COMS W1004

Fall 2020

Programming Project 3 (70 points)

Due October 26, 11:59PM

Consider the following game between two players: Both players simultaneously declare "one" or "two". Player 1 wins if the sum of the two declared numbers is odd and Player 2 wins if the sum is even. In either case the loser is obliged to pay the winner (in tokens) the sum of the two declared numbers. So Player 1 may have to pay 2 or 4 tokens but may win 3 tokens.

**Part 1:** Write a computer program in Java that allows a user to play this game against a computer. The computer's strategy will be as follows. A computer player will have a threshold variable, t. The computer will generate a random number between 0 and 1.

will declare "two" if the random number is less than t the computer will declare "one". I have included templates for a Game class, a computer Playerclass and a test class, OddEven on Codio.

**Part 2:** Modify the Game class so that it also allows two computer players to play a game against each other. Do this by overloading the constructor so that when a game is instantiated one may specify whether or not it is interactive or simulated. Since both players are computers in a simulated game each computer player will be a different object with its own threshold (instance) variables t and and its own score (tokens won or lost so far in a session).

Write another test class called Simulation that allows you to run some simulations (play many games of computer versus computer) using various combinations of the threshold variable t for each player. A simulated game need not print or return anything but you should add some functionality to the Game class so that you can access the current amount of tokens either player has won or lost so far. Check to see how much each player loses or wins for each combination of thresholds after many games. Is it better to be the odd player? The even player? Does it matter? Better here means that if enough games are played there is a strategy that one player can use that will guarantee positive average outcome regardless of the other player's strategy. We call it a fair game if there is no such strategy for either player. By using the computer vs. computer option in your program set up some extended sessions of computer vs. computer to test different combinations of player 1's t and player 2's t (*Hint: use a nested for loop structure to vary each player's threshold*). Determine if either player has an advantage and if so which player it is and determine a threshold value t\* that demonstrates the advantage. I have included a sample test class for this part called SimTest. You should not alter this class and your code must work with it. We will test your code using something similar to this.

What to hand in:

For this assignment you should submit the following classes: Simulation, Player, and Game. No need to submit OddEven or SimTest. In addition to these source files include a text file named readMe.txt with an explanation of what you did for each part. That is, write in plain English, instructions for using your software, explanations for how and why you chose to design your code the way you did, and most importantly ***how you used your simulations to determine whether or not the game was fair and the outcomes of your experiments***. The readMe.txt file is also an opportunity for you to get partial credit when certain requirements of the assignment are not met. Finally, you should also cite the names of students you worked with and any external sources you used in your readMe. Put the readMe.txt file in the same directory as your source and use the usual submission instructions for submitting the assignment.

Submitting your assignment

As with Programming 2 we will be using Gradescope to submit your work. In the navigation bar for Courseworks you will find a link to the course Gradescope page. Put your .java files and your readMe.txt file into a folder with the name [your UNI here]\_project3 and compress this folder after which you should have a file that looks like ac1076\_project3.zip except with your UNI instead of mine. Submit this file under

A word about Grading:

Part 1 and Part 2 will be graded using the following guidelines:

* 30% if it compiles
* 30% if it runs properly (expected output for given input, etc.)
* 20% for style (formatting of code, variable names, comments, etc. Use the style guide posted on Courseworks!)
* 20% for design (efficiency, class organization, method length, etc...)

Please make sure your program at least compiles before you submit! There will be no partial credit for a program that "almost" compiles. ``