Assignment 8 - Dice Simulation Class
CSC204 Spring 2017
Due: Wednesday, April 12, 2017

For this assignment you will need to create three Java files that work together. The first, and most involved Java code, is a dice simulation class named DiceSim. That simulation will need instances of a dice object - which you will need to write. Finally, for this assignment you will write a main driving program to create multiple instances that helps a user set up their dice simulation. Details for each Java file are given below.

Setup: Create an Eclipse project named *DiceSim*. You should create the following three Java classes in this project.

Dice.java: You should create your own Dice class. It should contain the following:

- This class should define two integer instance variables: sides and last.
- A default constructor, that is passed zero parameters, that initializes sides to 6 and last to 1.
- A constructor that is passed one integer that initializes sides to the passed in argument, and sets last to 1.
- A public method named roll that is passed nothing, and returns a random integer between 1-sides, inclusively. This method should also set the instance variable last to this generated random number.
- A public method named getLastRoll, which is passed nothing, and returns the value saved in the instance variable last.

DiceSim.java: You should create a DiceSim class that has the following:

- This class should define the following instance variables:
    - an integer named sidesOfDice,
    - an integer named numOfDice,
    - and integer named numOfRolls,
    - an array of integers named countOfRolls,,
    - an array of Dice named theDice.

- This class should include four constructors, each initializing the five instance variables as follows:
    - constructor with no arguments sets:
        - sidesOfDice = 6;
        - numOfDice = 1;
        - numOfRolls = 100;
        - make countOfRolls large enough to hold (sidesOfDice * numOfDice +1) integers, and initialize each to zero.
        - make theDice large enough to hold numOfDice Dice, and fill the array with that many Dice with the correct number of sides.
    - constructor with one argument arg1, sets:
        - sidesOfDice = 6;
        - numOfDice = 1;
        - numOfRolls = arg1;
        - make countOfRolls large enough to hold (sidesOfDice * numOfDice +1) integers, and initialize each to zero.
        - make theDice large enough to hold numOfDice Dice, and fill the array with that many Dice.
    - constructor with two arguments arg1 and agr2, sets:
        - sidesOfDice = 6;
        - numOfDice = arg1;
        - numOfRolls = arg2;
        - make countOfRolls large enough to hold (sidesOfDice * numOfDice +1) integers, and initialize each to zero.
        - make theDice large enough to hold numOfDice Dice, and fill the array with that many Dice.
    - constructor with three arguments arg1, arg2, arg3, sets:
        - sidesOfDice = arg1;
        - numOfDice = arg2;
        - numOfRolls = arg3;
        - make countOfRolls large enough to hold (sidesOfDice * numOfDice +1) integers, and initialize each to zero.
        - make theDice large enough to hold numOfDice Dice, and fill the array with that many Dice.
- This class should have a public void method named runSimulation that is passed nothing and then proceeds to ‘run the simulation.’ That is, this method should simulate rolling numOfDice Dice, numOfRolls times, and counting the values of each set rolled. The sum should then be recorded in the countOfRolls array (i.e. if there were two dice, and one roll gave 3 and 5, the sum would be 8 and you should increment the 8th ‘slot’ of the countOfRolls array).

- This class should have a public void method named displayCount, that is passed nothing. This method should display the results of the simulation. An example of two dice is given below:

```
 2: 6
 3: 10
 4: 14
 5: 17
 6: 20
 7: 24
 8: 19
 9: 18
10: 13
11: 10
12: 4
```

- This class should have a public void method named graphCount, that is passed nothing. This method should display a text bar chart of the results of the simulation scaled to fit on the screen. An example of two dice is given below:

```
 2: ******
 3: **********
 4: ************
 5: ******************
 6: ********************
 7: ************************
 8: ******************
 9: ****************
10: ************
11: **********
12: ****
```

P9DiceSim.java: You are to write a main routine that prompts the user to enter values for this simulation. The prompts and values should include:

- How many sides are on each dice?
- How many dice to you have?
- How many times do you want to roll these dice?

Your main should then create a DiceSim, passing it these three values. You should then go ahead and call your DiceSim's runSimulation, displayCount, and graphCount.

BONUS POINTS: Earn up to 25 bonus points to be added to this assignment. Generate a Java Graphic of the bar graph. Fit it into a JFrame with width 800, and height 600. Have the bars come from the bottom of the graphic, and scale the bars so that the largest bar touches the top of the graphic. This graphic should just pop up after your program prints the text bar graph.

Deliverables: When complete, copy your entire Eclipse project folder named *DiceSim* into your shared Google folder.