**Project Proposal**

Project Title: NBA myLeague

Description: My project, NBA myLeague, is a NBA Season simulating personal league program that allows the user to simulate head to head games between 2 teams, simulate a bracket of their choosing, build the 15 player rosters for the teams and switch players around teams using a salary cap limit.

Intended User: The users this program is aimed toward consist of people who watch NBA, play basketball themselves, or have any type of interest in the game of basketball.

This program will allow users to build a even higher knowledge of the game of basketball as they simulate games, see stats, and build custom teams giving them experience of how great teams are truly built.

The program will require a API to retrieve up to date player stats and team and player salary data from the NBA and some type of file system to store program data for when user exits out the program and loads it back up. It would also require a form of GUI.

**User Case Analysis:**

The program would start by prompting the user to either “Create a new League” or “Load a Saved League”. Loading a save would require User to select a file they have saved on their machine, which the program would pull the data from. First run, user will Create a New League, which will take them to a menu.

Menu would have choices like View Teams/Player which would allow user to view each team, the teams salary cap, the teams players and their season stats, and each players salary.

Change Rosters which would allow user to pick a amount of players and switch them with the same amount of players from another team without going over the salary cap which will be shown or completely randomize the rosters of all teams which the program will automatically follow teams salary caps.

Simulate Games, which would take user to screen that lets them either simulate a head to head game between 2 teams they choose. A head to head simulation would display the final score of the 2 teams and simulated game stats for each player on both teams.

The program would allow the user to save the league allowing them to load it back up when they wish to.

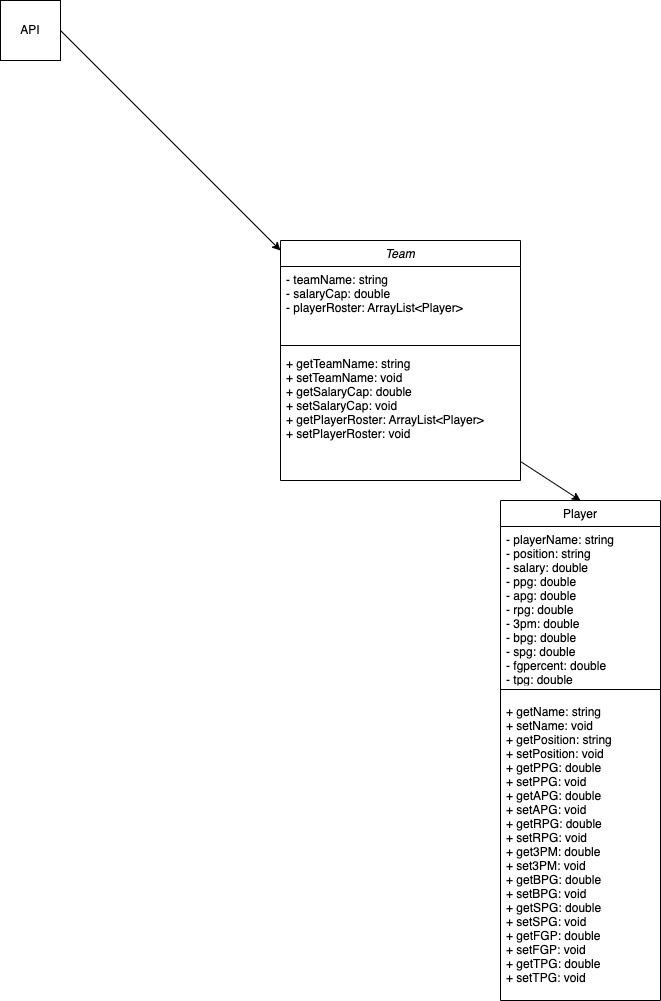
**Data Design:**

My program heavily revolves around the data of players, teams, player stats and salaries, and team salary cap.

Data would be pulled from API and stored into the respective objects and attributes.

The best way to represent this data would be in Arrays and ArrayLists. Teams, Players, Roster, TeamStats would be separate objects which would each have their own arrays. Teams would have attributes such as Team Name, Salaries, Player Roster which would be a ArrayList of type Player. Players would have attributes like name, position, and each individual player stats. A roster would be made up of an amount of players whose data can be accessed through it.

The stats date from the players objects would be compared with other player stats to simulate the games. We can also use standard deviation and averages to simulate probable main stats for each player and team. All simulated stats would be shown but the winner is determined by final scores.



**UI Design:**

Create New League takes User to Menu

View Teams/Players:

Allows User to choose the Team they wish to view through a dropdown, and then a list of all the players on that team and their stats

Trade Players:

Simulate Head to Head:

**Algorithm:**

In java, create a classes called Team, Player, and a class for each GUI page

In Team, create instance variable for teamName of type string, salaryCap of type double, playerRoster of type ArrayList<Player>

Create getters and setters for each instance variable

Getters will return the variable and setters will pull a parameter of the same type and set it to the instance of the variable using this.variablename

In Player, create instance variables for playerName type string, position type string, and all the corresponding stats in UML

Create getters and setters for each of these instance variables

In the Main class, create a function called loadLeague() that will load up a user saved file and load all the data into the objects and their arrays that is ran when user clicks the Load League Button

If user clicks Create New League button, initialize a new ArrayList<Team> object.

Initialize the API to extract the data

Pull each piece of data from the NBA API and use the setters from each object class to set that data to the corresponding variables in the Team arraylist

Pull salaries for HTML websites and store into Team ArrayList object

Create a menu for the user to choose 4 options, View Teams/Players, Trade Players, Simulate Head to Head.

If user chooses View Teams and Players, print a dropdown with names of all teams

Whichever team the user chooses from the dropdown, print the salary cap, team name, and all the players and their stats for that team in a GUI table

If user chooses to Trade a Player, print a dropdown and make them choose a team, print the teams salary cap and each of the players and their stats and salaries in a new GUI table

Create 2 other dropdowns that list all players from a team and allow user to select to multiple players from 2 teams. One dropdown for each

Once they click submit, print the names and salaries of the players they wish to trade and display it. Create a reset trade and submit trade button.

If the user clicks submit trade and has inputted players whose salary will cause one of the teams to go over their salary cap, the trade would not go through.

When the user inputs the names of the players they wish to trade their other players for, their salaries must balance out so neither team’s salary cap goes over the limit displayed.

If user chooses Simulate Head to Head, the program will print 2 dropdowns of teams and make user select 2 teams and simulate. The program will then simulate the game by getting the roster of each team and generate stats for that game using random, the average stats, and a standard deviation to find each teams stats and final scores. Finally, displaying each teams final scores and each players final game stats on both teams.