CSC 7200
Week 1 Programming Assignment

Welcome to our first lab. This week, we'll take it somewhat easy. We'll start off with two simple programs: one will be console based and the other will utilize a bit of a Graphical User Interface (GUI) as we get familiar with our development environment. You're free to use any development environment you like and I have posted guides for both the Eclipse and the NetBeans Integrated Development Environments (IDE).

Program 1

For the first program, we'll take a console based approach in which you'll use a Scanner object to get input from the user. You will write a Java application that takes **three integers** as input from the user and displays the following for the three integers:

1. Sum: Integer1 + Integer2 + Integer3
2. Average: (Integer1 + Integer2 + Integer3)/3
3. Product: Integer1 * Integer2 * Integer3
4. Smallest of the three Integers
5. Largest of the three Integers

You will use a Scanner object to get the input (of the three integers) from the user. Please use System.out to output the results to the console. The average should be a **floating point result** and should be displayed with exactly 2 digits after the decimal point.

Program 2

For your second program, you will write a Java application that will test the random number generation capability of Java. Random number generation is explained in the section below. Your application should simulate rolling a pair of dice some number of times. You will use a `JOptionPane` object to ask the user how many times the dice should be rolled. Each die roll can be a value between 1 and 6, so rolling 2 dice will result in values between 2 and 12.

You need to generate a random number between 1 and 6 to simulate rolling each dice. Keep track of the result of each roll using an array that is indexed by the sum of the roll of the two dice. E.g., if you roll a 7, it should be stored in yourArray[7] (there can be variations if you want to account for the 0 and 1 values or take advantage of the zero-indexed nature of arrays in Java).

You will then output the result in a table which will show each value (i.e., the values 2 – 12) and the number of times that value was rolled. The program should have the following structure:

1. The `main()` function should declare the array that keeps track of the count of rolls and also gather input from the user
2. A separate function should be called which takes an integer array parameter as well as the number of dice rolls to execute. This function should roll the dice the required number of times and uses the array provided to count the number of times each value is rolled.
3. The main program should take the count data from the array, format it, and display it as described above.

For the output, you should build a formatted string which should include tabs and new lines to represent your table. `JOptionPanes` by themselves are not capable of properly formatting tabs. However, there is another Java class which can be used with `JOptionPanes` to correctly display a formatted string; this class is the `JTextArea` class and it understands all string formatting instructions. The following code fragment illustrates how to use the `JTextArea` class with a `JOptionPane` to display a formatted string:

```
//import javax.swing.JOptionPane; // Import before class declaration
//import javax.swing.JTextArea;   // Import before class declaration

String output = "This \t is \t a \t formatted \t string. \n";
// Create a new JTextArea object
JTextArea area = new JTextArea( );
// Place a formatted string into the JTextArea
area.setText( output );
// Place the text area into the JOptionPane
JOptionPane.showMessageDialog( null, area, "Title for dialog box",
JOptionPane.INFORMATION_MESSAGE );
```

For each of the two programming problems, create an Eclipse or NetBeans project and develop a Java program to solve the problem. You will also need to take a screenshot of your program's output. The best way to do this is to click on the console window you want to capture and then press the Alt+PrintScreen (on a PC) or Shift+Command+3 (on a Mac) keys at the same time. For each of the two programs, you can then paste your captured screen image, along with a copy of your source code, into a Word document.

Random Numbers in Java

There are two different classes available to generate random numbers in Java. The static method `random()` from the `Math` class can be used to generate a random floating point number from 0 up to, but not including, 1: i.e., as [0,1). This number can be scaled to whatever range of random numbers is desired, as below:

```
(int)(Math.random() * 10);  //results in an integer between 0 and 9
```

The `Random` class from the `java.util` package can also be used to get random numbers. The `nextInt()` method returns an integer from 0 up to but not including the argument value provided to the `nextInt()` method, as shown below:

```
Random rand = new Random( );
int number = rand.nextInt(10); //results in an integer between 0 and 9
```

minimumValue + *differenceBetweenValues* * `rand.nextInt(` *scalingFactor* `)`

The formula shown above is used to generate a range of random numbers starting at some minimum value and generating some specified number of random values. The scalingFactor determines how many different random values will be generated by the call to `nextInt()`, starting with the value 0. The difference between values is usually 1.

Deliverables

- Screenshots of running program (for each of the two programs)
- Source Code for all classes (in each of the two programs)
- Paste Screenshots + Source Code into a single DOC/DOCX file

Lab Grading Criteria

```
Program 1                     60 pts.
Program 2                     60 pts.
Screenshots + Source Code 30 pts.
-------------------------------------
Total   150 pts.

*** Program1:
- Not getting 3 integers from user: -6
- Not using the Scanner object: -6
- Not using System.out: -6
- Not properly formatting average: -6
- No sum, avg, prod, smallest, and largest: 6pts/each

*** Program2:
- Not using JOptionPane: -6
- Not using random number generation correctly: -6
- Not using an array indexed by roll sum: -6
- Not showing output correctly: -6
- Not having the supporting function, as well as the main()
function: -15
```