CS 1632 – FINAL DELIVERABLE

BUILD APPLICATION USING TDD

MARCH MADNESS PREDICTOR

ALEC FOX

TuTh 4pm

<https://github.com/adf37/CS1632_Projects/tree/master/Deliverable6>

# Intro:

I choose to make a new application from scratch using test driven development, or TDD. With the craze, and personal passion, for the March Madness tournament that pits the best 64 teams against each other in a competition for the college basketball championship. Many fans and casual observers make brackets to try and best predict the outcomes of all the games. This got me to thinking why not make a program that tries to simulate and create statistic based system for predicting the winners?

This program uses the TDD style of development. This is a red, green, refractor loop of first making a test that initially fails, writing enough code to make the test pass, and then refractoring the code for more functionality or clean up code sections. I wrote simple assertion tests that would use doubles and stubbing to test the different methods and classes. Testing that setter methods for each Team class was checked along with other needed necessary first pieces of the program such as reading in the csv file containing the data, initializing the different regions, and so on. After the initial set up was tested and written there was a bottom up strategy used. First, head to head matchups needed to work as expected so entire regions could be simulated. Once each region could be simulated the final four teams could be determined resulting in a champion. TDD allows the building of functional code written in a way so it will pass necessary tests and ignore superfluous code and functionality that most likely won’t be needed.

# Problems faced:

The initial and biggest problem I faced when writing tests for this program was handling user input within the tests. The way the program works is that the user is responsible for putting an argument on the command line of one of the following: tournament, region, or game/matchup. This input would call the initialize method which in turn would run the subsequent methods needed to complete the user’s request. However, when using the command region or game/matchup there is additional user input that is needed to complete the program’s simulations. In the case of a matchup, the user needs to input two valid teams that are in the tournament to be able to simulate a matchup between the two. To do this I had to learn how to send in this input in the automated tests. Creating a byte array input stream and setting the system to take in this input stream for the scanner requests allowed the JUnit tests to run successfully.

Problems going forward with the program would be that this model of analysis uses too few variables and assumes that each team plays around their expected level of play during the season. However, most teams don’t always play as well as they did on average during the season and suffer letdowns are play far better than expected. The model doesn’t account for the possibility of upsets in general based on the current data. To make the model better, more variables can be used that analyze a variety of more different advanced metrics that are available.

# Quality Assessment:

The project under test is the March Madness Predictor. The program runs a simulation of the entire bracket of the 2016 March Madness Tournament. There are not any failed tests or problem areas in the code. The quality of the program is moderate with respect to the actual results of this year’s tournament. Teams that the program predicted to go far in the tournament did not in most cases and many of the upsets of the first two rounds were not picked correctly. Therefore, the program does not hold an edge over the casual fan making their best educated on how the matchups will play out.

The program will need to be modified with the data being used expanded. More advanced metric variables and proper weights being applied should give a more legitimate result to each individual matchup. Currently only five total metric variables are being used in the simulations to predict each game. A proper assessment of a program that would give the user an edge in prediction would be to generate data from a given time period, the last decade for example, and run simulations for each year comparing the actual results with those predicted. A program that generates results that are in the 75th percentile or higher is the goal of the program. At this point the program could be launched and distributed on the open market. For this reason, at this time, I am recommending that the software not be released.

# Screenshots:

