Laboratory 1 – Week 2

An Introduction to Java and Eclipse

### 1.1 Introduction

This worksheet is a short session to familiarise you with the Java and Eclipse programming and development environment.

You should have all run a program that tested your **Finch** robot during induction week. If you have not done so, make sure you do before attempting this laboratory sheet, i.e. complete sheet 0. This laboratory is similar to worksheet 0, but just involves writing your first program from scratch and covers some of the concepts that we “glossed” over.

**1.2 Preliminaries**

Make sure you have access to your home directory (we will assume this is the H:\ drive). Eclipse should create a place for all of your workspaces in H:\eclipseWorkspace. Check that this is the case.

### 1.3 Starting and Ending Eclipse

If you want to install Eclipse on your own computer (this is highly recommended) then you can do so from <http://www.Eclipse.org>. Here you can download the version of Eclipse you are after (we suggest either the 32 bit or 64 bit Eclipse IDE for Java Developers). You need the same number of bits (32 or 64) as your installed version of the Java runtime libraries which can be found at <http://www.Java.com>. This website will also determine if your computer has Java installed or not. If you are not sure if you need the 32 or 64 bit versions of Java and Eclipse, the general rule is that if you choose the version that matches your version of Windows (i.e. 32 or 64 bit). Note that any version you download from the Eclipse website might be slightly different to the versions that we are running on the department machines, but all of the concepts should be the same.

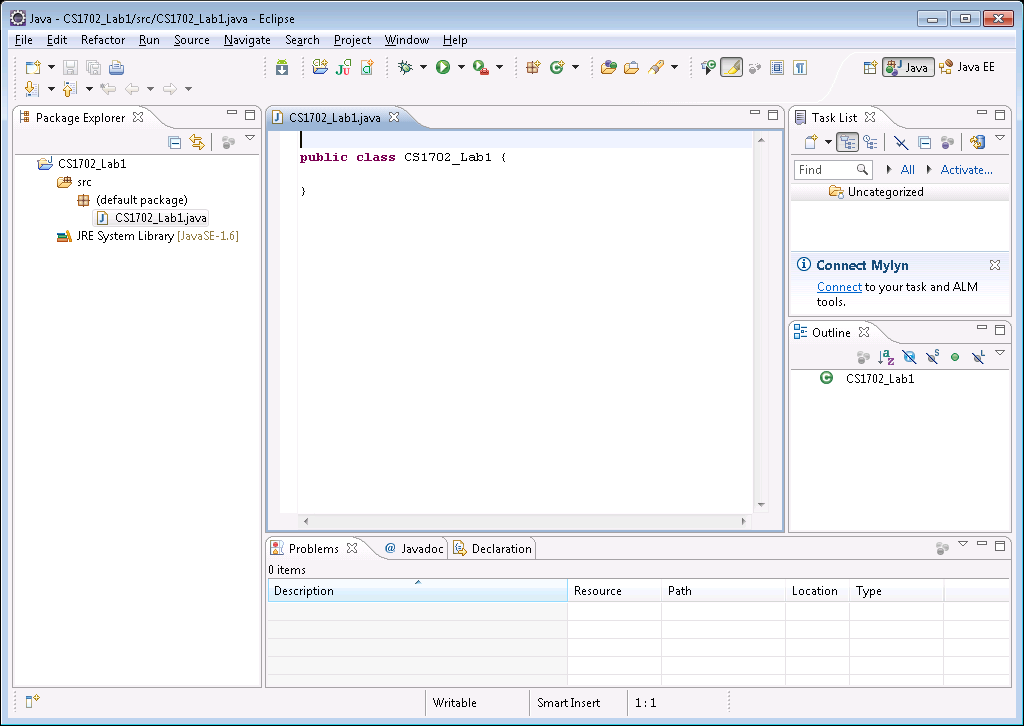
On the department machines, Eclipse can be found through searching for the word Eclipse through either the Windows 10 button or the Windows 10 *Search* button

This program works similar to any other software product, i.e. you open up the program, and you then either create a new program or load an existing program. When you have finished, you exit Eclipse by selecting *File-Exit*.

### 1.4 Java and the Eclipse IDE

Follow the instructions in Laboratory 0, but instead of creating a project and class called DoesMyFinchWork, create a project and class called CS1702\_Lab1. Note that this is just a suggested naming style, anything descriptive is fine.

You should now have something similar to:



The components of the Eclipse IDE that we will be working with are as follows:

**Workspace.** As discussed previously, the default Workspace location is H:\eclipseWorkspace. You can change your Workspace to any directory you wish, however you will not be able to access the projects/programs from the previous Workspace.

**Package Explorer.** This is a list of all of the projects (programs) you have created in the selected/current Workspace and is positioned in the left hand pane of Eclipse. All of the projects will be listed here in alphabetical order. Expanding the project name will give you access to the source code (within src and the (default package)).

**Console Window.** Any output generated by your programs will be displayed in this Window which is positioned in the bottom middle pane of Eclipse.

**Javadoc Window.** This pane appears alongside the Console pane and gives you context sensitive help pages on some of the aspects of Java. We will look at this later on in this worksheet.

If you close any of the panes within Eclipse and then find you need them again, you can reopen them using the Eclipse menu option *Window* and then *Show View*.

Exit Eclipse and open up Windows Explorer and find the following directory H:\eclipseWorkspace\CS1702\_Lab1\src. This assumes that you have placed your projects in the H:\eclipseWorkspace directory. Open up the file called CS1702\_Lab1.java using Notepad. This is the source code of your new project/Java program and should match the program that you saw in the Eclipse main window. Source code files are simply text files with a .Java extension (of course they must also contain Java code!). All of the other files placed by Eclipse in the CS1702\_Lab1 folder and sub-folders are automatically generated. These can easily be recreated, however make sure that at a bare minimum you keep your .Java files.

**Make sure you back up all of your Java programs onto the University file servers (your H:\ drive) and another location, i.e. a memory stick. Losing your Java programs will not be considered as a mitigating circumstance.**

Replace the program code (using cut and paste) with the following:

**public** **class** CS1702\_Lab1

{

**public** **static** **void** main(String args[])

{

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

}

}

Running the program (pressing the green play button as in Laboratory 0) should result in the following being displayed in the output (Console) Window at the bottom of the Eclipse program.

Hello World!

The components of this simple program are as follows:



**Class Definition.** We will be working with mostly one class to start with. The two keywords public class are always needed to define a class. The name of the class will be added for you by Eclipse. We add our Java code within the class, i.e. within the braces, coloured red in the diagram above. We will cover classes and Object Orientated programming later on in the module.

**Main Definition.** When we run a program, Eclipse looks for a section of code called main within the class file of our program and if it can find it, will run the code within it. Do not worry at this stage what the sections of the definition of main mean. All we need to know that any code within the main definition will be executed when we run our program. Code needs to be placed within the braces that follow the definition of main, this is coloured blue in the example above.

**System.out.println().** This is a command that simply displays in the Console Window anything that is within the end brackets. In the case above, this is the text “Hello World!”.

**Program Statements.** Every program statement must end in a semi-colon “;”.

### 1.5 Exercises

The following is a short list of tasks aimed at demonstrating some of the features of Eclipse and Java.

1. Change the text from "Hello World!" to something else. Run the program to see the results. Remove the semi-colon ";" from the program. Try running it again - what happens? Notice that the last bracket on the line if now underlined in red and that a red and white cross has appeared in the margins of the code next to where you made the deletion. This is how Eclipse tells you that there are *syntax* errors in your program. Missing a semi-colon from the end of a line is the most common type of error when programming in Java. Click on the *Problems* tab next to the Console output Window. This is where Eclipse lists all of the errors in your program. *Warnings* are possible mistakes (but your program will still run) and *Errors* need to be fixed before your program will run. Put the semi-colon back - what happens?
2. Double click on the partial statement println and then click on the *Javadocs* tab. Notice that a short help page has been brought up detailing what println does. The full *Javadocs* can be found at <http://docs.oracle.com/javase/8/docs/api/>, however this might not mean much at the moment. We will use the *Javadocs* extensively in some of the later laboratories.
3. Copy the whole line that prints to the *Console* Window and paste it underneath several times. Run the program again - what happens? You should get several sets of outputs. Experiment with changing the text so that you get different sets of output on each line. Delete the "ln" part of some of the printing lines. Run the program again - what happens? There are two type of displaying, println will add a return (new line) after the text whilst print will not.
4. Try cutting and pasting code around, and adding more code to get yourself familar with the editing system for writing code. Try using the search and replace (*Edit*-*Find/Replace*) to do question 3) above.

### 1.6 Completing Worksheets

Type (copy and paste) the following program into Eclipse (this assumes that your project is called CS1702\_Lab1).

**public** **class** CS1702\_Lab1

{

**public** **static** **void** main(String args[])

{

*Method1*();

*Method2*();

*Method3*();

}

**public** **static** **void** Method1()

{

System.***out***.println("Hello from Method1!!!");

}

**public** **static** **void** Method2()

{

System.***out***.println("Hello from Method2!!!");

}

**public** **static** **void** Method3()

{

System.***out***.println("Hello from Method3!!!");

}

}

Run the program and observe the output. The main *method* runs the code within the three defined methods Method1, Method2 and Method3 in the order they are called (listed). The process of breaking down a task into a set of sub-tasks is very important and should be something that you start doing for all of your Java programs.

Experiment with changing the text and the order the methods are called. Add extra calls, e.g. call Method2 twice or three times. Create an extra method called Method4 based on one of the others.

You are going to be assessed on some of the material from some of the worksheets (see the study guide and task details for more information). We expect you to structure your code logically using multiple methods, (in a similar manner to the example above), for example if worksheet seven (7) has four (4) questions, you might create methods:

Answers\_Q7\_1, Answers\_Q7\_2, Answers\_Q7\_3, Answers\_Q7\_4

Structuring your code this way is **VERY**, **VERY** important!