run, the values are coming out incorrect. What went wrong?

We can use Eclipse's Debugger to help us figure out what is happening.

- a. Double-click on the blue grey bar to the left of your code. A blue dot will appear; this will set a breakpoint in your code. It tells Eclipse where to pause the run so you can watch step by step how the code is executed.

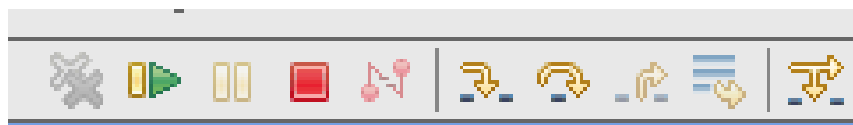


- b. Since we know we don't have any problems before the swap (the first set of values printed out correctly), set a breakpoint on the first swap (line 15).
- c. Save, build, and then click Run->Debug. The window will change slightly and new panes will be displayed.
- d. The values of the variables are displayed in the variables tab of the upper right hand corner pane.

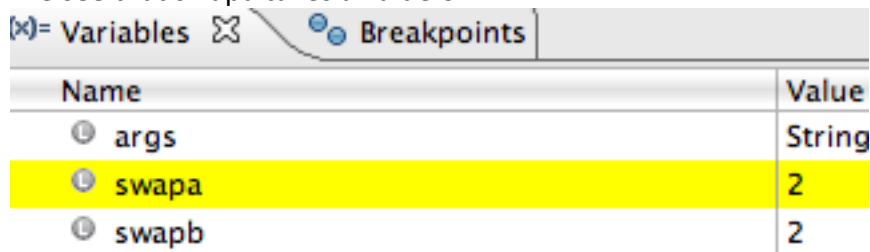


Name	Value
args	String[0]
swapa	1
swapb	2

- e. The values are displaying correctly and the execution has paused on line 15, which has not been executed yet.
- f. Above the upper left pane are the tools for controlling the execution of the program.



- i. The green arrow is to continue debugging, which will move to the next breakpoint that is set.
 - ii. The red square is to halt debugging.
 - iii. The first yellow arrow is step into, which will proceed to the next step and will move into any functions that it comes across.
 - iv. The second yellow arrow is step over, which will skip over any functions and proceed to the next step.
- g. Click step over once.
- h. We see that swapa takes a value of 2.



Name	Value
args	String
swapa	2
swapb	2

- i. If we click step over again we see the same result. So the second assignment did nothing.
 - j. This makes us realize that swapa's value is being written over before we can assign it to swapb. Which means we need a temporary variable to hold the data while we assign the values.

- k. Stop the debugger by clicking the red square.
 - l. Switch back to the normal view by clicking Window->Open Perspective->Java.
2. Now, if we fix the code, it should look something like this:

```
public class SwapVals {  
    public static void main(String[] args) {  
        int swapa = 1;  
        int swapb = 2;  
        int temp = 0;  
        System.out.print(swapa+" ");  
        System.out.println(swapb);  
        //swap values  
        temp = swapa;  
        swapa = swapb;  
        swapb = temp;  
        System.out.print(swapa+ " ");  
        System.out.println(swapb);  
    }  
}
```

3. Now the value of temp holds the value of swapa so that it can be reassigned to swapb.
4. Remove any breakpoints by double clicking on them again.
5. Save, build, and re-run the program.
6. The run output now looks like:

1 2

2 1

The code now does what it was intended to do. This is the end of the in class lab, the rest of the lab must be completed on a home or mobile workstation.

(Part 3 – Home Installation)

It will be very difficult during the latter labs to finish all of your work at school, because of this, and to gain familiarity with eclipse you should install eclipse at home.

Instructions for Windows users:

1. If you have not already installed JRE for previous development go to Java's download page at: <http://java.com/en/download/manual.jsp>

- a. Choose the download type for your operating system and run the executable file when it is finished downloading. If you are using Eclipse x64, make sure the JRE is also x64.
2. Once the JRE is installed, go to Eclipse's download page at:
<http://www.eclipse.org/downloads/>
 - a. Choose the IDE that you would like to use.
 - i. Eclipse Classic (currently 4.2.1) is recommended
 - b. To the right of the IDE, choose Windows to download the file.
 - c. A download site will be chosen for you, or you can select one from the list that appears.
 - d. Select a location on your computer to save the zip file to.
3. When the file finishes downloading, extract the executable with WinZip, WinRAR or some other file archiver to an appropriate location on your computer (ie. C:\eclipse).
 - a. Place in a location without any spaces in the path.
4. Run Eclipse by double-clicking on the executable (eclipse.exe) in the eclipse folder.
 - a. This is not going to run an installer, double-clicking on the executable will launch the Eclipse environment.
 - b. You may create a shortcut to Eclipse by right clicking on eclipse.exe and choosing Create Shortcut and dragging the shortcut to a more convenient location. (ex. Your Desktop or into your Start Menu)
5. A workspace folder will be created the first time Eclipse is launched, specify a location when prompted (ie. C:\eclipse\workspace).
 - a. It is important that the path to the workspace folder does not have any spaces in it.
 - b. You may create a shortcut to this folder if necessary.

Instructions for Mac OS X users:

1. Mac OS X already comes with a JRE already installed, however, if you would like to make sure that yours is the latest version you may run the Mac Software Update from the Apple menu.
2. Go to Eclipse's download page at: <http://www.eclipse.org/downloads/>
 - a. Choose the IDE that you would like to use.

- i. Eclipse Classic 3.4.1 at the bottom of the list is recommended.
 - b. To the right of the IDE, choose Mac OS X to download the file.
 - c. A download site will be chosen for you, or you can select one from the list that appears.
 - d. Select a location on your computer to save the tar.gz file to.
3. When the file finishes downloading, double-click the tar.gz to decompress the Eclipse files.
 4. Drag the Eclipse folder to your Applications folder in the Finder window.
 5. Double-click on the Eclipse Application in the Eclipse folder to launch the Eclipse environment.
 6. A workspace folder will be created the first time Eclipse is launched, specify a location when prompted (ie. Documents folder).

(Part 4 – Customization (Optional))

While not necessary, the customization of your IDE will greatly increase your programming speed and familiarity with the IDE itself. If you downloaded the classic version of eclipse it does not have the eclipse marketplace installed. To install the eclipse marketplace press

Help → Install new Software

Under the work with tab drop down to **Juno**, then drop down **General Purpose Tools**, and select **Marketplace Client** then follow the on screen wizard to complete the installation. The marketplace acts as a vista to all of the tools developed for eclipse.

Now under the help tab you will see the Eclipse Marketplace, There are thousands of development tools available, it is suggested that you gain some experience with eclipse before adding too many extensions, some of them will generate a large amount of information about your code which can be overwhelming.

You may find the following tools useful. It is suggested that you look into each one before installing them.

Subclipse – A well supported eclipse plugin for **Subversion**. Subversion is version control software that will be essential if you are working on a project with multiple people.

EGit - A Git plugin, Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Checkstyle - Checkstyle is a development tool to help you ensure that your Java code adheres to a set of coding standards. Checkstyle does this by inspecting your Java source code and pointing out items that deviate from a defined set of coding rules.

AnyEdit - AnyEdit Tools plugin adds several new tools to the context menu of text- based Eclipse editors, to output consoles, to Eclipse main menu, editor toolbar and navigator/explorer.

FindBugs - FindBugs is a defect detection tool for Java that uses static analysis to look for more than 200 bug patterns, such as null pointer dereferences, infinite recursive loops, bad uses of the Java libraries and deadlocks.

MouseFeed – **Not intended for experienced users.** MouseFeed helps to form a habit of using keyboard shortcuts. When the user clicks on a button or on a menu item, the plugin shows a popup reminding about the key shortcut. After a few times, you remember the keyboard shortcut and you will start using it, rather than clicking through menus. This will save you a lot of time at the end of the day.