Homework 03 - Coin Flips

How many coins would you need to flip in order to have a certain number of heads and a certain number of tails? Your objective is to create a simulation that answers this question. Your program must continue running until the desired number of heads and the desired number of tails are reached. Assume that for any particular coin flip, the chances of getting a head and the chances of getting a tail are both 50 percent.

The program will take in from the user the number of sequnces to generate. Then it will prompt the user to enter the number of heads and the number of tails that the user wishes. Note, this value will be shared by all of the sequences in the current simulation. Once all of the sequences have been generated (and printed with `H` representing a head and `T` representing a tail), the program should print out both the average and the maximum number of coin flips necessary to achieve the problem's constraint.

After that, the user should be prompted whether they want to run another simulation and will continue in the same fashion until the user decides to quit. To run again, the user will input `y`. To stop, the user will input `n`. However, the user might not enter `y` or `n` — in that case, the program should continue prompting the user whether they want to run another simulation or not.

A sample run might look like the following:

```
Ready to run a coin flip simulation. Enter the number of sequences: 4
How many heads should each sequence have? 2
How many tails should each sequence have? 3
Simulating Sequences
1 - HHTHTT
2 - THHTT
3 - HTTHT
4 - TTHTTH
The average number of flips was 5.5 and maximum was 6

Would you like to run another simulation? (y/n): maybe
Would you like to run another simulation? (y/n): y

Ready to run a coin flip simulation. Enter the number of sequences: 3
How many heads should each sequence have? 1
How many tails should each sequence have? 1
Simulating Sequences
1 - HT
2 - TTH
3 - HHT
The average number of flips was 2.67 and maximum was 3

Would you like to run another simulation? (y/n): n
```

Notes and Hints

Round the average number of flips to the nearest hundredth.

You can assume that the user will only enter an integer for the number of sequences, number of heads, and the number of tails but an invalid int can be entered like -1. Your program should handle situations where the user inputs an invalid int by going to the continue question, ie. `Would you like to run another simulation? (y/n):` and if the user chooses `y`, beginning another simulation. An example is below.

```
Ready to run a coin flip simulation. Enter the number of sequences: 3
How many heads should each sequence have? -4
Would you like to run another simulation? (y/n): y
```

```
Ready to run a coin flip simulation. Enter the number of sequences: 3
How many heads should each sequence have? 1
How many tails should each sequence have? 1
Simulating Sequences
1 - HT
2 - TH
3 - HT
The average number of flips was 2 and maximum was 2
Would you like to run another simulation? (y/n): n
```

Food for Thought

A particularly interesting case is the simple one of getting one head and one tail. On average, how many coin flips do you think that will take? Try running simulations with more and more sequences of coin flips. How does the average change as there are more sequences? Is it converging to a value? How does the maximum change as well?

We will not require you to turn in your observations. However, this highlights a pretty neat use of your `CoinFlip.java` file, so give it a shot!

Rubric

- [100] `CoinFlip.java`
  - [34] Input/Output Format
    - [10] Prompts for 3 numbers in the following order: number of sequences, number of heads, number of tails
    - [10] Outputs the correct number of HT sequence Strings
    - [10] Correctly presents the option to run program again (as many times as the user wants)
    - [4] Jumps to the `Continue? (y/n)` prompt if any negative number is input
  - [46] Functionality of simulation sequences Strings
    - [30] Valid String format
      - [15] Has the minimum amount of heads and tails requested (output isn't too short)
      - [15] Stops after has the requested number of heads and tails (output isn't too long)
    - [16] Strings match the random chances as described
      - [8] Strings can randomly change in length and composition from sequence to sequence
      - [8] There is an equal chance (50/50) of a H or a T appearing as the next character in a sequence string
  - [20] Average and Maximum Correct and appear in the output after the last sequence string
    - [10] Correct Average value (should be a floating-point value)
    - [10] Correct Maximum Value (should be a whole number value)

Allowed Imports

To prevent trivialization of the assignment, you may only import `java.util.Scanner` and `java.util.Random`.

Checkstyle

You must run checkstyle on your submission. The checkstyle cap for this assignment is **10** points. If you don't have checkstyle yet, download it from Canvas -> Files/Resources. Place it in the same folder as the files you want checkstyled. Run checkstyle on your code like so:

```
$ java -jar checkstyle-8.28.jar yourFileName.java
Starting audit...
Audit done.
```

The message above means there were no Checkstyle errors. If you had any errors, they would show up above this message, and the number at the end would be the points we would take off (limited by the checkstyle cap mentioned above). The Java source files we provide contain no Checkstyle errors. In future homeworks we will be increasing this cap, so get into the habit of fixing these style errors early!

Feature Restrictions

There are a few features and methods in Java that overly simplify the concepts we are trying to teach or breaks our auto grader. For that reason, do not use any of the following in your final submission: var (the reserved keyword) System.exit

Collaboration

Collaboration Statement

To ensure that you acknowledge collaboration and give credit where credit is due, **we require that you place a collaboration statement as a comment at the top of at least one java file that you submit**. That collaboration statement should say either:

I worked on the homework assignment alone, using only course materials.

or

In order to help learn course concepts, I worked on the homework with [give the names of the people you worked with], discussed homework topics and issues with [provide names of people], and/or consulted related material that can be found at [cite any other materials not provided as course materials for CS 1331 that assisted your learning].

Recall that comments are special lines in Java that begin with `//`.

Turn-In Procedure

Submission

To submit, upload the files listed below to the corresponding assignment on Gradescope:

- `CoinFlip.java`

Make sure you see the message stating "HW03 submitted successfully". From this point, Gradescope will run a basic autograder on your submission as discussed in the next section.

You can submit as many times as you want before the deadline, so feel free to resubmit as you make substantial progress on the homework. We will only grade your last submission: be sure to **submit every file each time you resubmit**.

Gradescope Autograder

For each submission, you will be able to see the results of a few basic test cases on your code. Each test typically corresponds to a rubric item, and the score returned represents the performance of your code on

those rubric items only. If you fail a test, you can look at the output to determine what went wrong and resubmit once you have fixed the issue.

The Gradescope tests serve two main purposes:

1. Prevent upload mistakes (e.g. forgetting checkstyle, non-compiling code)
2. Provide basic formatting and usage validation

In other words, the test cases on Gradescope are by no means comprehensive. Be sure to thoroughly test your code by considering edge cases and writing your own test files. You also should avoid using Gradescope to compile, run, or checkstyle your code; you can do that locally on your machine.

Other portions of your assignment can also be graded by a TA once the submission deadline has passed, so the output on Gradescope may not necessarily reflect your grade for the assignment.

Important Notes (Don't Skip)

- Non-compiling files will receive a 0 for all associated rubric items
- Do not submit `.class` files.
- Test your code in addition to the basic checks on Gradescope
- Submit every file each time you resubmit
- Read the "Allowed Imports" and "Restricted Features" to avoid losing points
- Check on Piazza for all official clarifications
