Griffen Marler
Pete Tucker
CS 372
January 24 2019

Final Project Write Up

For Iain and I's final project, we were successfully able to design and implement a simple NBA game simulation application with a user interface. As expected, we underwent a few changes from our initial functional in order to successfully complete our program.The first change from our initial specification is that we got rid of pulling in the data of field goal percentage, three point percentage, and free throw percentage completely from online. We did this because we figured the game simulation would be easier to pull off if we were to just use the makes and misses from each team and generate a random over/under range for these categories in which the simulation generated. The simulation then each time took this random range of makes and misses and calculates the teams shooting percentages for the given game, hence getting rid of the need to pull this data from online. It is also worth mentioning that at first we were creating our randomly generated three pointers and field goals in a separate function in our original code, but noticed we were getting unusually high scores. This is because a field goal is scored as any two or three point attempt, but we thought it was only two point attempts. This caused us to get rid of our Gen3PTM function in our initial interface and only use the GenFGM function to create both 2 pointer and 3 pointer makes and attempts for each team.

User Interface wise, we were able to successfully meet our goal of having an interface consisting of three main sections. We ended up using a lot of JLayered Panes and three different panes to pull this off. Different functions to call each of the frames were created which helped us

seamlessly transition between the screens that we needed to. We had our title page with a background image of all 30 NBA teams, and after this took the user to the team selection page which held lists on each side of all 30 NBA Teams, plus two East and West All Star teams which was a feature coded in by us. In the future, we hope to add more fun teams like this to the interface. One thing we did not do that we said we were going to do in the functional specification is prompt the user to select teams, and this is something we could add in the future. We also did not use a splash screen loading visual as the program completed the simulation, but I believe our program is fast enough that the splash screen is unnecessary. Both Iain and I were extremely happy with the way our interface turned out.

As I look into the code, a good amount changed from the initial specification. Our NBATeam class did not end up being abstract. The NBATeam class ended up storing private variables holding the vital statistics that were pulled in from online using our FileReader class. We added functions String[][] genStats(), and String[] genColNames() to help us with the implementation of our JTable in the user interface to display statistics. The NBATeamSelect class extended the NBATeam class as planned, inheriting information from its partner class while at the same time implementing the GenStats interface. The NBATeamSelect class also added the functions String[][] genGameStats() and String[] genGameColNames() to implement the JTable in the simulation interface pane. A setTiebreaker function was also added to break the tie if two teams were randomly generated the same amount of total points. Our FileReader class did not contain very much detail in the original specification, but was implemented using Matcher and Pattern classes using two custom created regular expressions. I was responsible for this part of the project and it was definitely was one of the more difficult parts of the project for

me. However, I am glad we stuck with the idea of pulling information from online as I learned alot about creating regular expressions and pulling information off the web in general. It was very tempting at times to give up and ask you if I could instead use an old CSV file of stats that I found online. One more thing that we added that was not in the original specification was a league class that held an ArrayList of the created teams and the FileReader. The League class also has a genTeams method.

List of concepts used from the class:

- ArrayList
- Interfaces
- Extend
- Swing GUI Components
- Reading File From Online(Pattern/Matcher/URL)
- JavaDocs
- Exception Handling

Things to do in version two of the project:

- Support for more sports/leagues
- Support for previous years(not hard to implement)
- Better algorithm for simulation could lead this to being a convenient and fun to use betting tool.
- Look into pulling from NBA API

Ultimately, I had a great experience working on this project and enjoyed working on it with Iain. I was mainly responsible for the FileReader class, and Iain created most of the NBATeam, NBATeamSelect, and GenStats interface. The League class was created together, as well as the GUI class and that is what really took up the bulk of our time. It was awesome having Iain as a partner as we could coordinate our schedules and whenever we worked on this project we were together which helped a lot as we could reference each other for simple bugs and troubleshooting. This project gave me great confidence and excitement with the Java language and I look forward to diving deeper into this language as I go throughout my programming career.