

CS161: Introduction to Computer Science

Lab Assignment 9

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

Die Experiments

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, you should organize your code into private, `static` methods to keep the `main()` method short and to cut down on any repeated code.

Overall, 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.

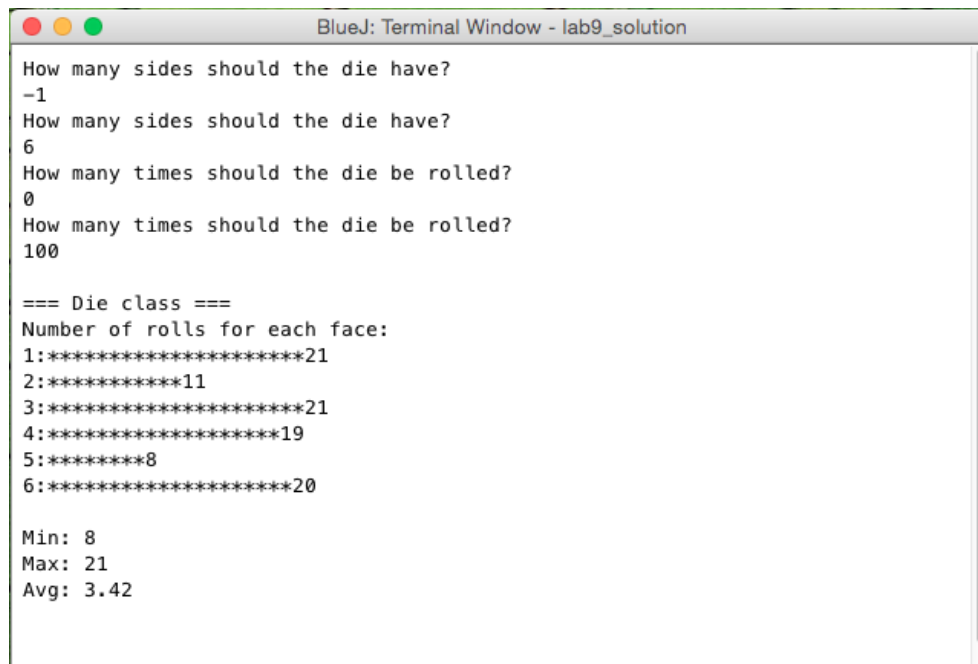
Once you have rolled the die, you should compute the following statistics about the rolls.

1. A (sideways) histogram of the number of times each face value was rolled (**see example below**)
2. The minimum and maximum number of rolls in the array.
3. Finally, compute the average face value. You can use the following equation to compute the average face value:

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

where S is the number of sides of the die, `array[i]` is the number of times the face value i was rolled, and `numRolls` is the total number of times the die 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 my program.



```
BlueJ: Terminal Window - lab9_solution
How many sides should the die have?
-1
How many sides should the die have?
6
How many times should the die be rolled?
0
How many times should the die be rolled?
100

=== Die class ===
Number of rolls for each face:
1:*****21
2:*****11
3:*****21
4:*****19
5:*****8
6:*****20

Min: 8
Max: 21
Avg: 3.42
```

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. Make sure that you use private methods to cut down on any repeated code.

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.

Extensions

When the number of rolls is large (e.g. 10,000), the histogram is not very informative. Each row of asterisks is so long, we have to scroll just to find the end. How can you fix this problem? How can you modify the histogram so it's easy to understand even when the number of rolls is large?

Submitting Your Lab

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