FIRST JAVA PROGRAM (AND SECOND)

The purpose of this lab is to acquaint you with the mechanics of creating, compiling, and running a Java program. This lab contains two parts. To ensure you get full credit, make sure you read this lab carefully and follow the instructions precisely.

Your First Java Program (for this class, anyway)

In this section, you will create a simple Java program using a simple text editor. Then you will compile and run it in a terminal window. Here are the steps in brief. Find detailed instructions below.

1. Create the program by typing it into a text editor and saving it to a file named HelloWorld.java.
2. Compile it by typing "javac HelloWorld.java" in a terminal window.
3. Run (execute) it by typing "java HelloWorld" in a terminal window.

The first step creates the program; the second translates it into a language more suitable for machine execution (and puts the result in a file named HelloWorld.class); the third actually runs the program. The "Hello World" program typically is the first program beginners write for any programming language. It helps you establish a basis for more complicated programs by learning the basics of compiling, executing, and printing output.

Creating a Java Program You can use any text editor you want. Use Notepad if you have no other preference. Type or copy-and-paste the following code into the text editor, replacing "Your Name" and the date with your name and the current date. Make sure the indentation and line breaks are the same as shown. You may have to add the line breaks.

FORMAT OF A JAVA PROGRAM

*The Java programs we will be writing in the first part of COMP B11 consist of a single class and a **main** method. The **main** method is called by the operating system to start the program. Java, class definitions start with the word "class" (possibly preceded by the word "public" or "private"), continues with the name of the class and the character "{" (a left brace or curly bracket), and ends with the character "}" (a right curly bracket). Inside the braces are definitions of methods and variables. A method definition similarly starts with naming information followed by a pair of braces that include the statements of the method. Braces appear inside methods as well, to group statements. Statements within such a group are terminated by semicolons.*

There are three formats for comments in a Java program. One, intended for short comments, starts with two consecutive slash characters and continues to the end of the line:

```
// this is one kind of comment
```

Another comment format starts with the character sequence "/" (slash followed by asterisk); it continues, perhaps over more than one line, and is terminated by the character sequence "*/" (asterisk, slash). Here's an example:*

```
/*
This is a
multiline
comment */
```

Some commenting styles add single asterisks at the start of each line in the comment after the first, to improve readability:

```
/*
* This is a
* multi-
* line comment
*/
```

*The third type of comment is just like the second one, except it starts with "/**". This comment type is used with an auxiliary program called javadoc, which automatically creates documentation for your classes.*

```
/*
* This program prints "Hello World".
*
* @author Hal Mendoza
* Course: COMP B11
* Created: Jan 13, 2013
* Source File: HelloWorld.java
*/

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

Notice the two different styles of comments. The comment at the top (everything above the line that starts with "public class HelloWorld") is known as a Javadoc class comment. You will include something like it in every Java program you turn in. The line that reads "public class HelloWorld" defines the name of the Java class you are creating, and must be the same name as the file in which you save it (with a .java file extension tacked on), so save your file as HelloWorld.java. Now you are ready to compile and run your program.

Compiling a Java Program At first, it might seem to you as though the Java programming language is designed to be best understood by the computer. Actually, on the contrary, the language is designed to be best understood by the programmer (that's you). A compiler is an application that translates programs from the Java language to a language more suitable for executing on the computer. It takes a text file with the .java extension as input (your program) and produces a file with a .class extension (the computer-language version). You'll need a terminal window for this operation. Find it under Start Button -> Accessories -> Command Prompt. This will bring up a window into which you can type MS-DOS commands. (See sidebar.) Navigate to the directory where you saved HelloWorld.java. Use the command cd to change directories. Ask you instructor for help if you can't find your source file (HelloWorld.java). To compile HelloWorld.java type the text below at the terminal. (We use the % symbol to denote the command prompt, but it may appear different depending on your system.)

Command Prompt Essentials
*You need to change your current directory to the directory where your source file is located. The command to change your current directory is **cd**. The name of the parent directory is ".." To view the files and directories in the current directory, use the **dir** command. To view a list of common commands, type **help** at the command prompt, and then press Enter.*

```
% javac HelloWorld.java
```

If you typed in the program correctly, you should see no error messages. Otherwise, go back and make sure you typed in the program exactly as it appears above.

*Note: If the system does not recognize the **javac** command, you may need to set the path environment variable. One way to do that is:*
set path=C:\Program Files\Java\jdk1.6.0_27\bin
*Depending on the location of the JDK on your system, you may need to set your **path** variable to point to a different location.*

Executing (running) a Java Program Once you compile your program, you can run it. This is the exciting part, where the computer follows your instructions. To run the HelloWorld program, type the following at the terminal:

```
% java HelloWorld
```

If all goes well, you should see the following response

```
Hello, World!
```

☆ When your program runs correctly, get it checked off by your instructor. This will count as your first **Programming Exam**.

Understanding a Java program This simple program has only one executable statement. It's System.out.println("Hello World!"), which sends the string, "Hello, World!" to a method called println, which prints the string to the output. When we begin to write more complicated programs, we will discuss the meaning of public, class, main, String[], args, System.out, and so on.

Second Program

The second programming task for this lab is to do exercise P1.6 from your book. It uses print statements to produce a given output. Probably the easiest way to do this assignment is to **copy** HelloWorld and make the necessary changes. (Don't just change HelloWorld – you need to turn that in as well.) The exercise is reproduced below for those who don't yet have a textbook. Call the class MyName. (**The name is important.**) Be sure to include a Javadoc class comment which includes an appropriate description of the program.

P1.6 Write a program that prints your name in large letters, such as

Logic Check

See the class web site for a separate Logic Check Quiz.

What to hand in

- Get checked off for HelloWorld.java and upload MyName.java to the assignment page.

How the lab will be graded

Grading Rubric	
Description	Points
Logic Check Question	8
MyName	92
Total	100