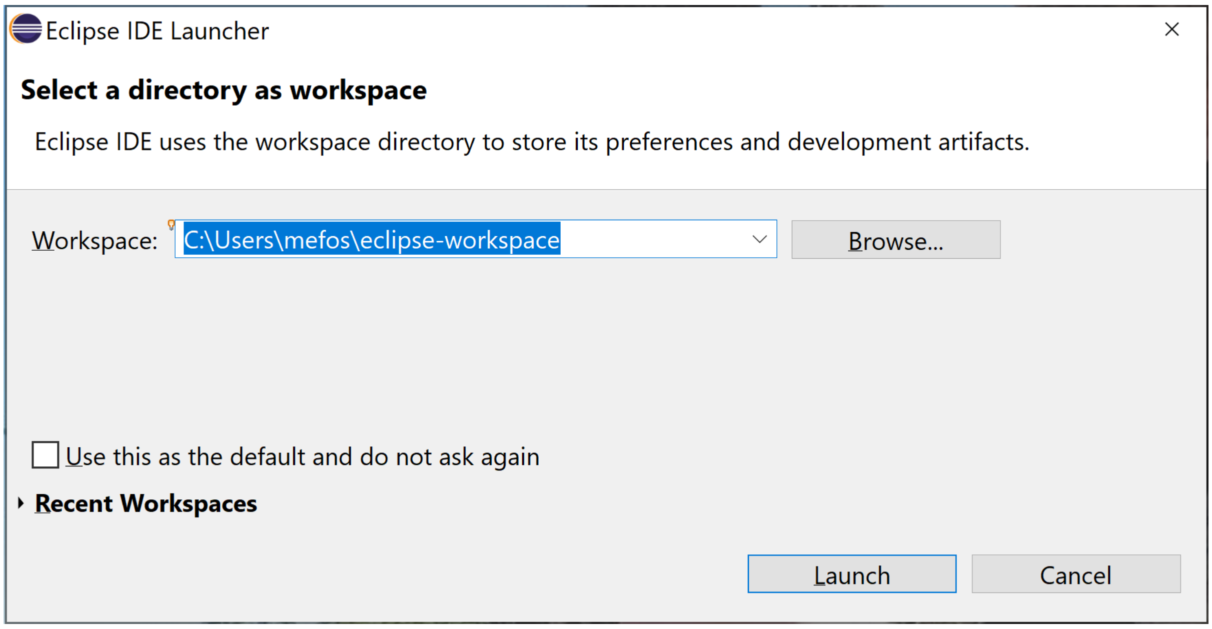
Running Eclipse

Using Eclipse

Eclipse is an open-source Integrated Development Environment (IDE) that will be used for coding, compiling, running, and debugging Java programs. Eclipse provides many more facilities than this, some of which you will use in later courses. It is a many-featured and versatile piece of software, and can appear a little daunting for beginners. We will be using only a small subset of its capabilities, and a key objective of this lab session is to begin the process of familiarisation with these.

- Launch Eclipse on your computer. The process for doing this varies depending on your operating system.
- After it launches, you will first be asked to choose a **workspace folder** – this is the folder that will contain all of the files you are working on. On a Windows machine, your default workspace folder will be in **c:\Users\<username>\eclipse-workspace**; the path will be similar on other operating systems. You can leave it as this if you want, or you can change it to a different directory if you prefer, and you can tick the box to use this workspace as the default if you do not want to be asked again. Press **Launch** to launch Eclipse in the selected workspace.


Eclipse IDE Launcher

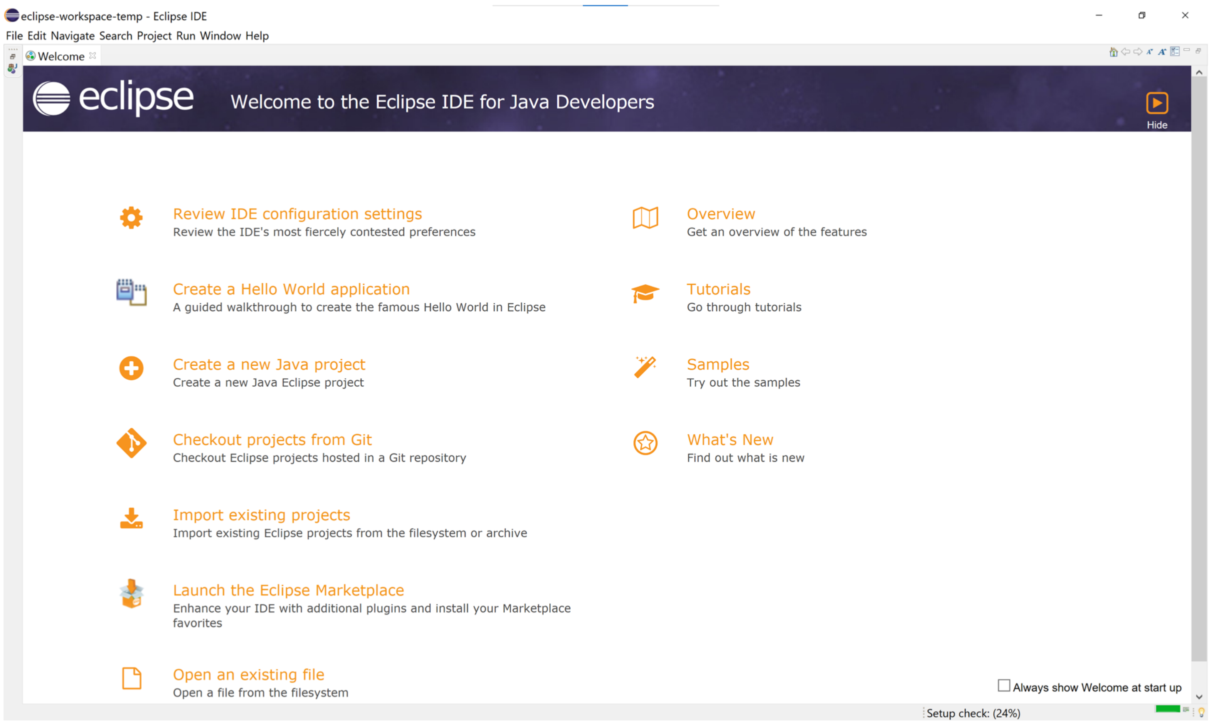
Select a directory as workspace

Eclipse IDE uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\Users\mefos\eclipse-workspace

☐ Use this as the default and do not ask again

- Eclipse takes a few seconds to start up. You will see Eclipse's 'welcome' screen appear - see below. You can make sure that the “Always show Welcome at start up” box is un-ticked (it is un-ticked by default), to ensure that you go straight to the workspace when you open it later on.



eclipse-workspace-temp - Eclipse IDE

File Edit Navigate Search Project Run Window Help

Welcome

eclipse Welcome to the Eclipse IDE for Java Developers

Hide

Review IDE configuration settings
Review the IDE's most fiercely contested preferences

Create a Hello World application
A guided walkthrough to create the famous Hello World in Eclipse

Create a new Java project
Create a new Java Eclipse project

Checkout projects from Git
Checkout Eclipse projects hosted in a Git repository

Import existing projects
Import existing Eclipse projects from the filesystem or archive

Launch the Eclipse Marketplace
Enhance your IDE with additional plugins and install your Marketplace favorites

Open an existing file
Open a file from the filesystem

Overview
Get an overview of the features

Tutorials
Go through tutorials

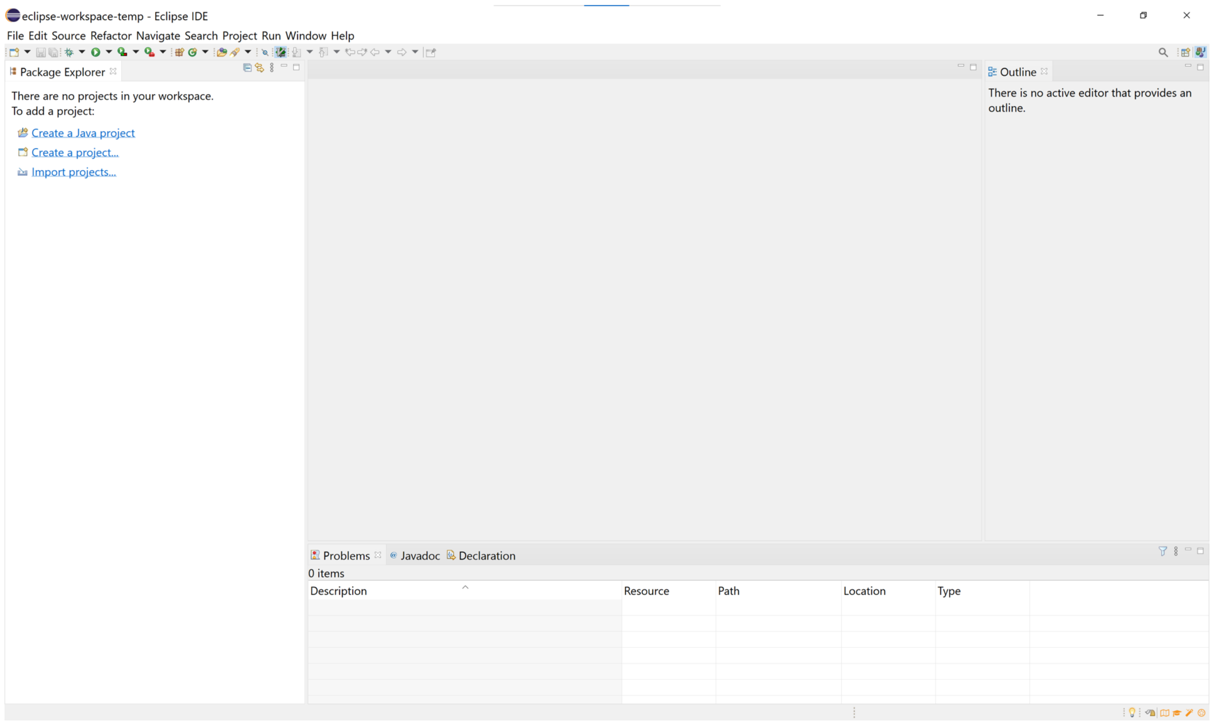
Samples
Try out the samples

What's New
Find out what is new

☐ Always show Welcome at start up

Setup check: (24%)

- Clicking on the “Hide” arrow (arrow in box on top right hand side) will take you to the main IDE. The screenshot below shows the initial appearance of the Java window.



eclipse-workspace-temp - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

There are no projects in your workspace.
To add a project:

Create a Java project
Create a project...
Import projects...

Outline

There is no active editor that provides an outline.

Problems | Javadoc | Declaration

0 items

Description	Resource	Path	Location	Type

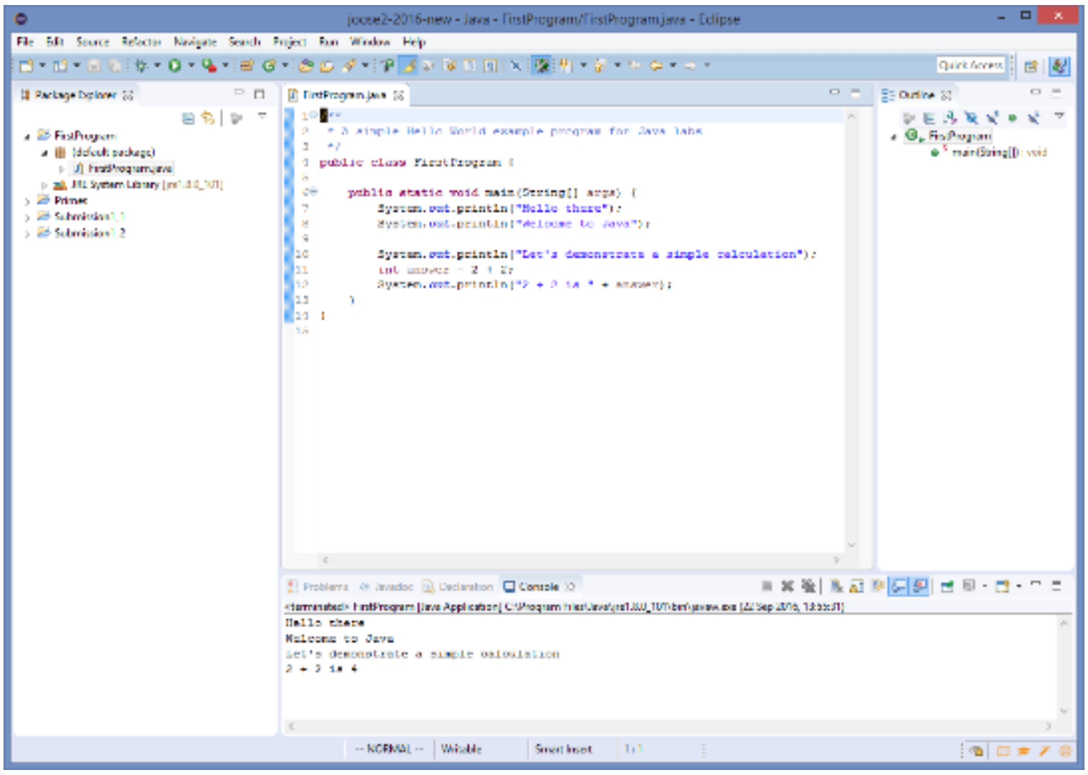
- Each Java program that you work on will form an Eclipse project, and this project has to be created. Each project will have its own folder, the name of the folder necessarily being the same as the name of the project.
- In Eclipse, select **File → New → Java Project** to create a new project, and then give your project an appropriate name (you can leave the rest of the fields unchanged). If it prompts you to create a module-info.java file, you can safely say "No".
- In the **Package Explorer** view (on the left of the Eclipse window), you should now an icon representing the newly created project. Clicking on the expander icon ▶ beside the project name reveals the contents of the project – namely components of the Java Run-Time Environment (JRE) together with the *default package*, which currently contains nothing as you have not created any classes.
- Right click on the project name and choose **New → Class** to create a new Java class. You can give the class whatever name you want, for example **FirstProgram** – if you want, you can also set any other features of the class, such as defining it as abstract or final or giving it an access modifier.
    - Here is some code that you can enter into your **FirstProgram** class if you want to play around with Java:

```
public class FirstProgram {

  public static void main(String[] args) {
    System.out.println("Hello there");
    System.out.println("Welcome to Java");

    System.out.println("Let's demonstrate a simple calculation");
    int answer = 2 + 2;
    System.out.println("2 + 2 is " + answer);
  }
}
```

- Now click on the **Editor view** to activate it. Whenever Eclipse detects an error in the java file displayed in the editor view, this is indicated by red underlines in the text, and red crosses to the left (and possibly also the right) of the view. Hovering the cursor over a marker throws up an error message – see below. This feature of Eclipse takes a bit of getting used to; it can be annoying as the compiler often does not give you time to finish typing a line before throwing up some spurious error indicator, but in general this can be a very useful feature once you get used to it.
- To run a program, you need to make sure that the Java class has a **main** method with the correct signature (as in **FirstProgram** above). To run such a class, first make sure that the **Editor view** is active (as indicated by a blue border). Then select the Menu option **Run → Run as → Java application**, or click the green 'Play' button in the toolbar. You should see the output from the program in the **Console view**, at the foot of the Eclipse window.



```
/**
 * A simple Hello World example program for Java labs
 */
public class FirstProgram {

	public static void main(String[] args) {
		System.out.println("Hello there");
		System.out.println("Welcome to Java");

		System.out.println("Let's demonstrate a simple calculation");
		int answer = 2 + 2;
		System.out.println("2 + 2 is " + answer);
	}
}
```

```
<terminated> FirstProgram [Java Application] C:\Program Files\Java\jre1.8.0_101\bin\javaw.exe (22 Sep 2016, 12:55:31)
Hello there
Welcome to Java
Let's demonstrate a simple calculation
2 + 2 is 4
```

- Now introduce an error into the Java code. For example, delete one of the letters from the word `static`. You will see a red cross appear on the extreme left of the Editor view, and moving the cursor over this marker reveals a helpful error message. (Error messages are not always as immediately helpful as this.) Experiment with some further errors – for example, try deleting punctuation marks like semicolons or curly brackets to see what errors are indicated.
- You may have noticed that when the cursor hovers over an identifier (in the **Editor** view) information about that identifier is displayed. You will discover a variety of useful features of the Eclipse editor in due course.
- You are now ready to work on the programming exercises in the lab!
-
- A postscript: These slides focus on getting started with Eclipse, which is obviously primary, but once you are comfortable with the basics of writing and running code in Eclipse, I suggest you (at some time of your choosing) also get used to debugging code in Eclipse — using its *debug mode.* It will be worthwhile! Online tutorials include this one:
    - https://www.baeldung.com/eclipse-debugging