

**Csci 1301, Project 1**  
**2008.06.30**  
**PlayoffSeries.java**

The second project is a fairly complicated program. Everything we've studied so far will make an appearance: conditions, loops, even random numbers. You're going to be investigating how to simulate a playoff series repeatedly in order to approximate the probability of the various outcomes. Now, we could devise formulas for calculating what you are going to approximate. But, once written, your code will allow you to quickly and easily modify the problem's parameters to get new results. Instead of repeating a laborious calculation, you will change one or two values and rerun the program.

**Matt and Tim are friends and athletic rivals. They compete at everything and, while Tim occasionally pulls out a victory, it's obvious that Matt is the better athlete. Recently Tim challenged Matt to a series of croquet matches. The first person to win at least four matches, lead by two matches, and win the last two matches in a row will be declared the winner of the series. Matt's superior reflexes and strategic prowess mean that, on average, he will win two out of every three matches played.**

**Meanwhile, their friend Betty is eager to place a wager on the outcome. But first she wants to estimate the odds of each player winning. You are going to complete a program that will estimate these odds for her. In order to do this you will simulate a large number of series.**

A playoff series increases the probability that the better player will win relative to a single championship match. The requirement that the winner also lead by two matches further benefits the better player. And requiring the series to end in a two game winning streak for the leader adds yet another benefit to the better player. So we know that Matt will be favored to win. Betty needs to know just how likely that outcome is. I've provided the skeleton of a main method that will perform the simulations in a file called PlayoffSeries.java. I've included a very basic structure:

- (1) an outer for loop to handle a (potentially) large number of playoff series
- (2) an inner while loop to handle each series until either Matt or Tim wins
- (3) code to perform a single match and update any variables necessary
- (4) following the for loop there will be an output statement showing how many series each player won

I have used block comments- `/* ... */` -to indicate where code needs to be added. You don't need to add code elsewhere, but if you want to, be my guest. Just be careful not to do anything too complicated unless you can make it work; the solution I've suggested is a pretty simple one. All of the other comments are the actual comments from my complete version of the code. If I've done a good job, they should help you, even though the original code is gone. Feel free to add/remove/change comments.

## Csci 1301, Project 1

2008.06.30

### PlayoffSeries.java

Once you have the program working, answer the following questions in comments at the beginning of your code. I recommend running 1,000,000 simulated series. A large number of simulations will make our approximation more accurate. Give at least four digits (for example 12.34% or 0.1234 instead of 12% or 0.12). Feel free to give more.

1. Roughly what percentage chance does Matt have to win a single series as described above?
2. Now we'll adjust the rules so a player still needs to be the first to win four matches, but now only needs to have a one match edge and win the last match (one in a row, if you prefer that terminology). (That is, you can now win 4-3.) How likely is Matt to win a series under these rules?
3. Sticking with this new set of rules, let's say Matt is suffering from a serious wrist injury that affects his skill with a mallet. Now Matt only has a 51% chance of winning each match. How likely is Matt to win a series now?
4. Alright, let's change it up one more time. Tim and Injured-Matt, who has a 51% chance to win each match, are going to compete in a truly epic playoff series. The winner is the first to win at least 21 matches, have a four-match lead over their rival, and win the last three games in a row. How likely is Matt to win this series?

#### Hints and Advice

1. Plan ahead. Don't just dive right in. Which variables and constants will you need? How should they be used?
2. When you are first writing and debugging your code, try using a smaller number of trials. This will make your testing process go much faster. Once you are confident things are working, run it for 1,000,000 simulated playoff series. This should give you a pretty accurate approximation.
3. Remember that well-written code is its own documentation. Use descriptive variable names and extra parentheses where appropriate.
4. Questions 2, 3, and 4 can be answered by making VERY minor changes to the code. Don't try to reinvent the wheel.

**Csci 1301, Project 1**  
**2008.06.30**  
**PlayoffSeries.java**

**Project Grading**

As usual, 80% will come from having correct code and correct answers to the questions. 20% will come from having code that is of good style.

Correctness:

- 2 points for answering the four questions correctly (0.5 points each).
- 2 points for having your program output the correct result for two test cases. (I will go in and modify your code. I will only change the constant values at the beginning of main! Your code needs to work when only those changes are made.)
- 1 point for a correct `for` loop and correctly adjusting values within it.
- 1 point for a correct `while` loop and correctly adjusting values within it.
- 1 point for a correct `if-else` conditional to determine the winner of a match (inside the `while` loop) and correctly adjusting values within it.
- 0.5 points for a correct `if-else` conditional to determine the winner of a series (inside the `for` loop) and correctly adjusting the values within it.
- 0.5 points for a correct output statement at the end of the program.

Style:

- 0.5 points for avoiding hard-coded constants. This means declaring a named constant instead of just throwing numbers like 0.51 or 21 into your conditions or loops.
- 0.5 points for following the tabbing conventions within loops or conditionals.
- 0.5 points for having well-named variables and constants and following the capitalization conventions.
- 0.5 points for using the `Random` class instead of `Math.random()`. The `Random` class is preferable, because it allows the programmer to set a random seed if they like. Though we will not be explicitly setting a seed, doing so allows the exact sequence of random numbers generated to be replicated. This makes the results reproducible- an important consideration when writing code to be used in research.

Other requirements:

**If your code does not compile, you will get a 0.** Make absolutely certain your code compiles before you submit it via email.

**If your code does not have a header, you will be penalized 20%.** You must include a comment header on every project. The header must include your name and a one to two paragraph description of how your program works. The statement of academic honesty must also be a part of the header.