CS161: Introduction to Computer Science
Lab Assignment 9

The goal of this week's lab is to give you more practice with arrays and `static` methods. You and your partner will use your detective skills (and your math skills) to investigate the fairness of various die classes.

━━━ Directions ━━━

Download the `lab9` starter code. Inside is the `DieStats` class which you will be implementing for today's lab. The `DieStats` class is responsible for creating and rolling a die. After rolling the die, various statistics should be printed to the screen.

Usually, all of this code would go inside of the `main` method. However, if we did this `main` would become too long. Instead, we will once again move some of this code into other `static` methods and call these methods from `main`.

Inside the `main` method in `DieStats`, you should:

1. Prompt the user to enter the number of sides for the die and the number of times to roll the die. Continue prompting the user until they enter a valid number of sides (i.e. greater than 1) and a valid number of rolls (i.e. more than 0).

2. Once you have this information, create an instance of the `Die` class with the appropriate number of sides. You'll also need to create an array that will hold the number of times each face value is rolled. For now, ignore the other mystery classes.

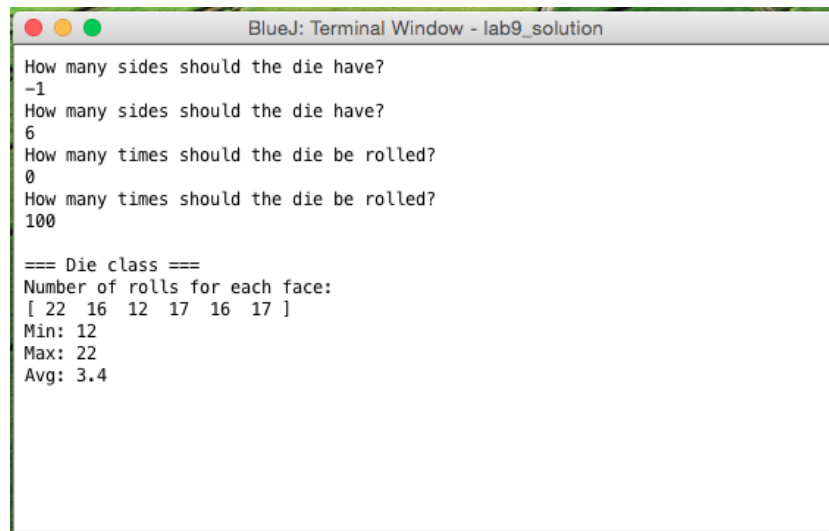3. Roll the die the specified number of times recording the face value each time.

Once you have rolled the die, you should compute various statistics about the rolls. Each of these statistics corresponds to a `static` method that you need to implement. Once you have implemented the method, you can call it from `main`.

1. The `printArray` method should print the elements of the array to the screen.

2. The `min` and `max` methods should find and return the minimum (or maximum) number of rolls in the array.

3. Finally, the `averageValue` method should return the average face value. You can use the following equation to compute the average face value:

$$\text{avg} = \sum_{i=1}^{S} i \cdot \text{array}[i]$$

where S is the number of sides of the die and `array[i]` is the number of times the face value i was rolled. Computing the average face value doesn't have much meaning in the real world but it may prove helpful later on.

Here's a sample run of the program.

Mystery Dice

Now that you have `DieStats` working, it's time to use your code to investigate the various "mystery" die classes. You'll notice that the starter code has 3 additional die classes named `MysteryDie1`, `MysteryDie2`, and `MysteryDie3`. BlueJ displays the message "(no source)" for each of these classes. This is because you don't have access to the source code. You only have access to the the compiled Java bytecode for each class.

Even without the source code, you can still create instances of each class. Inside `DieStats`, repeat the same process that you did for the `Die` class for each of the mystery die classes. The next page shows a sample run of my program.

Use the output of your program to answer the following questions:

1. Are any of the mystery die *weighted*? How do you know?

2. How many rolls of the die must you do before you're absolutely certain that a die is weighted or not weighted? Is 100 rolls enough? 1000? 10,000?

Call me or the lab assistant over to discuss your findings.

Finally, since we're discussing "creative gaming strategies", let's perform one last dubious operation on our array of counts. There is still one unimplemented method in the `DieStats` class called `swapMinToEnd`. This method finds the minimum value in the array and swaps it with the value in the last spot of the array. In other words, it "modifies" the array so that a face value of 6 always seems to have the fewest number of rolls.

Implement this `static` method in the `DieStats` class and then run your code to make sure it works correctly.

```
●  ●  ●              BlueJ: Terminal Window - lab9

How many sides should the die have?
6
How many times should the die be rolled?
100

=== Die class ===
Number of rolls for each face:
[ 15   18   18   17   22   10 ]
Total rolls: 100
Min number of rolls: 10
Max number of rolls: 22
Avg face value: 3.43

=== MysteryDie1 ===
Number of rolls for each face:
[ 15   17   15   16   9   28 ]
Total rolls: 100
Min number of rolls: 9
Max number of rolls: 28
Avg face value: 3.71

=== MysteryDie2 ===
Number of rolls for each face:
[ 20   10   14   14   20   22 ]
Total rolls: 100
Min number of rolls: 10
Max number of rolls: 22
Avg face value: 3.7

=== MysteryDie3 ===
Number of rolls for each face:
[ 17   19   19   13   15   17 ]
Total rolls: 100
Min number of rolls: 13
Max number of rolls: 19
Avg face value: 3.41
```

Submitting Your Lab

Rename your folder with both of your names and then upload it to Moodle.