Homework 0

Elements of Programming

Programming in Java. We break the process of programming in Java into three steps:

1. Create the program by typing it into a text editor and saving it to a file named, say, MyProgram.java.
2. Compile it by typing "javac MyProgram.java" in the terminal window.
3. Execute (or run) it by typing "java MyProgram" in the 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 MyProgram.class); the third actually runs the program.

- Creating a Java program. A program is nothing more than a sequence of characters, like a sentence, a paragraph, or a poem. To create one, we need only define that sequence characters using a text editor in the same way as we do for email.**HelloWorld.java** is an example program. Type these character into your text editor and save it into a file named HelloWorld.java.

```
public class HelloWorld {
   public static void main(String[] args) {
      // Prints "Hello, World" in the terminal window.
      System.out.println("Hello, World");
   }
}
```

- Compiling a Java program. 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). To compile HelloWorld.java type the boldfaced text below at the terminal. (We use the % symbol to denote the command prompt, but it may appear different depending on your system.)

```
% 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.

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

```
% java HelloWorld
```

If all goes well, you should see the following response

> Hello, World

Input and output. Typically, we want to provide input to our programs: data that they can process to produce a result. The simplest way to provide input data is illustrated in **UseArgument.java**. Whenever this program is executed, it reads the command-line argument that you type after the program name and prints it back out to the terminal as part of the message.

```
public class UseArgument {
    public static void main(String[] args) {
        System.out.print("Hi, ");
        System.out.print(args[0]);
        System.out.println(". How are you?");
    }
}
```

Executing it :

```
% javac UseArgument.java
% java UseArgument Alice
Hi, Alice. How are you?
% java UseArgument Bob
Hi, Bob. How are you?
```

Exercises

1. Write a program **TenHelloWorlds.java** that prints "Hello, World" ten times.
2. Modify **UseArgument.java** to make a program **UseThree.java** that takes three names and prints out a proper sentence with the names in the reverse of the order given, so that for example,

> % java UseThree Alice Bob Carol

gives

> Hi Carol, Bob, and Alice.

Built-in Types of Data

Library methods and APIs. Many programming tasks involve using Java library methods in addition to the built-in operators. An application programming interface is a table summarizing the methods in a library.

- Printing strings to the terminal window

`void System.out.print(String s)`	*print s*
`void System.out.println(String s)`	*print s, followed by a newline*
`void System.out.println()`	*print a newline*

- Converting strings to primitive types:

`int`	`Integer.parseInt(String s)`	*convert `s` to an `int` value*
`double`	`Double.parseDouble(String s)`	*convert `s` to a `double` value*
`long`	`Long.parseLong(String s)`	*convert `s` to a `long` value*

- Mathematical functions:

`public class Math`

`double`	`abs(double a)`	*absolute value of `a`*
`double`	`max(double a, double b)`	*maximum of `a` and `b`*
`double`	`min(double a, double b)`	*minimum of `a` and `b`*
`double`	`sin(double theta)`	*sine of `theta`*
`double`	`cos(double theta)`	*cosine of `theta`*
`double`	`tan(double theta)`	*tangent of `theta`*
`double`	`toRadians(double degrees)`	*convert angle from degrees to radians*
`double`	`toDegrees(double radians)`	*convert angle from radians to degrees*
`double`	`exp(double a)`	*exponential* (e^a)
`double`	`log(double a)`	*natural log* ($\log_e a$, *or* $\ln a$)
`double`	`pow(double a, double b)`	*raise `a` to the `b`th power* (a^b)
`long`	`round(double a)`	*round `a` to the nearest integer*
`double`	`random()`	*random number in* [0, 1)
`double`	`sqrt(double a)`	*square root of `a`*
`double`	`E`	*value of e (constant)*
`double`	`PI`	*value of π (constant)*

You can call a method by typing its name followed by arguments, enclosed in parentheses and separated by commas. Here are some examples:

method call	*library*	*return type*	*value*
`Integer.parseInt("123")`	`Integer`	`int`	`123`
`Double.parseDouble("1.5")`	`Double`	`double`	`1.5`
`Math.sqrt(5.0*5.0 - 4.0*4.0)`	`Math`	`double`	`3.0`
`Math.log(Math.E)`	`Math`	`double`	`1.0`
`Math.random()`	`Math`	`double`	*random in* [0, 1)
`Math.round(3.14159)`	`Math`	`long`	`3`
`Math.max(1.0, 9.0)`	`Math`	`double`	`9.0`

Exercises

1. Write a program **Distance.java** that takes two integer command-line arguments x and y and prints the Euclidean distance frome the point(x, y) to the origin (0,0).
2. Write a program **SumOfTwoDice.java** that prints the sum of two random integers between 1 and 6(such as you might get when rolling dice).
3. Write a program **SumOfSines.java** that takes a double command-line argument (in degrees) and prints the value of sin(2t)+sin(3t).