CMSC206 Data Structures (Fall 2014: Deepak Kumar)
Lab#1: Java & Eclipse (An Introduction)

Objective: Today, you will get your first introduction to writing simple *Java* programs. Also, you will learn how to use the *Eclipse IDE* to write, compile, and run Java programs. Beyond the basic "Hello World" program, you will also learn how to do *simple dialog-based interaction*. Also, you will learn how use the *Java Math class* and to do some simple computations.

Hello Java

The program shown below prints out a simple message in the *Java Console*:

```
public class HelloWorld {

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

} // class HelloWorld
```

The following steps are required to write, compile, and run the program:

1. Enter the code and save it in a file called, `HelloWorld.java`
2. *Compile* it using the Java compiler:

   ```
   >java HelloWorld.java
   ```

   This creates a file called `HelloWorld.class`

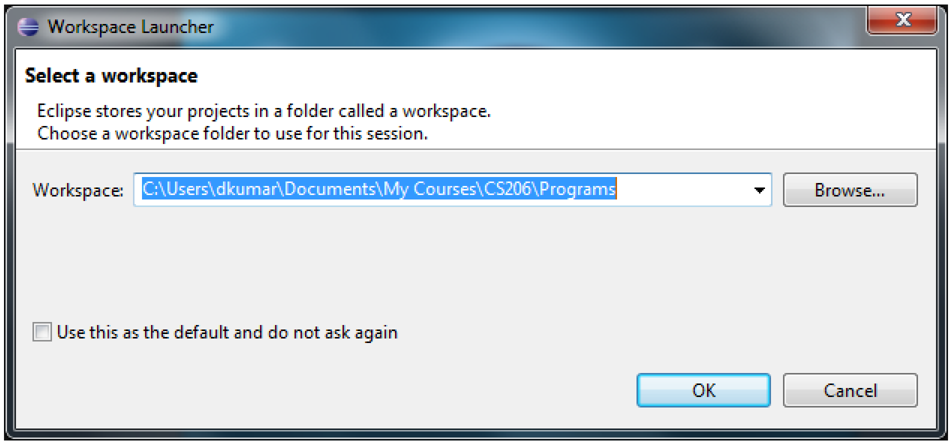
3. *Run* the program by using the *Java Virtual Machine (JVM)*:

   ```
   > java HelloWorld
   Hello World
   ```

This approach requires you to have an editor (like Notepad) to create and save your program. Then, using the command line you have to enter the command to compile and run the program. This will be shown to you in the lab. As you can see, this can get tedious even for the shortest of programs.

Hello Eclipse!

The *Eclipse Integrated Development Environment (IDE)* enables you to edit, compile, and run your programs in a more organized and structured manner. In fact, Eclipse is a powerful software development environment and may appear too daunting at the other end of the scale at first. Fear not, we will learn gradually and ease into it. Let's get started. Find Eclipse on your computer and start the application. You will see a window shown below:



Select a workspace

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

Workspace: C:\Users\dkumar\Documents\My Courses\CS206\Programs

☐ Use this as the default and do not ask again

Figure 1: Eclipse will ask you for your workspace.

When you start Eclipse, it will ask you to specify your *workspace*. This is the folder where you will be saving all your Java programs. Think about a good place (for instance, I save my Java programs in a folder called MyCourses\CS206\Programs). Yours could be on your USB drive, or a folder in your account. Browse to your programs folder (create one if you do not have one yet), and then click the OK button. You will see the screen shown below:

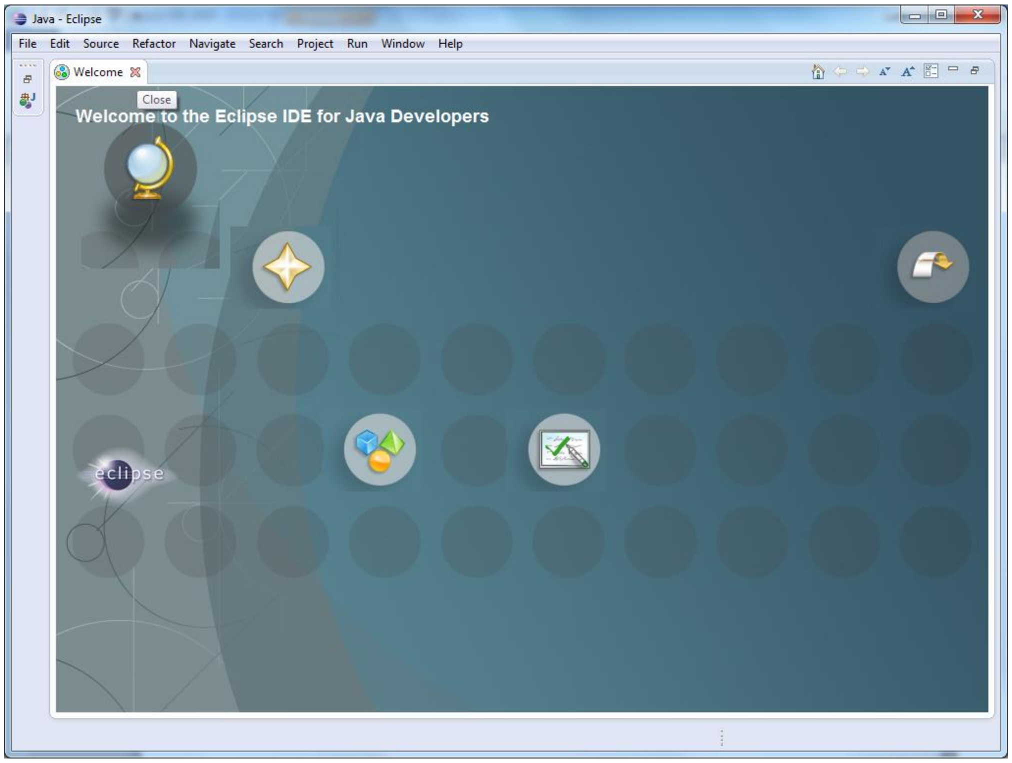

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Welcome
Close
Welcome to the Eclipse IDE for Java Developers
eclipse

Figure 2: The Eclipse Start Screen

The welcome screen will appear for first time users. If you mouse over the bubbles, they will reveal various places you could go to get more information. For now, just close the folder by clicking on the X in the Welcome tab in the top-left part of the screen (just above the `Welcome to the Eclipse IDE for Java Developers` message).

The Java Perspective

Once you close the Welcome screen you will see the screen shown below. This is the *Java Perspective*. Go ahead and maximize the application window. This is essentially your home base for doing all your Java work. Let's spend some time getting familiar with it. Then we can write and run our first program.

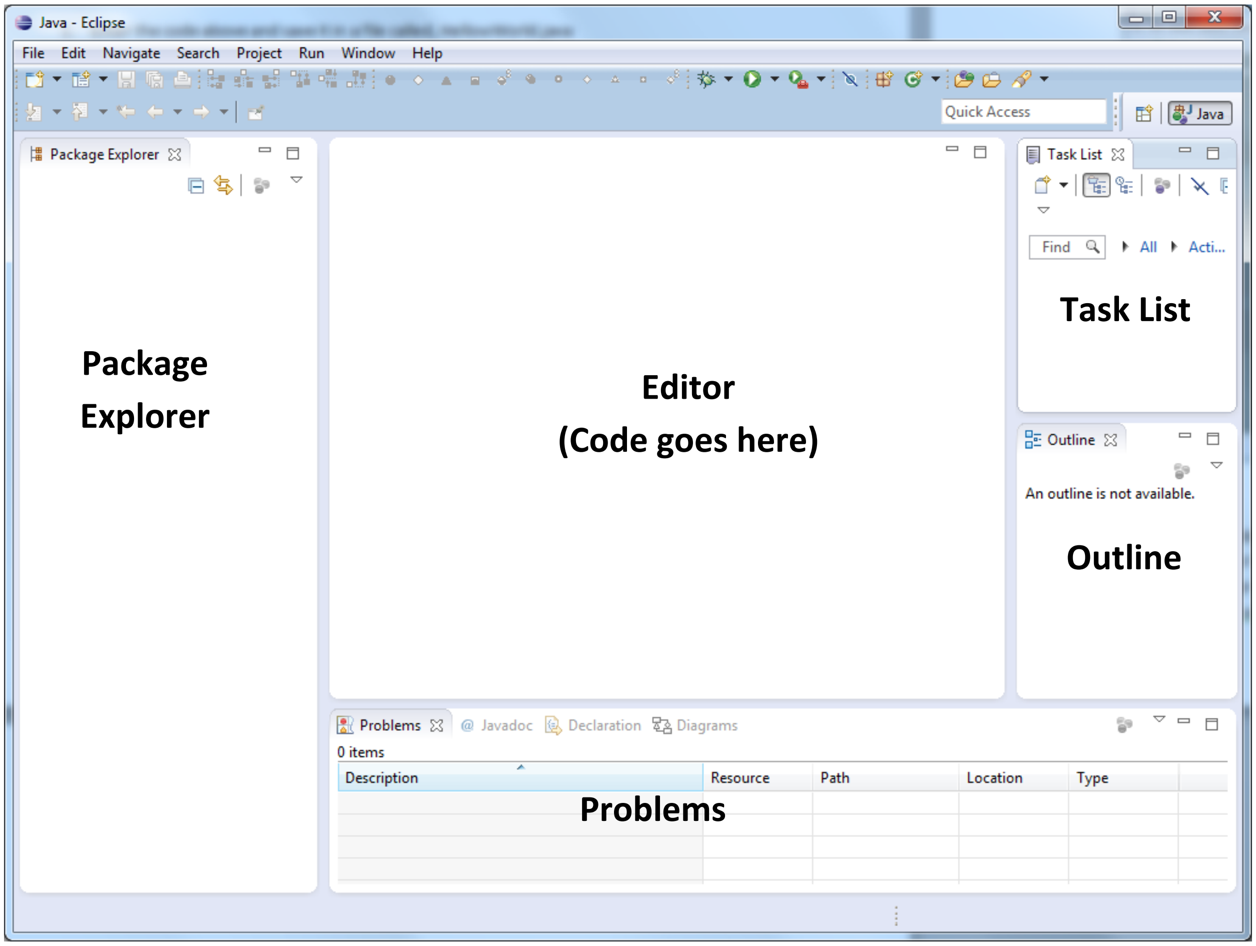


Figure 3: The Java perspective

Apart from the Menu options/icons, the application is divided into 5 major panes/sections/views (see above):

1. **Package Explorer:** This will show all your project files
2. **Editor:** This is where you edit/write code
3. **Problems:** This is where problems are reported. Also notice tabs for **Javadoc**, **Declaration**, **Diagrams**, etc. We will visit these later.
4. **Task List:** TODO tasks listed here. You can close this. We will not need this for some time.
5. **Outline:** This will show an outline of your code (to be used later).

For now, just focus on the Menu options, the Package Explorer, and the Editor panes. Let's create a project to write and run our Hello World program.

The Hello World Program

From the File Menu, select New and from the second menu that pops up, select *Java Project* (see below):

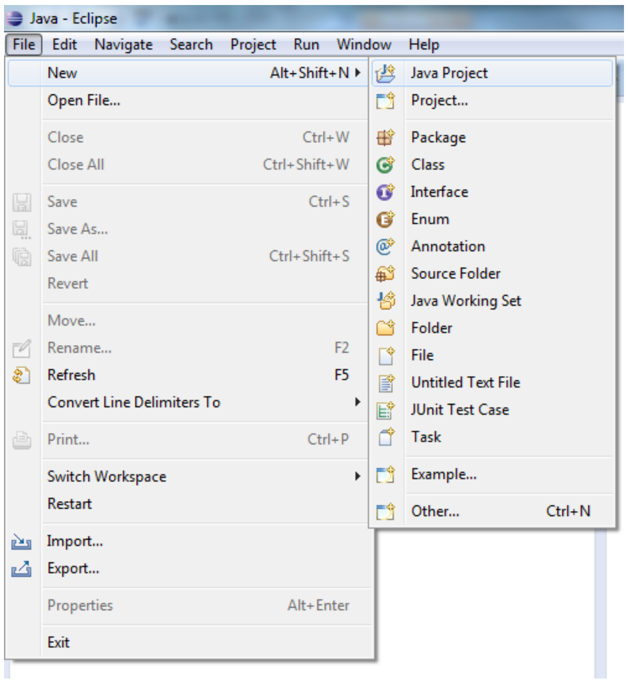


Figure 4: Creating a new Java project.

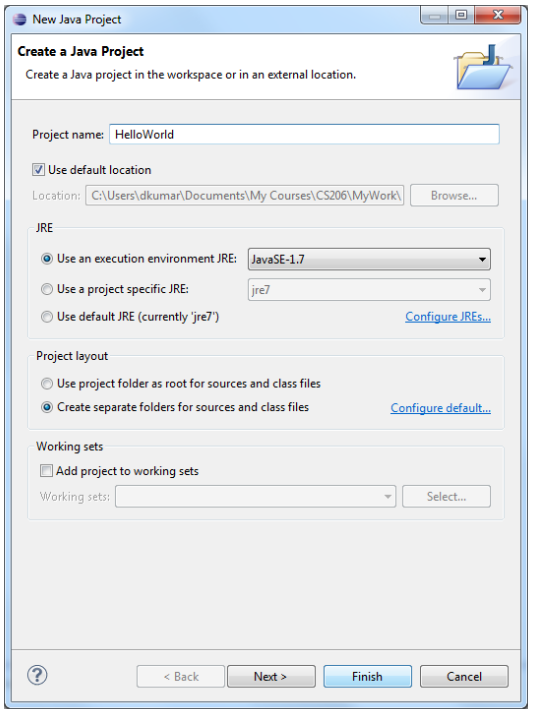
Every Java program you write will be part of a Project. After you select the New Project option, you will get a pop up window (see Figure 5). This is a form to fill out some details about the kind of project you are creating.

All you have to do here is enter the name for your project in the Project name field. We have chosen HelloWorld as you can see.

Click Finish to go ahead and create the project, now named HelloWorld.

You will notice that the HelloWorld project is now listed in the Project Explorer pane. Go ahead and click on the little triangle to expose the contents of the project. This is shown in Figure 6.

Essentially what Eclipse has done is created a folder called HelloWorld in your specified workspace folder. In it, it has placed another folder called **src** which is currently empty. **src** is short for *Source Code* and this is where the code for your program needs to go.


New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: HelloWorld

[✓] Use default location

Location: C:\Users\dkumar\Documents\My Courses\CS206\MyWork\ Browse...

JRE

[●] Use an execution environment JRE: JavaSE-1.7

[ ] Use a project specific JRE: jre7

[ ] Use default JRE (currently 'jre7') Configure JREs...

Project layout

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

[●] Create separate folders for sources and class files Configure default...

Working sets

[ ] Add project to working sets

Working sets: Select...

< Back Next > Finish Cancel

Figure 5: The New Java Project Form

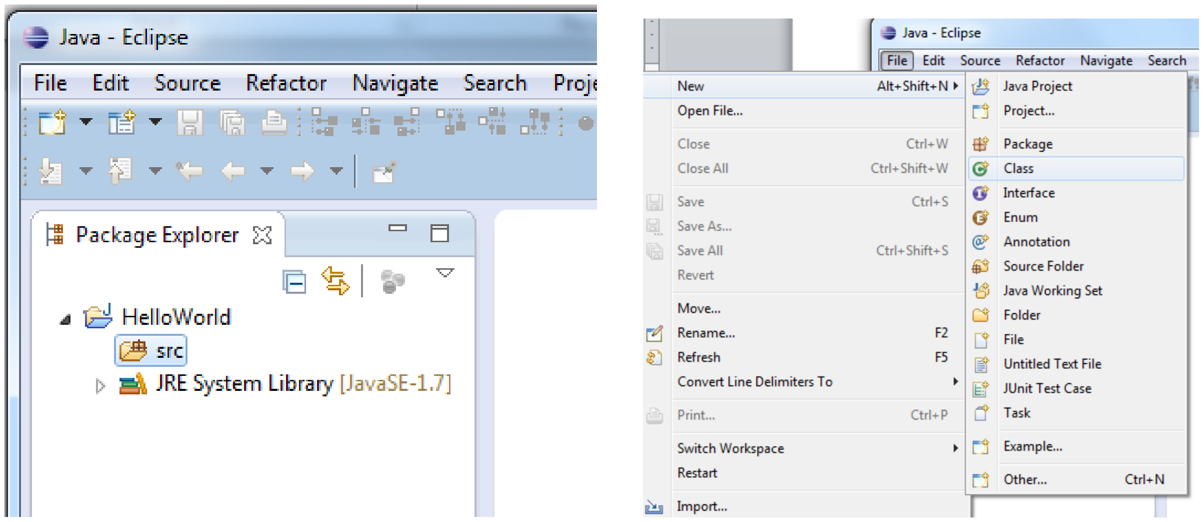


Figure 6: Looking at the HelloWorld Project, and creating a new class.

Next, select the File Menu again, followed by New, and this time select the Class option (see Figure 6). Another form window will pop up as shown in Figure 7.

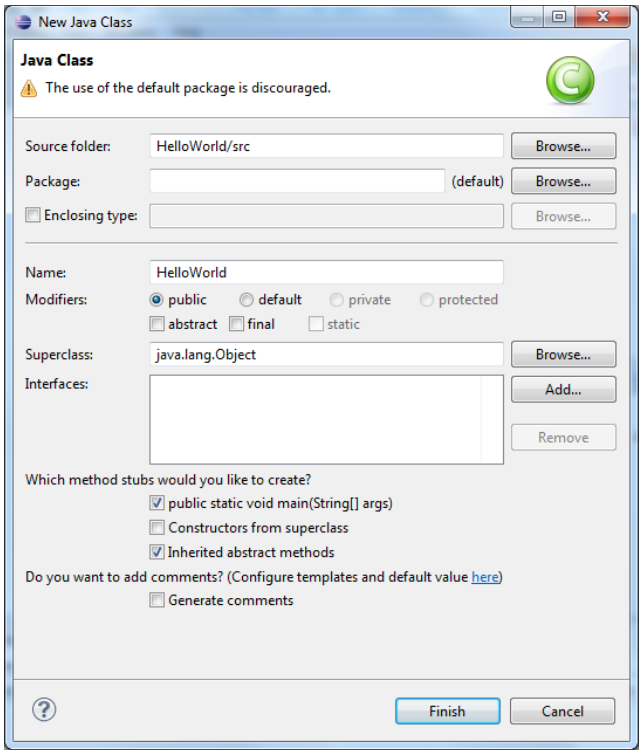
Again, enter the name of the class (HelloWorld) in the Name field of the form.

Also, make sure the little box against `public static void main(String[] args)` is checked.

You do not need to do anything else, but be sure that your form matches with the one shown.

Next, click Finish.

You will notice that Eclipse has created what looks like most of our HelloWorld Java program!



New Java Class

Java Class

The use of the default package is discouraged.

Source folder:	HelloWorld/src	Browse...
Package:	(default)	Browse...
☐ Enclosing type:		Browse...
Name:	HelloWorld	
Modifiers:	[✓ public] default private protected	
	☐ abstract ☐ final ☐ static	
Superclass:	java.lang.Object	Browse...
Interfaces:		Add... Remove

Which method stubs would you like to create?

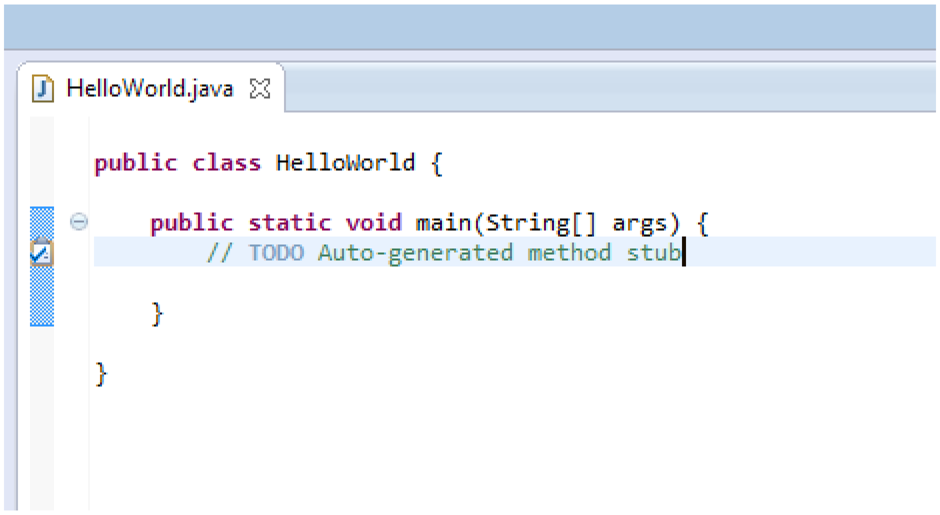
☑ public static void main(String[] args)
☐ Constructors from superclass
☑ Inherited abstract methods

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

☐ Generate comments

Figure 7: The new class form.

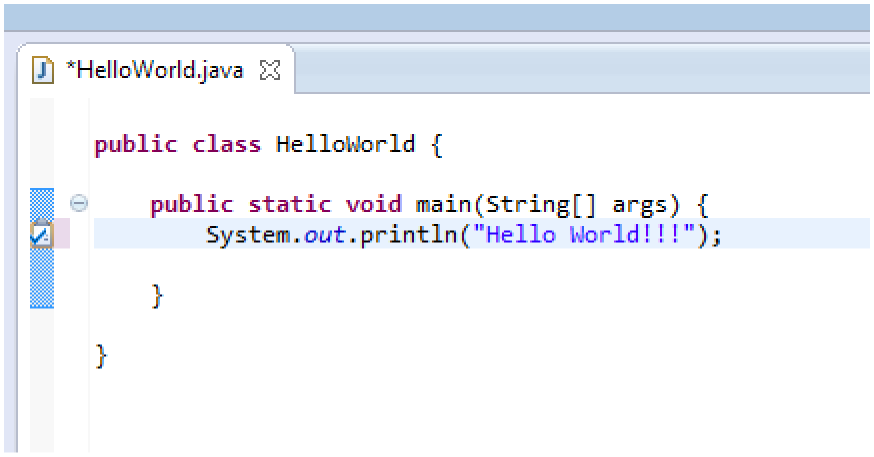
```
public class HelloWorld {

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

} // class HelloWorld
```



```
public class HelloWorld {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }

}
```

Figure 8: The HelloWorld program template.

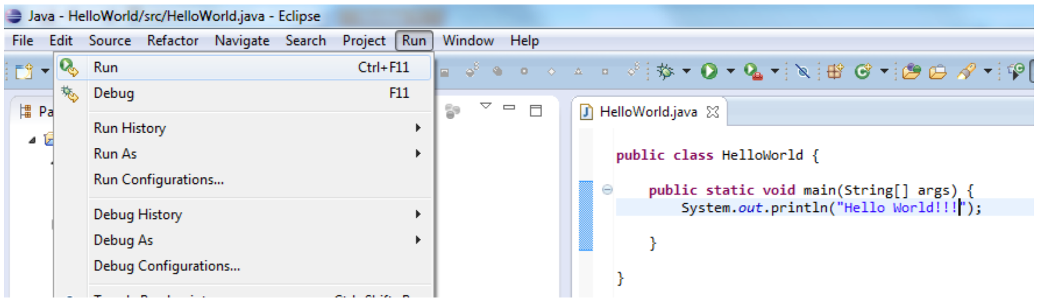
Eclipse has created, based on the information you provided in the previous form, a template Java program for you (see Figure 8). Now, all you have to do is delete the TODO comment stub and add the `System.out.println(…)` command in the `main()` method. This is shown below:



```
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello World!!!");

    }

}
```

Figure 9: Your First Java Program!

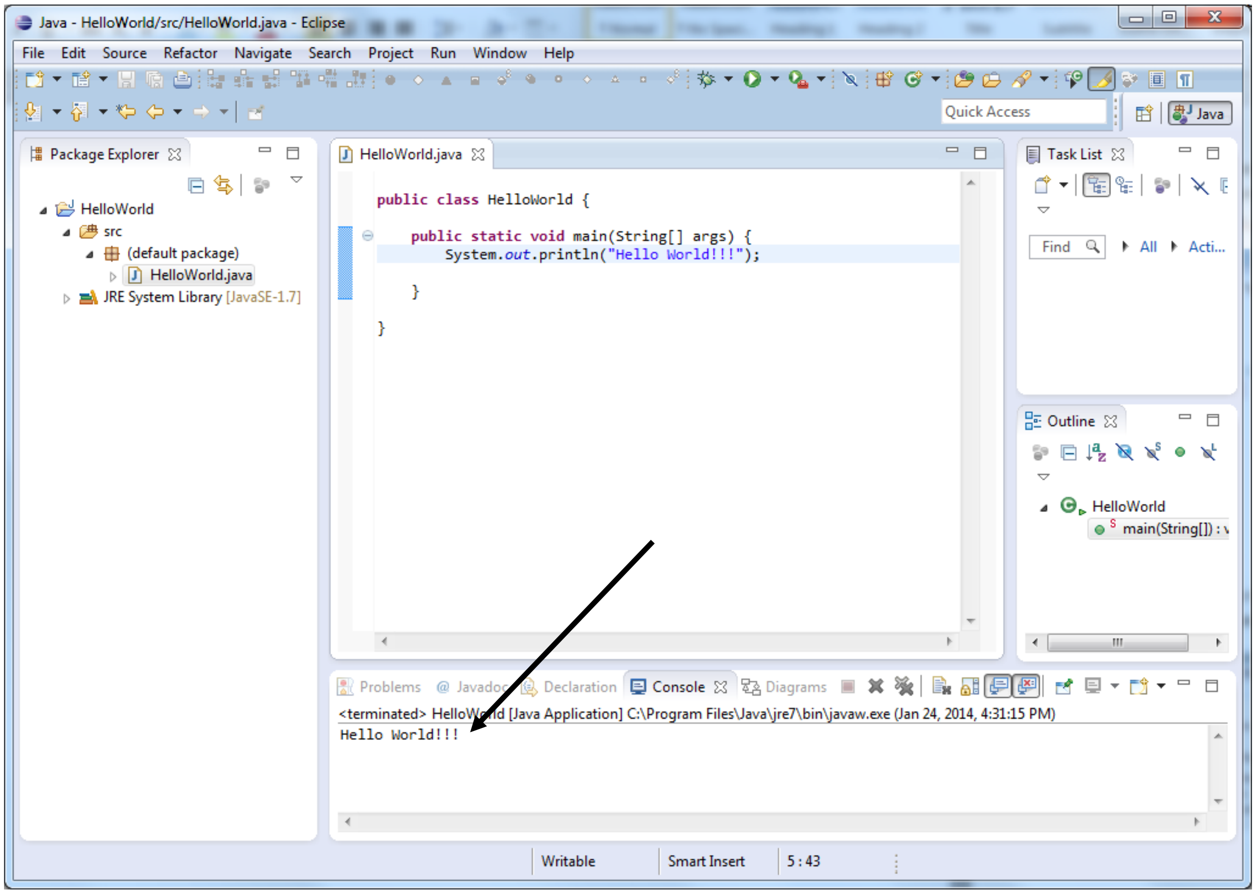
Now you are ready to run the program. But first, be sure to *Save* the program (from the File Menu select the Save option). From the Run Menu, select the Run option (see Figure 10).



```
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello World!!!");

    }

}
```

Figure 10: Select the Run option from the File Menu

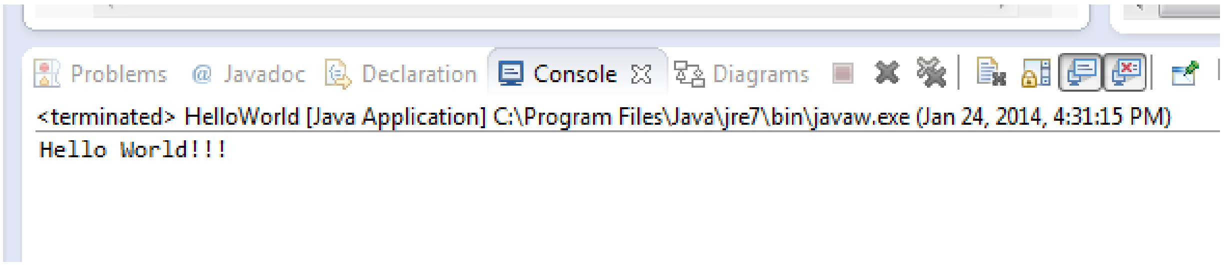
Stuff happens. Very fast! If you focus on the pane below the Editor window, a Console window is shown with the output of your program (see Figure 11):



```
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello World!!!");

    }

}
```

```
<terminated> HelloWorld [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Jan 24, 2014, 4:31:15 PM)
Hello World!!!
```

Figure 11: You just ran the Hello World program!



```
<terminated> HelloWorld [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Jan 24, 2014, 4:31:15 PM)
Hello World!!!
```

Figure 12: The result of the Hello World program.

Make some changes to the program and run it a few more times. Also, notice the icons bar below the Menu Bar. Many of the Menu items are available here so you can just click and go. Your instructor will give you an overview of these during the lab session.

Congratulations, now you are a Java & Eclipse newbie!!

After completing, be sure to say YES to Item#2 on the lab report.

Dialog Based Interaction

Modify the Hello World program to look like the one shown below:

```
import javax.swing.JOptionPane;

public class HelloWorld {

	public static void main(String[] args) {
		JOptionPane.showMessageDialog(null, "Hello World!!!");

	}
}
```

Your instructor will explain the details of the dialog boxes in the lab session. Run the program. This time you will see a pop up window shown below:

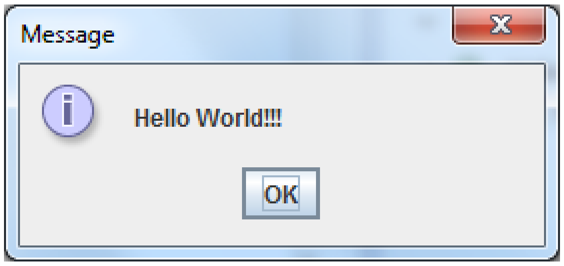


Figure 13: Hello World in a pop-up window!

Next, try the code below:

```
import javax.swing.JOptionPane;

public class HelloWorld {

	public static void main(String[] args) {
		String name = JOptionPane.showInputDialog("Enter your name:");
		JOptionPane.showMessageDialog(null, "Hello " + name + "!!!");
	}
}
```

Run it. Enter your name and Click OK.

Now we're having fun!

After completing, be sure to say YES to Item#3 on the lab report.

The Math class

Section A.4 in the Appendix of your text describes the Java Math class. It defines a number of useful mathematical functions. For example, you can compute the logarithm of a number, various trigonometric functions, etc. We will next learn how to use functions from this class to do some simple computations:

Exercise 1: Write a Java program that prints out the square roots of all the numbers between 1 and 10.

The function

```
double Math.sqrt(n)
```

returns the square root of `n`. The program shown below uses it to solve the exercise:

```
public class Squares {
      public static void main(String[] args) {
            for (int i=1; i <= 10; i++) {
                  System.out.println(i + "\t" + Math.sqrt(i));
            }
      }
}
```

Go ahead, create a new project, enter the program shown and observe the output (shown below):

```
1       1.0
2       1.4142135623730951
3       1.7320508075688772
4       2.0
5       2.23606797749979
6       2.449489742783178
7       2.6457513110645907
8       2.8284271247461903
9       3.0
10      3.1622776601683795
```

Notice how we used the TAB character (`\t`) to align the numbers in the second column. Please write and run the above program. After completing, be sure to say YES to Item#4 on the lab report.

Here is another exercise.

Exercise 2: Write a program to compute and print all the perfect squares between 1 and 100. A perfect square is an integer that is a square of another integer and itself. For example, 9 is a perfect square (3x3). As is 25 (5x5).

The program shown below is a slight variation of the one above. Study it. Then enter and run it. You may use a new project or just modify the project from above.

```
public class Squares {

      public static boolean perfectSquare(int n) {
      // Returns true if n is a perfect square, false otherwise
            int sq = (int) Math.sqrt(n);
            return (n == sq * sq);
      }

      public static void main(String[] args) {
            for (int i=1; i <= 100; i++) {
                  if (perfectSquare(i)) {
                        System.out.println(i);
                  }
            }
      }
}
```

Notice that we used a function `perfectSquares()` to tell us if a given number is a square or not. While this is not essential, it is good practice. It also shows you how to write static functions in your main program. Also, notice the type casting of the value returned by `Math.sqrt()` into an integer. Why is this needed?

After completing, be sure to say YES to Item#5 on the lab report.

Lab#1 Recap

This was a great start to your journey into learning Java. Review what we did. Then read **Appendix A** of your text (pages 597 through 517). Practice today's lab again on your own. Just follow along this lab handout and connect what you are doing to the materials you read from the Appendix.

Assignment 2 (Due in class on Tuesday, February 4): Write a Java program to print out all the prime numbers between 1 and 100.

Hint: You will need to write a function `isPrime()` to tell you if a number is prime or not.

What to Hand in: A print out of your Java program along with its output.

After reading, be sure to say YES to Item#4 on the lab report. Write down your reflections as well.

Additional Things to try (Not part of the assignment)

Modify the program to input a number to compute all the primes between 1 and the input number. Further, input two numbers and compute all the primes between the two input numbers.

Notice the function `Math.random()` in the Math class. Write a program to generate 10 random numbers between 1 and 6.