StatTracker 3000: User Manual

Ayush Goyal, Nikhil Raman

Overview:

What follows in this document is a user manual for executing and using the supplied code for our homework 5 project. The project consists of 3 files, a Player class, a parsing class to get the necessary data, and a User class, user-facing class with a main method that can interact with a given user.

Running the Program:

In order to run the StatTracker 3000 make sure you first do the following :

- Put or import all three provided files into the same directory (package)
- Also download and place the public jsoup-1.8.x.jar external library in the same parent directory and/or in the associated build path for the project (the parsing class uses jsoup)

Next, simply compile and run the User.java class.

IMPORTANT NOTE: Due to how data-heavy and computationally-heavy this project is by nature, upon running the program the initial data retrieval takes approximately **250 seconds** (4 minutes). After this starting up time, the program responds to user input very quickly. It is also important to ensure that there is a strong internet connection throughout this pre-processing otherwise the connection will timeout and the stat-tracker will not work.

If you have any issues running the program, it is almost certainly an internet connection issue. Make sure you have internet.

The program handles all user interaction and input through standard in and standard out (in Eclipse: the console). Once the pre-processing has finished, you will be prompted with an initial menu.

Using the Program:

The program will successively prompt the user with different menus and options. In order to select an option, please type the number associated with the option and press enter.

You will first be prompted with this menu:

> Welcome to the stat tracker 3000!!
> Here is the Menu, choose a menu number!:
> 1. Search for a player's stat
> 2. Get the ten players who play more than 20 minutes a game, average more than 5 assists a game with the lowest turnover to assists ratio
> 3. Get the ten players with the most points per 36 minutes played
> 4. Get the ten players with the highest points per million dollars salary
> 5. Quit

Option 1: If you choose option 1 then you will next be asked the question: “What player?”. Please answer the question by typing in the desired player name with proper capitalization (E.g. Omer Asik). Note that the stat-tracker only supports NBA players currently on a roster (active players). Only properly spelled, full, and capitalized names will be accepted.

After typing in the name, you can choose from a list of 18 stat options via this menu:

> What stat, choose a number?
> 1. games played
> 2. minutes
> 3. field goal percentage
> 4. three point percentage
> 5. ftPercentage
> 6. rebounds
> 7. assists
> 8. blocks
> 9. steals
> 10. fouls
> 11. turnovers
> 12. points
> 13. free throws made per game
> 14. free throws attempted per game
> 15. field goals made per game
> 16. field goals attempted per game
> 17. three pointers made per game
> 18. three pointers attempted per game

After you choose a number, you will be prompted for a year. Type in the year properly (E.g 2016).

Then you will receive an output such as: “Omer Asik's field goal percentage in 2016 : 0.533”

Finally you will be returned to the main menu.

Option 2: If you choose option 2 on the menu, then you will be presented with the ten players who play more than 20 minutes a game, average more than 5 assists a game with the lowest turnover to assists ratio. The purpose of this is to get the best value point guards in the league. We measure efficiency in this stat category as assists to turnovers since a good point guard should be able to distribute the ball well without turning the ball over.

Option 3: If you choose option 3 then you will get the ten players with the most points per 36 minutes played. This is a measure of the players that provide the highest scoring adjusted for the amount of time they play. This statistic is important because we want to see the players that may score a lot of points but perhaps do not get as much time to play currently. Additionally some players such as stephen curry average a lot of points but also do not play the whole game. Moreover there are lots of players who score a great deal but also play a significant amount of minute and may not actually be all that efficient. Hence we adjust this raw stat for 36 minutes.

Option 4: If you choose option 4 then you will get the ten players with the highest points per million dollars salary. The purpose of this is to obtain the players that are the best value players. A general manager may be seeking high scoring but would like players that can fit into the budget well. Hence this stat gives you a measure of high production yet cheap players.

Option 5: This option allows the user to exit the program.

StatTracker 3000: Project Summary
Ayush Goyal, Nikhil Raman

Final Notes:

Taking too long: The reason the initial gathering of information takes so long when the program is run is because the data parser searches through (and takes the time to connect to) a number of of different web pages. Usually this process takes under four minutes but if for some reason it is taking longer than this, then I recommend commenting out the following code found on the given line numbers in the getPlayerData() method in the ParsePlayerData class:

```
113
114         /* ---------- --------- GET PLAYER SALARIES ------------------------ */
115
116         // GET PLAYER SALARIES  --- Highlight and Comment out section below: cmd+ctrl+/
117
118
119         Document infoDoc = null;
120         try {
121             infoDoc = Jsoup.connect(playerInfoURI).get();
122         } catch (IOException e) {
123             System.out.println("unable to connect");
124             e.printStackTrace();
125             return null;
126         }
127
128         Elements salaryInfo = infoDoc.select("li.first.last");
129
130         if (salaryInfo.size() > 0) {
131             String salary = salaryInfo.first().child(0).child(0).child(0).text();
132             double salaryNum;
133             try {
134                 salaryNum = NumberFormat.getNumberInstance(java.util.Locale.US).parse(salary.substring(1)).doubleValue();
135                 x.salary = salaryNum;
136             } catch (ParseException e) {
137                 // TODO Auto-generated catch block
138                 e.printStackTrace();
139                 return players;
140             }
141             //System.out.println(playerName + ": " + salaryInfo.size() + " - " + salaryNum);
142         }
143
144         /* ------------ FINISHED GETTING PLAYER SALARIES ----------------------- */
145
```

Commenting out this section will drastically speed up the program but Option 4 in the first menu will no longer be available.

Inputs: Our program is an NBA stats program. We are only concerned with the NBA and its active players. We provide all stats and information that are enumerated in the displayed menus. Bad inputs are handled by returning the user to the main menu.

Thanks!!