EECS 2030: Lab 0

(1 % of the final grade, may be done in groups of up to three students)

Motivation

This lab will allow you to practice using an Eclipse IDE, and review some basic Java features.

Part 1: Getting Started

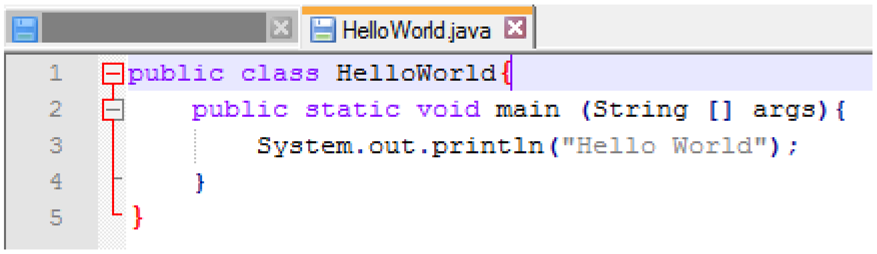
Prerequisites: JDK and Eclipse

Developing Java applications requires a Java Development Kit (JDK). In this course we will also use Eclipse – an integrated development environment. Note that some JDK must be installed *before* one can install and use Eclipse. Please follow the installation instructions online[1]. As a backup, you may also use the EECS lab remote access (the link is posted on the course page).

Java: Command-Line

One way to compile and run Java code is to use a command line. For that, one writes the Java code using any plain-text editor (preferably an editor capable of at least highlighting the language's syntax, such as Visual Studio Code, Atom, Notepad++, then compiles and runs the application that was compiled to byte-codes.

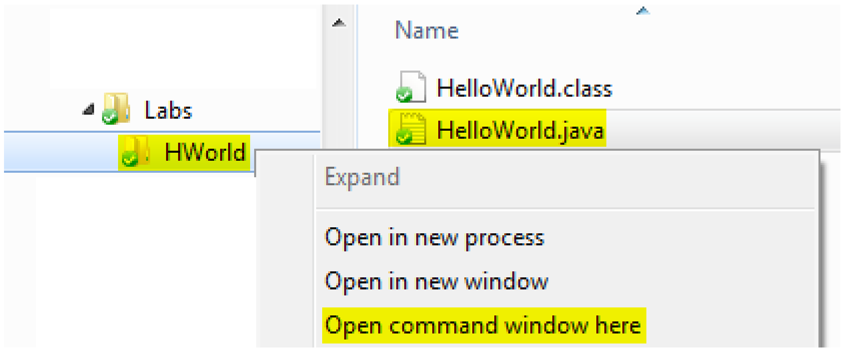
Let's assume the following content is saved to a file called *HelloWorld.java*:



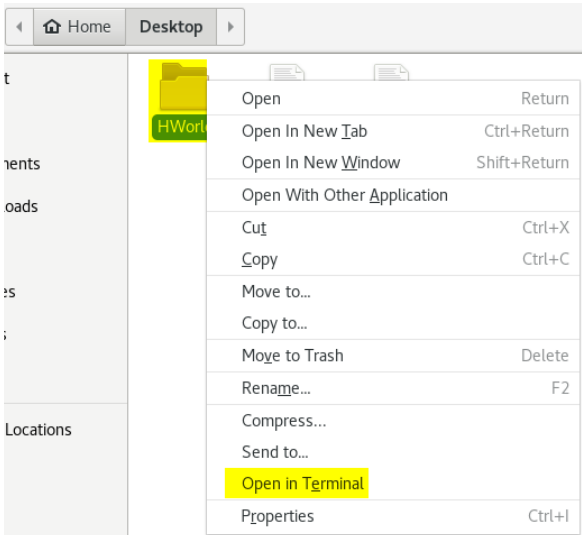
```
public class HelloWorld{
    public static void main (String [] args){
        System.out.println("Hello World");
    }
}
```

In order to open the command line tool in the location where the file had been saved (and not have to use a *cd* command), in Windows, one can right-click on the directory containing the file while holding the Shift key:

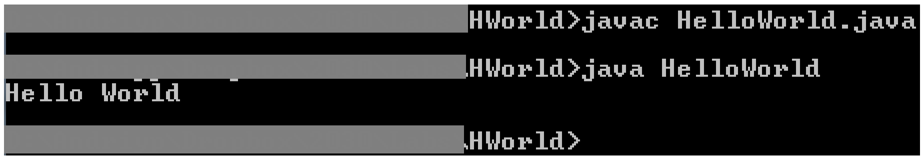
[1] https://wiki.eclipse.org/Eclipse/Installation



Similar context-menu choices exist for other operating systems, e.g., in Linux (CentOS):



Then, the following commands compile and run the code (the full path is greyed out):



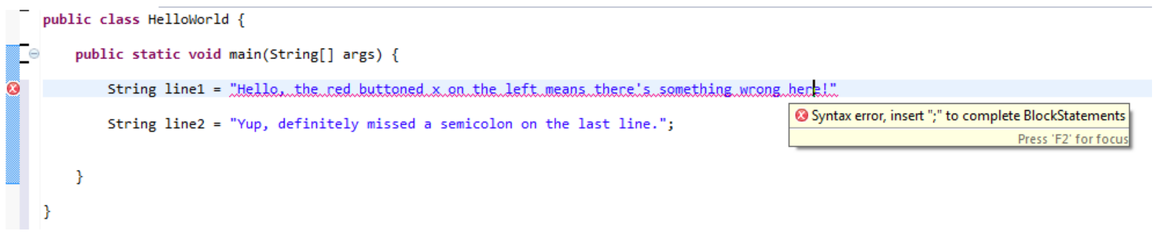
```
HWorld>javac HelloWorld.java

HWorld>java HelloWorld
Hello World

HWorld>
```

In case there are compilation or runtime errors, various error messages will be printed. Try introducing errors and see what kind of messages you receive.

Java: Eclipse

Eclipse is an IDE (an Integrated Development Environment), meaning that it is a program *made* to provide you with all the tools you need to code, wrapped up in a nice user interface. Here are some of the top reasons for using an IDE:

Spot compiler errors quickly - if you've made some small mistake like forgetting a semicolon, or spelling a variable type wrong, the IDE will spot it quickly and let you know:

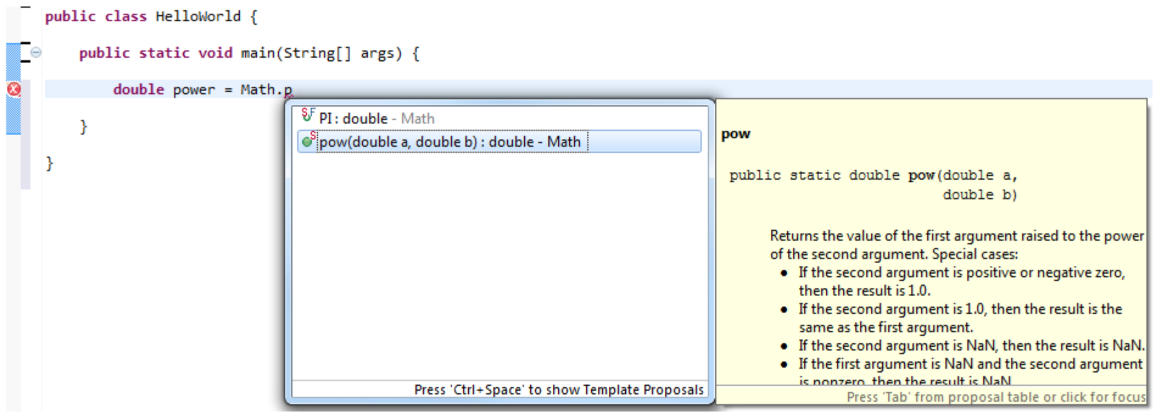


```
public class HelloWorld {

    public static void main(String[] args) {

        String line1 = "Hello, the red buttoned x on the left means there's something wrong here!"

        String line2 = "Yup, definitely missed a semicolon on the last line.";

    }

}
```

Syntax error, insert ";" to complete BlockStatements

Press 'F2' for focus

Autocomplete/Content Assist - this feature will save you time and effort by suggesting methods (and their parameters) and types. See below for an example:



```
public class HelloWorld {

    public static void main(String[] args) {

        double power = Math.p

    }

}
```

PI : double - Math

pow(double a, double b) : double - Math

Press 'Ctrl+Space' to show Template Proposals

pow

```
public static double pow(double a,
                         double b)
```

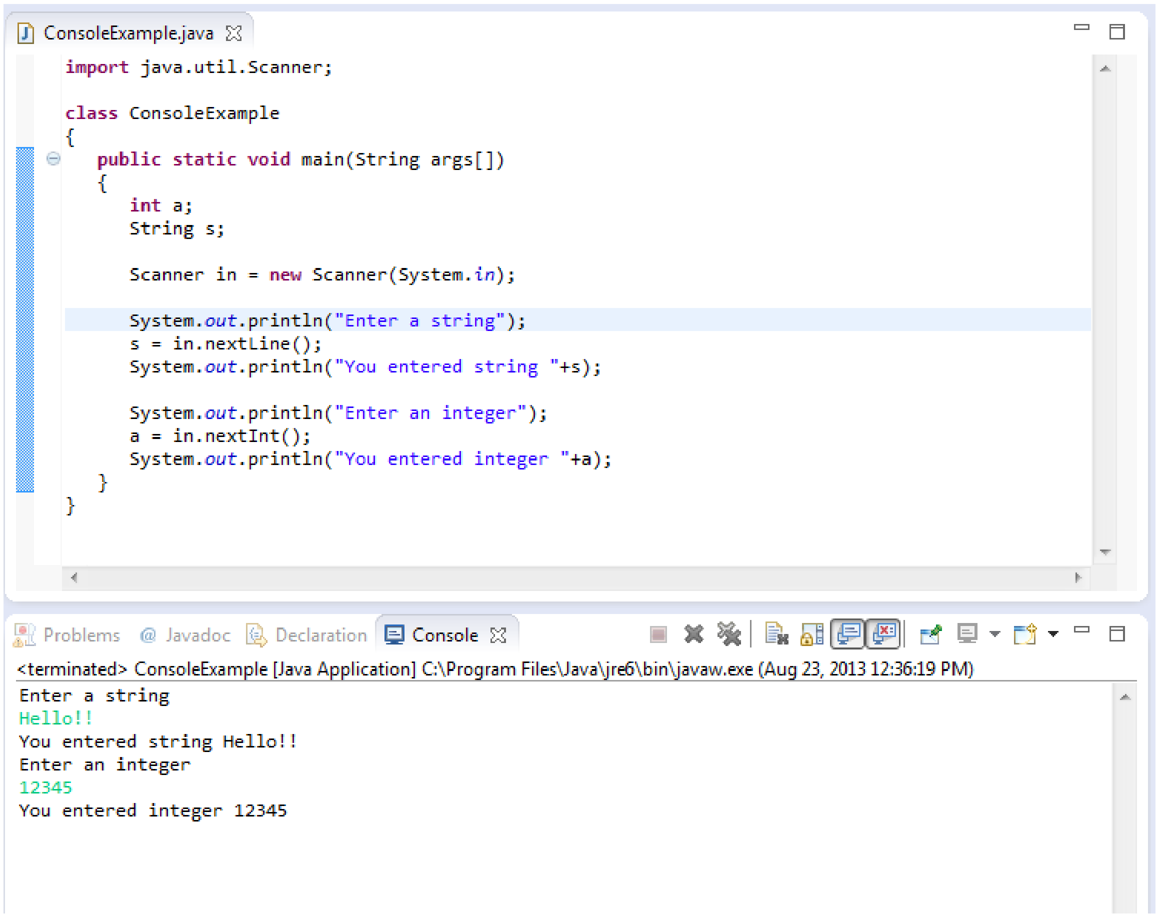
Returns the value of the first argument raised to the power of the second argument. Special cases:

- If the second argument is positive or negative zero, then the result is 1.0.
- If the second argument is 1.0, then the result is the same as the first argument.
- If the second argument is NaN, then the result is NaN.
- If the first argument is NaN and the second argument is nonzero, then the result is NaN.

Press 'Tab' from proposal table or click for focus

Debug Mode - allows you to watch your variables and source code at any point in your program so you can find out exactly where your program is going wrong.

Built in console and compiler access - saves you the time from going back and forth from editor to console and vice versa.

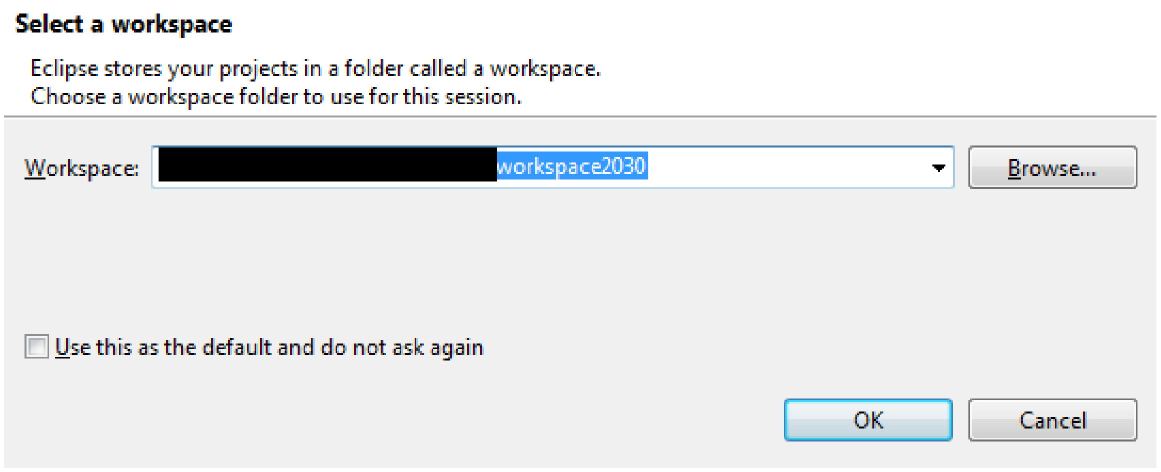


```
import java.util.Scanner;

class ConsoleExample
{
    public static void main(String args[])
    {
        int a;
        String s;

        Scanner in = new Scanner(System.in);

        System.out.println("Enter a string");
        s = in.nextLine();
        System.out.println("You entered string "+s);

        System.out.println("Enter an integer");
        a = in.nextInt();
        System.out.println("You entered integer "+a);
    }
}
```

```
<terminated> ConsoleExample [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (Aug 23, 2013 12:36:19 PM)
Enter a string
Hello!!
You entered string Hello!!
Enter an integer
12345
You entered integer 12345
```

A sample program in Eclipse:

- Start Eclipse
  The following window appears (with some variations)



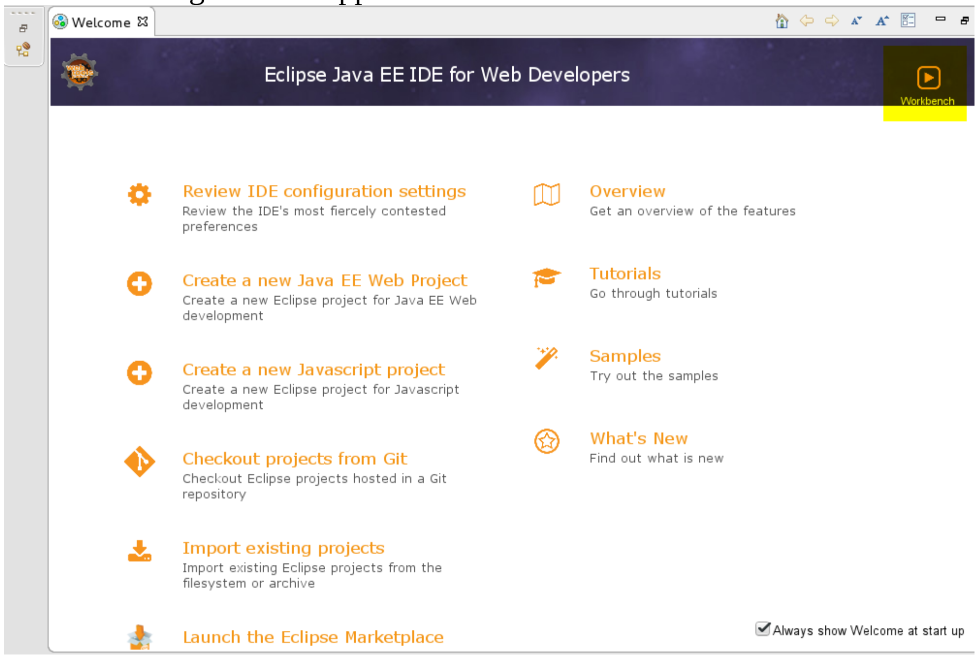
Select a workspace

Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: [redacted]workspace2030

  you will be asked to select a workspace. This is important! This is the root directory to which all of your programs will be saved to, so make sure you know where it is. One may click *Browse...* and create a directory called **eecs2030, workspace2030** –use any directory name that you like (including the default one **workspace**); try to avoid spaces in the directory names.

- If the following window appears:



Eclipse Java EE IDE for Web Developers

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

Create a new Java EE Web Project
Create a new Eclipse project for Java EE Web development

Create a new Javascript project
Create a new Eclipse project for Javascript development

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

Overview
Get an overview of the features

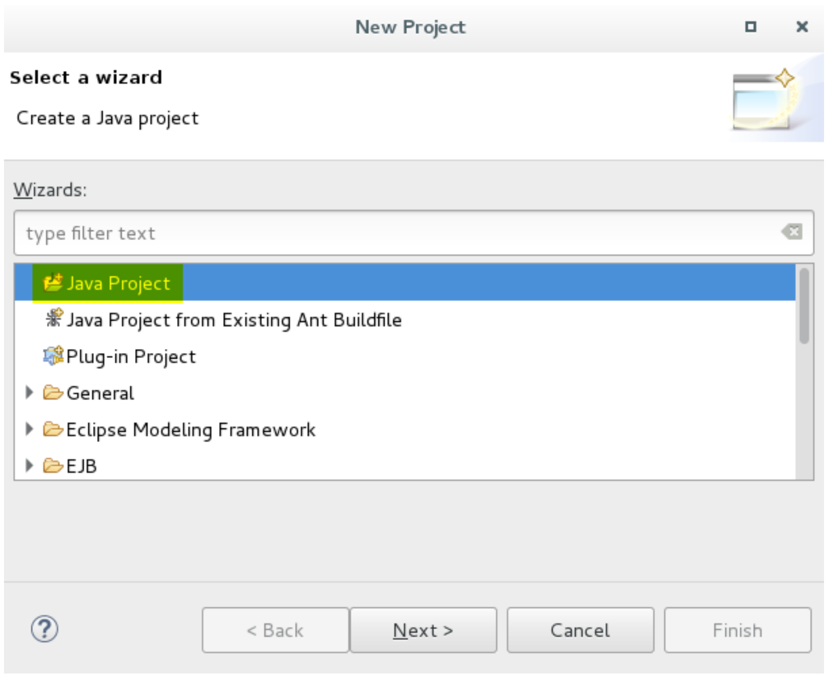
Tutorials
Go through tutorials

Samples
Try out the samples

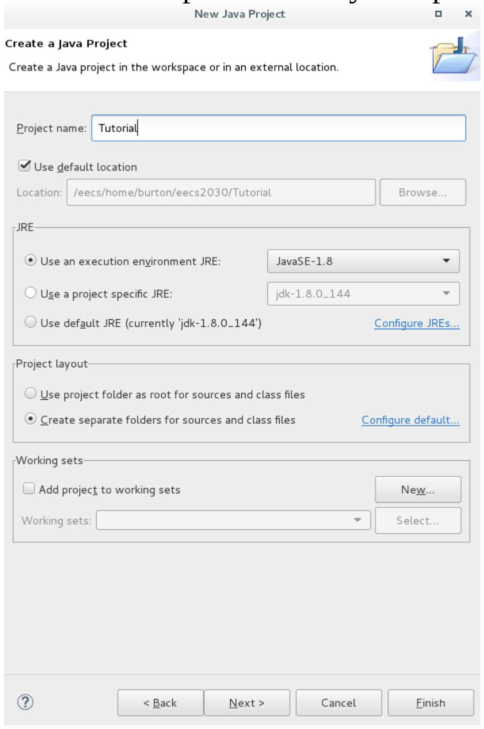
What's New
Find out what is new

  click on the orange **Workbench** button found near the top right corner.

- create a Java Project. You can do this by doing: *File* -> *New...* -> *Project....* The project wizard looks like this - click on *Java Project*.



- Now, you will see a window like the one shown. Give your project a name, e.g., 'Tutorial'. You can name your project whatever you like, but ***avoid using spaces in the project name*** because this complicates navigating the directory structure of your project. Note that doing this now creates a directory in your previously created workspace directory. Keep the default settings and click finish.



Project name: Tutorial

[✓] Use default location

Location: /eecs/home/burton/eecs2030/Tutorial

[●] Use an execution environment JRE: JavaSE-1.8

[ ] Use a project specific JRE: jdk-1.8.0_144

[ ] Use default JRE (currently 'jdk-1.8.0_144')

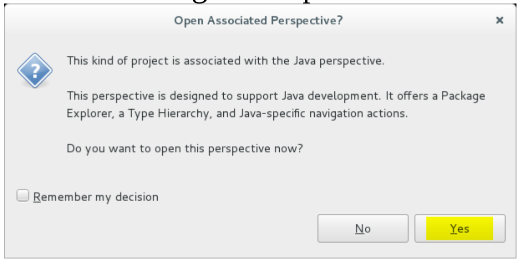
[ ] Use project folder as root for sources and class files

[●] Create separate folders for sources and class files

[ ] Add project to working sets

Working sets:

- If you see a popup window like the one showed here. Click *Yes* if that is the case. This will configure eclipse so that it enables Java specific features.



Open Associated Perspective?

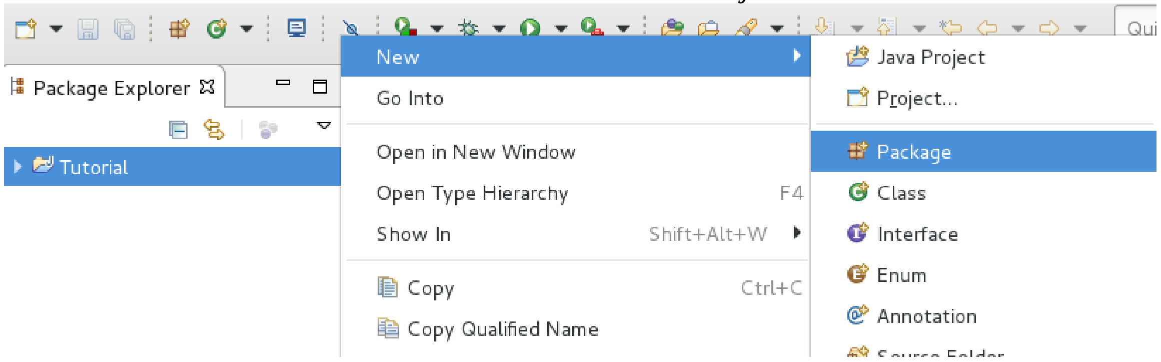
This kind of project is associated with the Java perspective.

This perspective is designed to support Java development. It offers a Package Explorer, a Type Hierarchy, and Java-specific navigation actions.

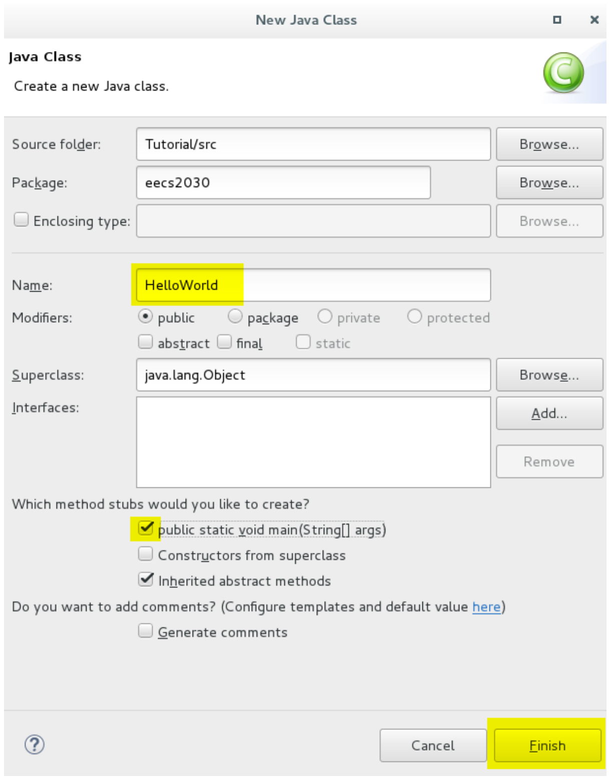
Do you want to open this perspective now?

☐ Remember my decision

- create a package. To do this, right-click the new project you just created that will now appear in the *Package Explorer* on the left side, and then click *New -> Package*. Name it **eecs2030**. Now if you look in the Package Explorer, our **eecs2030** package is under *Tutorial -> src -> eecs2030*. *src* is the source folder. Hence, when it comes time to submit your files and you want to locate your source file, it will be in *workspace->Project->src->package->file*. So in our case, it would be: *eecs2030/Tutorial/src/eecs2030/HelloWorld.java*



- create a HelloWorld.java. You can do this by right-clicking the package, then clicking *New*, and then *Class*. Enter the name of your class, in this case **HelloWorld**, and check the *public static void main(String[] args)* box - this will create the main method in your class for you. Our simple program does not inherit from any other program, so the other two boxes don't really matter.



New Java Class

Java Class

Create a new Java class.

Source folder: Tutorial/src

Package: eecs2030

☐ Enclosing type:

Name: HelloWorld

Modifiers: ● public ○ package ○ private ○ protected

☐ abstract ☐ final ☐ static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

☑ public static void main(String[] args)

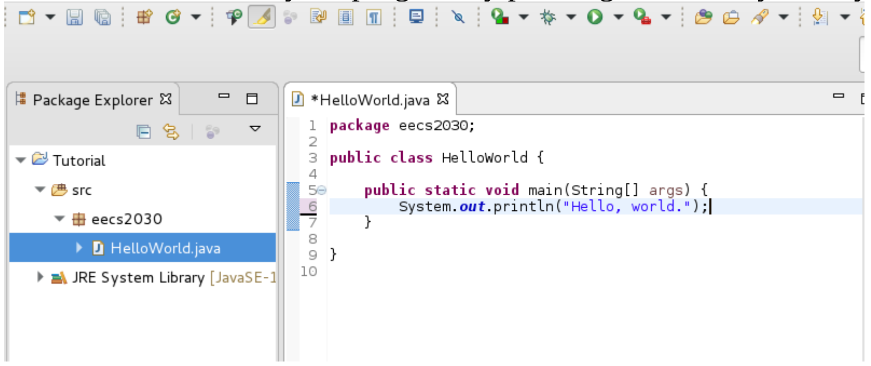
☐ Constructors from superclass

☑ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

☐ Generate comments

- Add **System.out.print("Hello, world.");** into the main body in the TODO section and then save your program by pressing *Ctrl+s* on your keyboard.



```java
package eecs2030;

public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello, world.");
    }

}
```

- Now you can run your program by either clicking the green run button on the toolbar, or by right-clicking your package in the package explorer and selecting *Run As...->Java Application* (find a keyboard shortcut for doing it quickly!). Note that the output appears in the console window at the bottom.

- Submit that **HelloWorld.java** file via Moodle and continue with Part 2.