Week 1 Lab

Welcome to FIT2099

Objectives

In this week's lab, you will:

- meet your demonstrator
- meet your classmates
- learn how to use Eclipse to edit and build a Java project
- learn how to import code into an Eclipse project

Preparation
For this week's lab, you must prepare answers to the following questions about yourself:

- What is your name?
- What degree are you enrolled in?
- What's your dream career?
- What is the biggest challenge or difficulty you have faced when developing a piece of software so far? Explain why.

Task 1. Introducing yourself

The first part of this lab will be devoted to introductions. Share your answers to the questions in the Preparation section with the class, and listen to theirs. Remember, you will be working with one of your classmates for the assignments and some lab exercises, so it's important to get to know them.

Task 2. Accessing the development environment

Our default Java IDE (Integrated Development Environment) for this unit is Eclipse[1]. The examples in lectures and screenshots in lab sheets all use Eclipse because it is available in the PC labs; however, if you are bringing your own laptop to class, you may use any IDE you like as long as it can build Java 8 SE. We won't provide any support for any environment other than Eclipse, though — if it breaks, you get to keep both pieces.

Installing Eclipse on your own computer is quite straightforward. Go to the Eclipse website (`http://www.eclipse.org`) and follow the instructions. Note that we can not promise to be able to give you technical support if you have trouble installing Eclipse on your own computer. In this case we recommend using the lab PCs.

At Clayton, you can start Eclipse by clicking "Eclipse" in the Windows 10 start menu. Note that it takes a few seconds for Windows to populate the start menu on the lab PCs, so be a little bit

[1] If you would like to use a different IDE, such as IntelliJ or NetBeans, please make sure that you know how to use it to do all the things this guide shows you. You'll also be using git to share files later in the semester, so you may wish to explore your IDE's git integration.

patient if you can't see it initially.

If you are at Malaysia, ask your lecturer for instructions to start Eclipse.

Ensure that you can access Eclipse, or another Java development environment, before you proceed with this lab sheet.

Task 3. `Hello World!` in Java

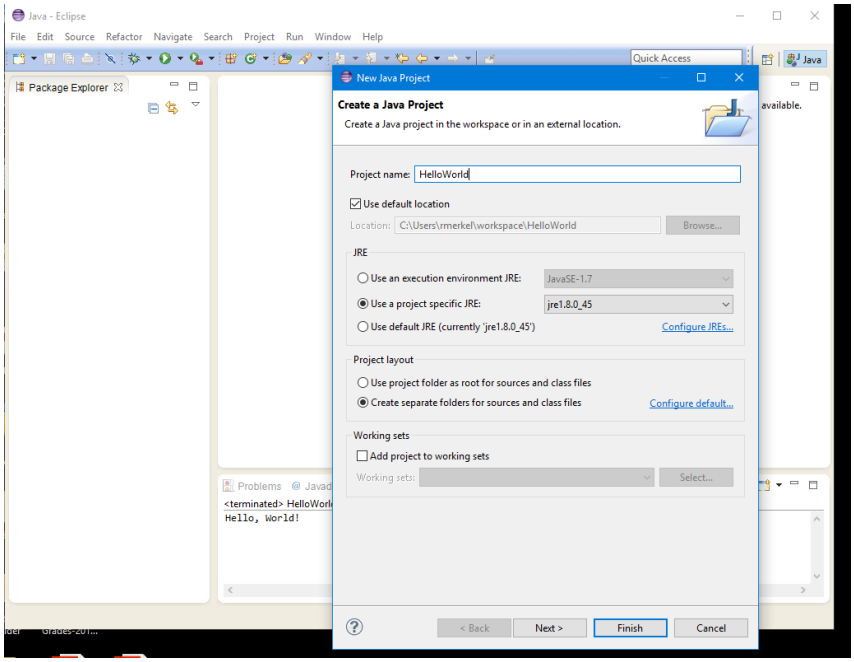
Next, create your first Java program. By tradition, the first program you create in a new language should print the string "Hello, World!". We'll stick with that tradition.

Projects in Eclipse

Java programs almost always contain more than one file of source code, and may contain hundreds of them. This can be hard to keep track of, especially if you are responsible for maintaining several programs. IDEs like Eclipse help you deal with this complexity by organizing your code (and other program assets such as images or data files) into *projects*. In fact, you can't even compile and run a program in Eclipse unless you create a project for it. That's because, as well as organizing your source code for you, the Eclipse project file also keeps track of information such as which system libraries to use.

That means that our first step will be to create a new project to store our source code. To do this, select `File` → `New` → `Java Project` from the menu. A dialog box will pop up where you can select a name and some other properties for your project.

Enter the name `HelloWorld` as the name of the project. Note that there is a section in this dialog box labelled "JRE", for Java Runtime Environment. This is the interpreter that will actually execute your Java instructions – without one, you can't run a Java program. We will be using JRE 1.8, which is the default in the labs at Sunway and Clayton. If you are on your own machine, you should ensure that you have a JRE or JDK 1.8 installed (`http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html`), and if necessary select it using the "Use a project specific JRE" option.



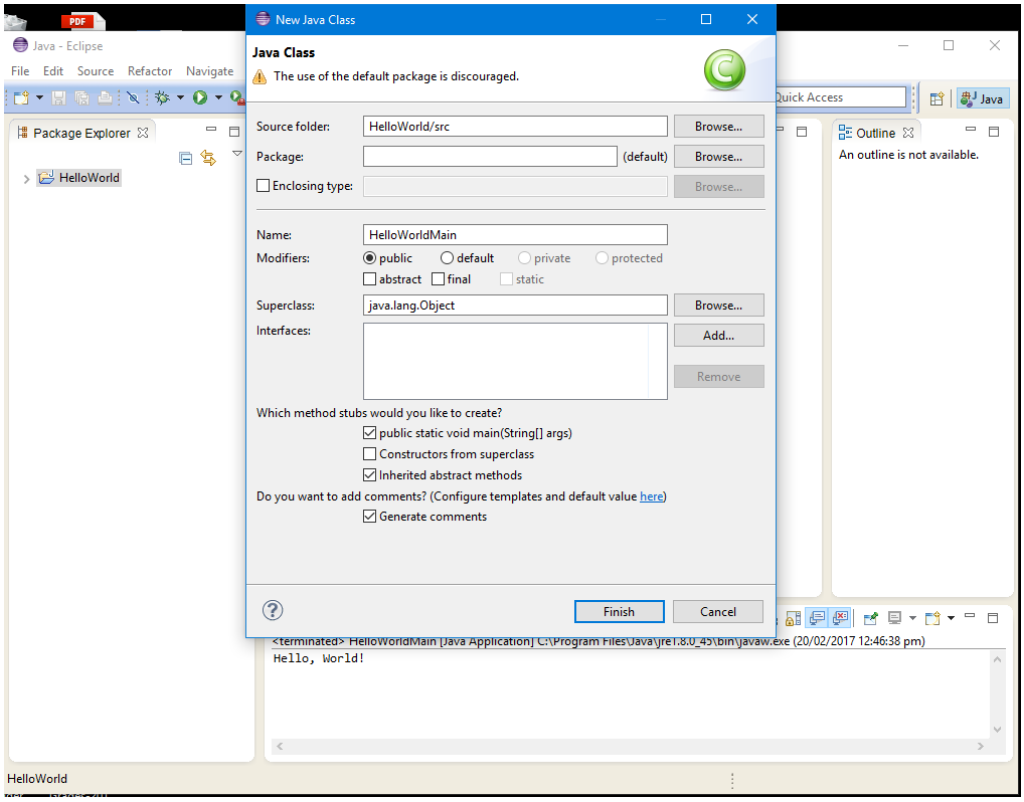
The `HelloWorld` project is then added to your Eclipse Workspace. This is a folder on your shared drive where all your working source code is stored.

Adding a class

Next, you will need to create a class - in Java, all program source code is organized as part of a class. This project is very simple, so we will only need a single class with a single method.

To create the new class, right-click on the `HelloWorld` project and select `New` → `Class`. Name the class `HelloWorldMain`[2]. Select the checkbox that says that you would like to create a method stub for `public static void main(String args[])`, as well as the checkbox for "generate comments".

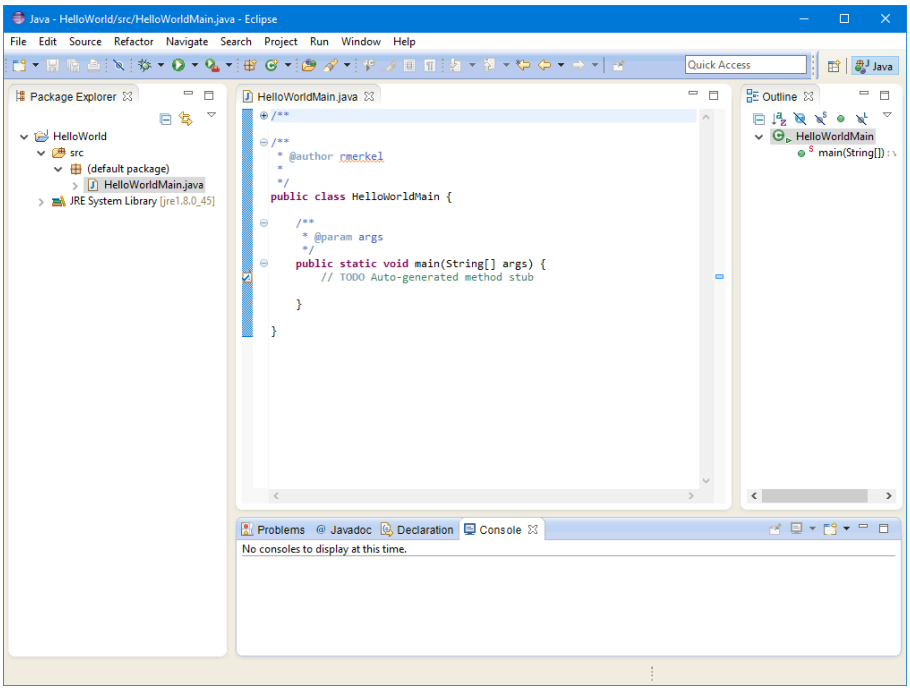
You may get a warning about "creating a class in the default package". We'll discuss Java packages, and why it is a good idea to use them, later in semester; however, for the time being, it's enough to know that you can safely ignore this particular warning.



The source code for the new class appears.

You'll see a source code window something like the following:

[2]Note: while it is not compulsory to use CamelCase to name Java classes, it is is a very widely-used convention. See for instance Google's Java style guide (`https://google.github.io/styleguide/javaguide.html`).



You can run this program right now in several ways. One way to do so is to right-click the HelloWorld project folder icon, and select `Run As` → `Java application`.

The program will compile and run. It will, however, do nothing, as the body of the program is completely empty.

Now, we'll add some code to do something.

Writing the `main()` method

In a standard text-based Java program, execution begins with a method called `main()`. Generally, you should only have one `main()` method in each Java project.

Where you see the comment line

```
1 // TODO Auto-generated method stub
```

replace it with the following source code line.

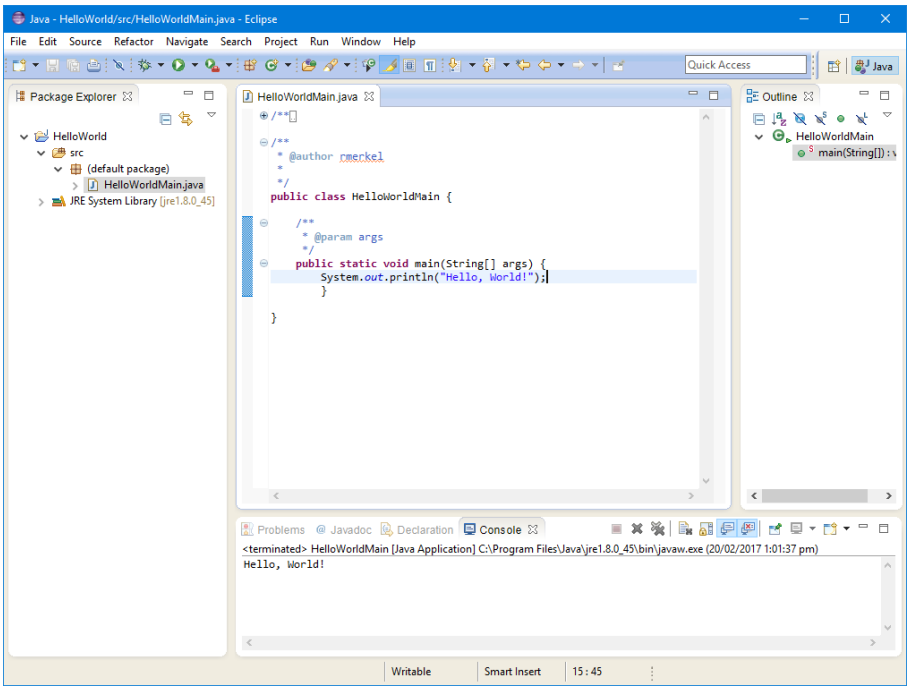
```
1 System.out.println("Hello, World!");
```

Tip: Type this by hand rather than copying and pasting! Pasting code from a PDF into an IDE often fails to work. The PDF may contain smart quotes or other characters that don't belong in your source code.

As you type the line of source code, an X in a red circle will appear to in the left margin of the line you're typing on. This indicates that your code is not (yet) syntactically correct Java. Eclipse is generally very good at identifying syntactic mistakes in Java code as you type. When you are finished typing the line, the X should disappear. If it doesn't, you have made a mistake somewhere; if you can't find it ask the person next to you for help. If neither of you can figure it out, ask your demonstrator for help!

Save your modified class by clicking on the save icon in the toolbar.

Run the modified code. Confirm that it indeed prints "Hello World!" in the console log:



Check with your demonstrator before proceeding to the next exercise.

Task 4. Importing and exporting code with Eclipse

Despite what popular culture would have you believe, programmers very rarely work alone on a project. In order to work as a programmer, you'll need to learn how to share your source code with teammates.

Later on in this unit, you will be using version control systems which allow programming teams to create shared file repositories. These not only allow easy sharing, but keep track of all the thousands of changes that a team of programmers will make in the course of a large project. Sometimes, however, it can be very useful to make a zip file containing an Eclipse project to send around, so we'll try that now.

Download `MovingBall.zip` from the FIT2099 Moodle page. This archive contains an Eclipse project for a slightly more complicated Java program.

Select `File` → `Import` from the main Eclipse menu. In the dialog box, select "Existing Projects Into Workspace", select the file you've downloaded, and click "Finish".

A new project called `MovingBall` will appear in the Package Explorer on the left of your Eclipse window.

Try to run `MovingBall` in the same way you ran `HelloWorld`. You'll see a window pop up, and a circle will gradually move across it.

Now change the window title from `Sample Frame` to *`*name*`* `modified this` where *`*name*`* is your name. (If you can't figure out how, ask your demonstrator).

Run your modified code to ensure that it still works.

Now, export your project to a zip file archive. To do this, right click on the `MovingBall` project in the Package Explorer, select `export`, then select `General`, then `Archive File` in the dialog box, then click the `Next` button. Ensure that you include the source code files, but *not* the contents of the `bin` directory, in the export. Finally, choose an appropriate file name to save to and click the `Finish` button to create the archive.

Now, you're going to swap your archives with a classmate, but before we do that, you should rename your own project so that it doesn't get overwritten when you import your classmate's archive.

Right-click on the `MovingBall` icon again. Select `Refactor` → `Rename`. Change the name of your project to `MovingBallTmp`.

Now, swap archives with a classmate. How you do this is up to you – you can use a USB stick, email them, or share them via Google Drive or Dropbox. Import their archive into Eclipse as before. Confirm that you can compile and run their project.

Congratulations

You've completed the introductory lab. Be aware that future labs will:

- be more challenging, and
- be assessed.

Preparing for these labs in advance is strongly recommended!

See you next week.