CS 151: Object-Oriented Design

Fall Semester 2013

Department of Computer Science
San Jose State University
Prof. Ron Mak

Assignment #4

Assigned: Thursday, October 3
Due: Friday, October 25 at 11:59 pm
Team assignment, 100 points max

Smart computer throw calculator

In this assignment, you will add a "smart algorithm" to your Rock-Paper-Scissors game application that you wrote for Assignment #3.

When a human player plays the game, the human's choices are not truly random. (Assume the human player isn't using dice to determine choices.) If during a match the computer records sequences of the last N throw choices ending with the human's choice, it can detect patterns and thus attempt to predict what the human will choose next. For N = 3, an example record of choice sequences is the following (in no particular order; the human's choices are underlined):

SRP, PRS, RPR, PSR, SRP, SPS, PSS

Suppose the last pair of throws is SR. The computer can predict that the human most likely will next throw Paper, since SRP appears twice in this small sample. Therefore, the computer's throw calculator should choose Scissors to beat the human's predicted throw of Paper.

The longer the matches are, the more throw choice sequences the computer has recorded during the match, and the harder it will be for the human player to win the match. This is a very rudimentary example of **machine learning**, a branch of the computer science discipline **artificial intelligence**.

For this assignment, you can initialize a fresh record of choice sequences each time you restart your application. In later assignments, you can store the record in a disk file so that the application's "experience" grows over time with more matches.

Implement the smart computer throw calculator

As in Assignment #3, when you run your application, number of throws per match should be a parameter on the command line. Add another command-line parameter that indicates whether the computer's choices should be calculated using the random algorithm from the previous assignment or the smart algorithm for this assignment.

Before printing what the computer chose to throw, your program should print what the computer predicted that the human will throw.

Use good object-oriented techniques to design how your application records throw choice sequences of length N = 3 and how your throw calculator looks up sequences to predict the human's next choice. Make your design flexible enough to handle larger (odd) values of N.

JUnit tests

Include three JUnit test classes. Each test class should contain at least two test cases.

What to turn in

Each team should create a zip file containing all the Java source files of its application. Zip the **src** and **test** subdirectories that NetBeans creates when you create a new project. Do not include any .test or .jar files.

Email the zip file as an attachment to ron.mak@sjsu.edu. CC all your team members. Do not email any executable files because some mailers will reject the entire message. In the subject line, please include **CS 151 Assignment #4 team name**.

Your email message should include instructions on how to run your application, such as what the second command-line parameter should be.

NOTE: If your mailer rejects the zip file as an attachment, try renaming the file so that the suffix is something other than **zip**, such as **zzz**.

This is a team assignment. Each member of the team will receive the same score.