HW4: Simulating the NCAA Men's and Women's Basketball Tournament

In this assignment, you will use the Elo rating system to generate predictions for the NCAA men's basketball tournament and the NCAA women's basketball tournament. If every game were 50/50, the odds of a perfect NCAA bracket would be 1 / (2^67) (counting the 4 play-in games) in the men's, and 1 / (2^63) in the women's, which are very small numbers [citation needed]. In reality, some outcomes are more likely than others, so the odds of a perfect bracket are likely closer to only 1 in a few billion. Thus, rather than simulate using just a 50/50 coin flip model, we will attempt to add some context using Elo ratings.

In this assignment, you will predict the NCAA Tournaments using Design Patterns, specifically the Singleton, Factory, Observer, and Strategy Design Patterns. Additionally, you will use the 3-Tier Architectural Pattern covered in class.

Elo rating system

Elo is used to predict how good a participant is at some event. It was used first, and developed for Chess, wherein a player with an Elo of 1500 would be expected to beat any player with an Elo of 1499 or less more often than half the time. The bigger the difference, the more likely the win. So if two NCAA teams are the same rating, the game is 50/50. If two teams are seperated by 100 points, it's 64/36 (the favored team wins 64% of the time). If two teams are seperated by 800 points, it's 99/1, meaning the favored team would win 99% of the time.

Example, in the first round of the 2016 West Virginia University plays Bucknell. West Virginia has an Elo of 1966, whereas Bucknell has an Elo of 1679. This means we would expect West Virginia to win roughly 84% of the time (for the formula used, see this web page). **Note that we are not simulating the score, only the Win/Loss outcome.**

The advantage of Elo is that it is simply to use. The disadvantage is that it requires a large number of games to converge on accurate numbers. Further, because most play in college basketball is intra-conference play, it can lead to difficulty in comparing teams in different conferences.

Functional Specification

Please make sure your main method is in the “edu.upenn.cis350.hwk4.Main” class.

Resource Files

I'm giving you the following resource files on this assignment:

eloMen.csv and eloWomen.csv - (Note that currently these link to the 2017 NCAA Tournaments' data files. I will update these files with the 2018 NCAA Tournaments after selection Sunday on March 11.) These files contain the seed, the region and the starting Elo ratings of each of the 68 teams in the Men's NCAA Tournament Field and 64 teams in the Women's Tournament Field, respectively.

eloMen.json and eloWomen.json - **These files will not be given to you until 1 week before the due date.** Why? Because, to test your design choices, I want you to be able to add json functionality very easily into your existing program. This ties in to the design patterns portion of the assignment.

gamesMen.csv and gamesWomen.csv - (Note that currently these link to the 2017 NCAA Tournaments' data files. I will update these files with the 2018 NCAA Tournaments after selection Sunday on March 11.) The files list of games to be played. In this sheet, each game is numbered, and each row states which game the winner goes to (this matches the NCAA Tournament bracket). In each row, you have a game number, Team 1 and Team 2, and where the winner goes. The exception is the championship game, which has no game for the winner to go to. So the "winner goes to" column is negative 1. **You can assume you only need to worry about the csv file here.**

Command Line Arguments

Your program will be run as follows:

```
YourProgramName elofile gamesfile logfile
```

elofile is your elo rankings (you are welcome to change these and play around with them. I personally made West Virginia 9001 so I can live in the delusion that they might actually win).

gamesfile are the games in the NCAA Tournament. You do not need to check this file to assume it's correct. This assignment isn't about error checking. Just assume it's well formed.

logfile is, you guessed it, a log file that writes all user inputs. Additionally, every time a game is simulated, the results should be written to the file. There's no specific format required, but "Team A beat Team B" will do.

User interface

Your system will implement a command line user interface with the following options:

1) Simulate a single hypothetical game
2) Simulate the entire tournament
3) Change a single team's elo rating
4) Exit

Option 1) Simulate a single hypothetical game

In this option, you will prompt the user to enter the name of two teams. When you prompt the user for a team, assume the user has to spell it exactly like it is in the resource csv/json file including white space, but ignoring case. If a team is misspelled or doesn't exist, just prompt the user again until they enter a correct team name.

You will ask the user to enter two teams. You don't have to do any clever string mapping. Assume the user will type the teams exactly as they are written in ***elofile.*** You will then state the % chance that Team A will beat Team B, then "simulate" the game by using a random number generator. (Easiest idea is generate number between 0 and 1. If less than win%, Team A wins, otherwise Team B wins.) You should only do an elo simulation here. You shouldn't do coin flip OR favorite always wins.

Simulating a single hypothetical game **should NOT** change the Elo rankings of the involved teams.

Option 2) Simulate the entire tournament

Ask the user HOW they want to simulate the tournament. Give them 3 options:

1) Coin flip -- every game is 50/50 odds
2) Elo -- every game is randomly decided but considering Elo
3) Favorite wins -- The team with the higher Elo rating always wins. If two teams have the same Elo, pick one randomly.

You will then simulate every game in games.csv in order. **DO NOT EDIT games.csv** during this option. Only display the output to a file or to the console if specified. Games.csv should be unchanged after running this option.

Simulating a tournament game **should not** change the Elo rankings of the involved team for the rest of that simulation run.

You are encouraged to implement other simulation models if you would like, but these 3 must be present.

Your system will simulate the bracket and generate and print an output of the format found in outputExample.txt. The file must list each team and their win-loss record (it does not have to list their elo) in that single tournament simulation. The team win/loss should not be cumalative of all simulations, only the last tournament simulation (as a result, all but 1 team should have exactly 1 loss, and the last team should have zero losses). Then, each game should be listed, showing both the participants and the winner. The game numbers must match up with those found in game csv files.

Option 3) Change a single team's elo rating

In this option, prompt the user for a team; assume the user has to spell it exactly like it is in the resource csv/json file including white space, but ignoring case. If a team is misspelled or doesn't exist, just prompt the user again until they enter a correct team name.

Then, ask the user to enter a number for that team's elo.

From there, all future simulations must use that team's new elo. For example, if I change West Virginia's elo to 999999, then they should almost certainly win every simulated tournament and game after that. Note that you should be allowed to change a single team's elo multiple times.

Option 4) Exit

Exit the system and flush and close the log file.

Design Patterns

In this project, you must implement the following 5 design patterns:

1. **Three Tier Architecure** must be used. There is no GUI here, but you can use a command line user interface. An abstract example was covered in class and can be found on canvas.
2. **Observer** - Use the Subject.java and Observer.java files uploaded to Canvas. You are welcome to modify these.
3. **Factory** - You may use either a concrete or abstract factory (or even a combination as discussed in class)
4. **Strategy -** At some point you should use a Strategy design pattern

5. **Singleton** - Use a singleton to implement logging.

You may implement other design patterns as well.

Tips on using design patterns:

- Consider that you have a bracket object made up of game objects. The bracket object would need to know who won each game to modifying who is playing in the next bracket game. Additionally, teams need to monitor who wins a game to update their own win-loss record. So two different objects need to monitor what happens with game, and need to do different things as a result of the winner being chosen. Which design pattern allows for this behavior?
- Consider that you can have either a JSON or a CSV file. Additionally, as noted earlier, you won't get the JSON file until one week before the homework is due. Which design pattern will allow us to add the JSON implementation most easily.
- Each bracket can be simulated one of three ways: 1) Coin flip 2) Favorite always wins 3) Elo Simulation. Which design pattern can allow us to do this at run time?

We will not answer questions on Piazza asking which design pattern to use for what. Additionally, you don't have to use the design patterns specifically in the way the question above says you have to. However, the choice of how to use it must be made on your own.

Write-up

In addition to implementing the application, using the patterns described above, you must create a PDF document in which all of the following are answered:

1. In your 3-Tier architecture, which classes comprise the Presentation Layer? the Controller Layer? the Date Layer? (Not every class has to be one of these 3, but list all the ones you think fit into these classes)
2. Which classes/methods implement the factory pattern? Is this concrete or abstract?
3. In which classes/methods did you use the Observer pattern? Which classes/interfaces were the Observers and which were the Subjects? Include the class names and line numbers that indicate that you used the pattern correctly.
4. In which classes/methods did you use the Strategy pattern? Why did you choose those classes/methods? Include the line numbers that indicate that you used the pattern correctly.
5. Compare this program to previous work you've coded before we covered design and design patterns (such as HW1). Do you feel that this code is more maintainable? Why or why not? Be completely honest here, you can say you felt it was a waste of time. But also consider your time spent not just developing, but testing and debugging as well.

Submission

All deliverables are due by **Monday** March 26, at 5:00 p.m. The usual late policy applies (10% per 24 hours up to 1 week late).

Submit your assignment through a single zip file on Canvas. This Zip file should take the form of "yourPennkey.zip", i.e., mine would be "paulmcb.zip." You may submit multiple times but only the last submission will be graded.

Academic Honesty and Collaboration

You must work on this assignment alone.

You may discuss your implementation strategy and observations with other students, but you absolutely must not share code or answers to the writeup questions.

Although you can, of course, look online for help with the Java API and things like that, you should not be receiving any help from outside sources, including students not taking this course or online resources.

Failure to follow these policies will be considered academic dishonesty, and you will receive a grade of 0 on this assignment.

If you run into problems, please ask a member of the teaching staff for help before trying to find help online!

Grading:

- Functionality - 25%
- Write-up - 15%
- 3-tier Archicture - 20%
- Strategy Patterns - 10%
- Factory Pattern - 10%
- Singleton Pattern - 5%
- Observer pattern - 15%

Extra Credit Opportunity

There will be an extra credit opportunity that involves you generating your own Elo file. This will be posted on Piazza Sunday night (as it requires the bracket to be set), and will be discussed in class on Tuesday.