CS 1063 Lab 6: Simulating Playoffs

The `Playoffs` Class

Objectives

- Call methods with parameters and return values.
- Use the Scanner and Random classes.
- Use while statements

Hand-in Requirements

All projects and laboratories will be submitted electronically through Blackboard. Zip up your entire lab directory to submit as the source. (Right click on the lab folder and follow **Send To > Compressed (zipped) Folder**.) The lab folder should include the following:

- `Playoffs.java`
- `PlayoffsOutput.txt`

Task

Simulate a best-of-seven games playoff using information supplied by the user. Print information about the result of each simulated playoff and about the result of multiple simulations. Ensure that the user enters a valid value for the simulation.

Similar to previous programs, your program should start by printing

```
Lab 6 written by YOURNAME
```

Details

Suppose two teams are playing each other in a best-of-seven playoff (the two teams play each other until one team wins four games). Suppose one team is more likely to win each game (say 51%). What are the chances that the better team will win the playoff?

We can solve this problem mathematically (the answer is about 52.2%), but simulations are often needed for more complicated problems involving uncertainty. Multiple simulations can give us an idea of what the chances are. For example, if we simulate a playoff 2500 times, our results will probably be within 1% of the correct answer (oversimplifying statistics here).

How can you simulate a 51% chance? The standard technique is to generate *pseudorandom* numbers, a sequence of numbers that appears to be random, but is actually generated by a program. The Random class provides pseudorandom numbers in Java. Here is an example.

```
// you need import java.util.*; at the top of your program
Random rand = new Random( );
int num1 = rand.nextInt(100);
if (num1 < 51) {
    System.out.println("Team 1 won the first game");
} else {
    System.out.println("Team 2 won the first game");
}
int num2 = rand.nextInt(100);
if (num2 < 51) {
    System.out.println("Team 1 won the second game");
} else {
    System.out.println("Team 2 won the second game");
}
```

Here, the `nextInt` method of the Random class is used to generate a random `int` between 0 and 99. Change the 100 to a different value if you want a different range. Each integer between 0 and 99 is equally likely. 51 of these integers are less than 51 (from 0 to 50), so testing if the integer is less than 51 means that 51 integers make the test true, and that 49 integers make the test false, a 51% chance of being true! Change the 51 to a different value if you want a different percentage.

We will need to be able to simulate a single game. If we can simulate a single game, we can simulate multiple games until one team wins four games. If we can simulate a playoff, we can simulate any number of playoffs. The user will be asked for the chance that Team 1 will win a given game.

Methods

Write one method to play a single game. This method should have two parameters: the percent chance that Team 1 will win the game, and a Random object. This method should only generate one random number. This method should return one value if Team 1 wins and a different value if Team 1 loses. In your final submission, this method should not print anything (comment out any print statements).

Write another method to simulate a playoff, that is, to play games until one team wins four games. This method **must** use a while loop for flow of control. This method should have two parameters: the percent chance that Team 1 will win the game, and a Random object. This method should return different values depending on which team wins. In your final submission, this method should not print anything (comment out any print statements).

Write a third method to keep doing playoffs until one team has won 10 more playoffs than the other team. This method **must** use a while loop for flow of control. This method should have two parameters: the percent chance that Team 1 will win the game, and a Random object. This method should print a 1 every time team 1 wins, and print a 2 every time team 2 wins; this should all be on one line. This

method should also print the final results.

The main method should ask the user for the percent chance that team 1 will a game. It should ensure that the user enters an integer between 0 and 100. This method **must** use a while loop to ensure that a valid value has been entered. After obtaining a valid value, the main method should construct a Random object. Finally, the main method should pass the two values (the percent chance and the Random object) to the third method.

Optional

There is no extra credit for doing any of the following. It is for those students who have nothing better to do. Or it is for those students who do have something better to do, but would rather play around with their programs.

1. Change the method that simulates multiple playoffs so that it prints how many playoffs ended up four games to none, four games to one, four games to two, and so on. One way to do this is to encode both the winner and the number of games in the return value of the single playoff method. For example, a positive value could be used to indicate that Team 1 won, and a negative value to indicate that Team 2 won. The absolute value could indicate the number of games in the playoff. For example, a -6 would indicate that Team 2 won in 6 games, that is, Team 2 won the playoff four games to two.

   Actually, this sort of encoding is not a good idea because it makes the program more complex to understand. We really should have a way to return two separate values, but we haven't covered arrays yet and programming your own objects is the next programming class. Despite the drawbacks, programmers often like to be overly clever anyways, so it's good to see at least one example.

2. Prevent the program from crashing if the user doesn't enter a number. If program tries to execute the `nextInt` of the Scanner class, but the user has entered a word instead of an integer, the program will crash. We can test whether the next token is a `int` by using the `hasNextInt` method.

   This part of the program should use the `hasNext` method to determine if the user has entered a token, the `hasNextInt` method to determine if that token can be read as an `int`, the `nextInt` method to read the `int`, and finally, the `next` method to read any token that is not an `int`. You will need to use the `next` method so that you get the bad token out of the way.

3. Allow the user to enter a real value, for example, 51.5 to indicate 51.5%. In this case, the random number generation must be modified. The `nextDouble` method of the Random class returns a `double` uniformly distributed between 0 and 1 (specifically, greater or equal to 0 and less than 1). Multiplying this number by 100 will give us a number between 0 and 100. Testing whether this number is less than 51.5 will correspond to a 51.5% chance. If you do this, be sure that your parameters are `double`s instead of `int`s.

4. Allow the user to repeat simulations or not. The program as specified allows the user to do one set of multiple playoffs. Maybe the user would like to try it again with the same percent chance or a different one. Of course, you should allow the user to quit if desired.

Rubric

Your program should compile without any errors. A program with more than one or two compile errors will likely get a zero for the whole assignment.

The following criteria will also be used to determine the grade for this assignment:

- [2 points] If your submission includes the following:
  - The main method prints "Lab 6 written by [...]".
  - Your submission was a Zip file named `lab6.zip` containing a folder named `lab6`, which contains the other files.
  - If your Java program was in a file named `Playoffs.java`.
  - If the output of your Java program was in a file named `PlayoffsOutput.java`.
  - If your program contains a comment that describes what the program does and contains a comment describing each method.
  - If your program is indented properly.
- [3 Points] For a correct implementation of the method that simulates one game. See the methods section above.
- [5 points] For a correct implementation of the method that simulates one playoff. See the methods section above.
- [5 Points] For a correct implementation of the method that simulates many playoffs. See the methods section above.
- [5 Points] For a correct implementation of the main method. See the methods section above.

Revision Date Mon Oct 31 2011 10:20:55 GMT-0500 (Central Daylight Time)