

# **COMP132: Advanced Programming**

## **Programming Project Report**

**NBA Game Simulation Desing and Development**

**Ahmet Koca 0076779**

**Fall/2023**

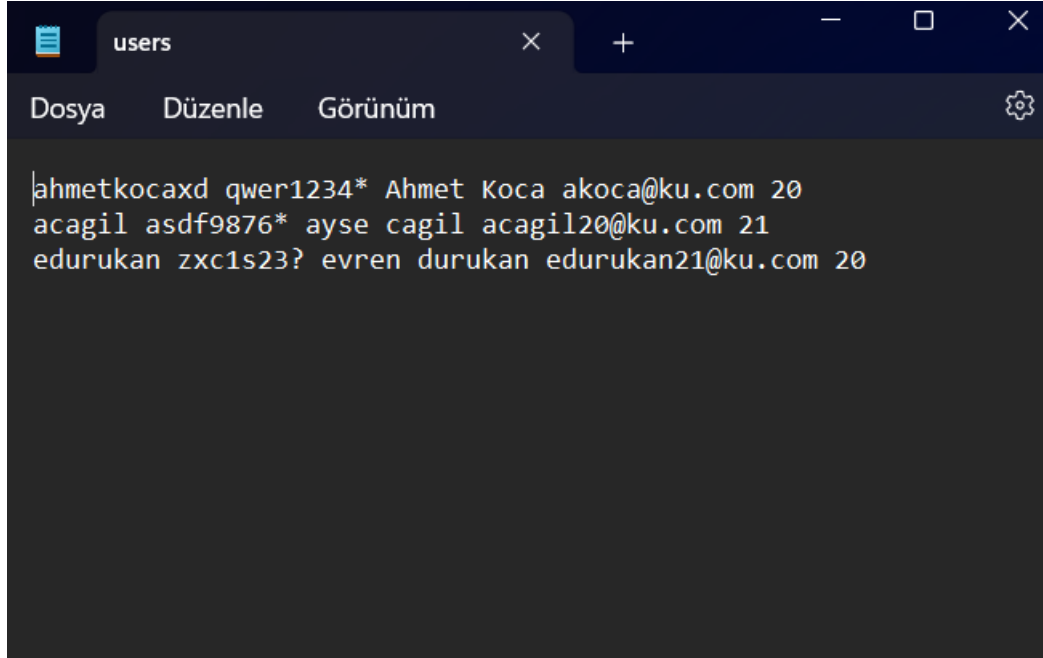


# Part 1

---

## 1. General Demo Information:

In my application, user's information is stored in a text file named "users.txt"

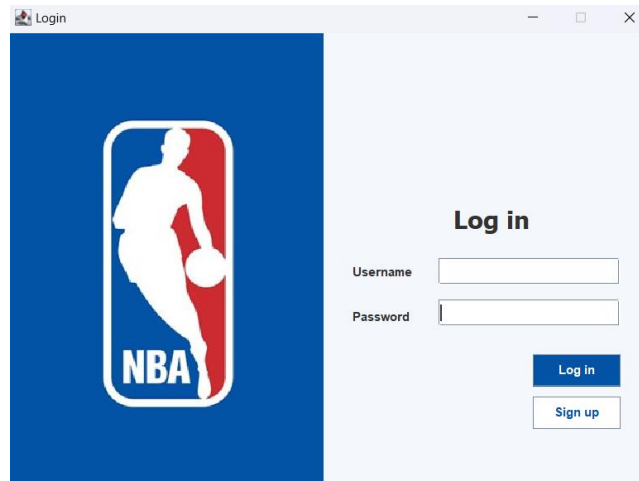
A screenshot of a text editor window titled "users". The window has a dark theme and a menu bar with "Dosya", "Düzenle", and "Görünüm" options. The text content is as follows:

```
ahmetkocaxd qwer1234* Ahmet Koca akoca@ku.com 20  
acagil asdf9876* ayse cagil acagil20@ku.com 21  
edurukan zxc1s23? evren durukan edurukan21@ku.com 20
```

**Figure 1: users.txt sample**

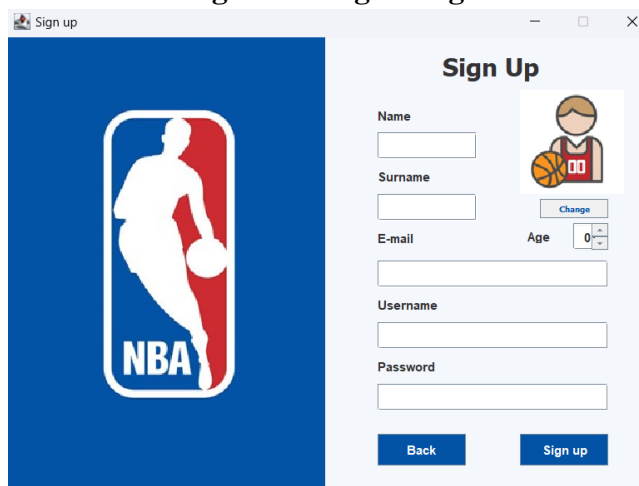
When users.txt file is processed in the project, each line is separated by whitespaces and necessary fields are checked. Each element then would store information of {username, password, name, surname, email, age}.

## 2. Application usage information



The screenshot shows a web application window titled 'Login'. On the left, there is a blue vertical banner featuring the NBA logo. The right side of the window has a white background with the heading 'Log in'. Below the heading, there are two input fields: 'Username' and 'Password'. To the right of these fields are two buttons: a blue 'Log in' button and a white 'Sign up' button with a blue border.

Figure 2: Log in Page



The screenshot shows a web application window titled 'Sign up'. On the left, there is a blue vertical banner featuring the NBA logo. The right side of the window has a white background with the heading 'Sign Up'. Below the heading, there are several input fields: 'Name', 'Surname', 'E-mail', 'Age' (with a spinner), 'Username', and 'Password'. To the right of the 'Surname' field is a 'Change' button. To the right of the 'Age' field is a small icon of a basketball player. At the bottom of the form are two buttons: a blue 'Back' button and a blue 'Sign up' button.

Figure 3: Register Page

- **Sign up/Login Guide:**

When the application is started, the first screen that appears is log in page. While trying to log in, the application checks whether a user exists in user.txt file with the given username. Also, if that user exists, checks whether the given password is the same password with the information inside users.txt.

If user has no account, he can sign up by clicking sign up button next to the log in button. Once it is clicked, a register page shows up and waits for necessary fields to be filled. Each field also checks whether the given input is valid, for example checks if the e-mail is in the correct format. Also, user can optionally select a .png file for a profile picture. If he does not select a new profile picture, a default picture is set for the user.

**NBA**

**WELCOME AHMET KOCA**

Username: ahmetkocaxd  
 Name: Ahmet  
 Surname: Koca  
 Age: 20  
 E-mail: akoca@ku.com  
 Password: qwer1234\*

[Change Info](#)

[Change](#)

[Start](#)

**Figure 4.1: User Info Page**

**NBA**

**WELCOME AHMET KOCA**

Username: ahmetkocaxd  
 Name: Ahmet  
 Surname: Koca  
 Age: 20  
 E-mail: akoca@ku.com  
 Password: qwer1234\*

[Change](#)

[Cancel](#) [Save Changes](#)

[Start](#)

**Figure 4.2.: User Info Page**

- **User's Guide:**

After logging in, user encounters a page where he can see and change user information except for name, surname, and username. User can change this information by clicking change info button. Again, inputs to the fields should be valid in order to save the new information. Also, user can change his profile picture. Profile picture of each user is stored as -username-.png inside the src under pfpImage file. That is how it is accessed and modified.

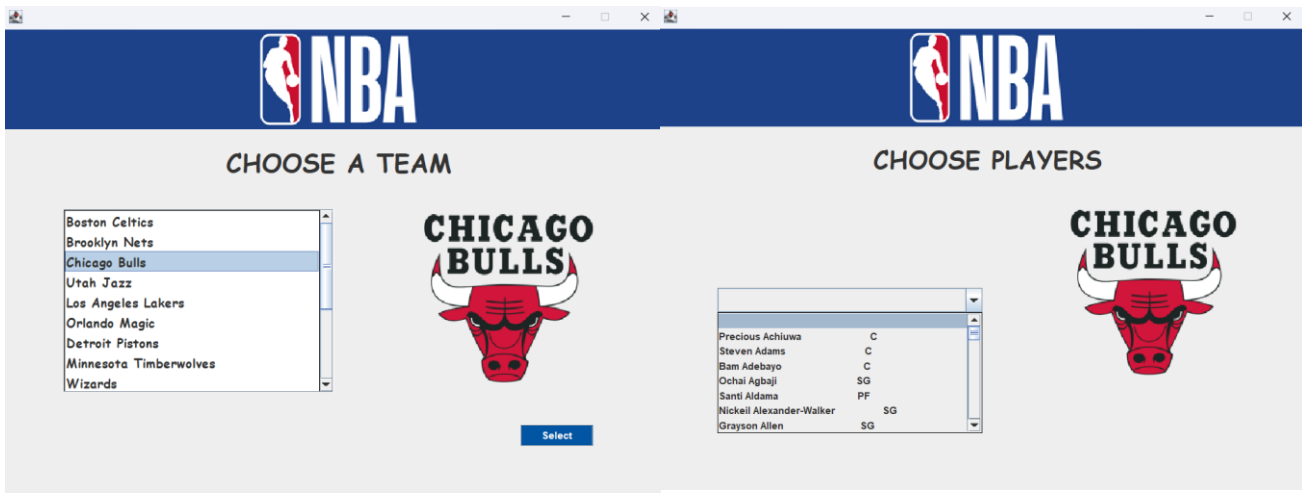


Figure 5: Drafting.

- **Drafting:**

The drafting process as soon as the user clicks Start button on user info page. A team selection page gets open and lets user to select a team among 16 teams with showing logo of each team. Once the user selects and clicks the select button, player draft page gets open, and process begins. Player information is obtained by reading the provided csv file and duplicates are eliminated and average stats of duplicates are assigned to one object of that player. Each team can have 5 players, each one having a different position. The team list is shuffled, and each team selects a random player. Once it is user's team turn, user selects a player and when it is end of the list, team list gets reversed. After every cycle, the option of players that is provided to user is updated with deletion of selected players and players that share the same position with user's team's player. The detailed logic behind the drafting process with computer side is described on the documentation inside the code.

Team Name	Win	Lose	Draw
Denver Nuggets	0	0	0
Minnesota Timberwolves	0	0	0
Brooklyn Nets	0	1	0
Atlanta Hawks	1	0	0
Miami Heat	0	1	0
Golden State Warriors	0	1	0
Utah Jazz	1	0	0
Toronto Raptors	1	0	0
Detroit Pistons	1	0	0
Los Angeles Lakers	1	0	1
Boston Celtics	1	0	1
Chicago Bulls	0	2	0
Orlando Magic	1	1	0
Indiana Pacers	1	1	0
Wizards	2	0	0
Portland Trail Blazers	0	3	0

Figure 6: Match making.

- **Match Making:**

After the drafting process, match making starts and each team plays at least a randomly determined number of matches which is at least 32 ( $2 * \# \text{of teams}$ ). User can observe each match played and the win-lose-draw stat on the screen. There is also view team button and a pause toggle button. Pause button pauses the match making and clicking it again resumes the match making. When view team button is clicked, user's team information appears on a new window and matchmaking automatically pauses. User also observe each player's detailed information by clicking on them on this window. After match making is over, a button appears that says start playoffs. Once it is clicked, it gets the 8 wins with the highest number of wins and shows the result of each match on the screen again. When it is over, user is informed which team has become champion. Unfortunately, I was not able to show the playoffs as a tree and show it on a different window.

- **How to check log information:**

The simulation keeps log information of users' login information (users.txt), users' profile pictures (under src under pfpImage), drafted teams with its player's (under teams file), season match records (Season.txt), win-lose-draw stats of teams after season (League.txt) and playoff results (Playoff.txt)

## Part 2

---

### Project Design Description:

- **Class relations:**

Each class is connected to each other, that is way I only create a login frame in main class. Login calls register class when sign up is clicked, or UserInfo class when login is clicked. UserInfo class calls TeamSelect class and TeamSelect class calls Season class. Starting from Login class, user information is passed throughout all simulation as a parameter in order to obtain and manipulate user information. Also, I created User, Player, Team and other classes in order to obtain and manipulate necessary fields under GUI components.

- **Inheritances, type hierarchies, interfaces, abstract classes:**

There are different subclasses for each position of Players and they are: PlayerC, PlayerPF, PlayerPG, PlayerSF, PlayerSG. These subclasses inherits from the abstract superclass Player. There are common attributes that each player has such as name, position, points etc. But each position have different data value weights and therefore a different way of `calculatingScore()`. Player has an abstract method `calculatingScore()` and it is implemented under the subclasses. I created double weight values and getters and setters of them in both superclass and subclasses. The reason is that all the operations I do in simulation with players is conducted with downcasting to Player. When I want to get the information of Player for PlayerInfo page, I want to call the getters and setters of weight variables from subclass.

Also, for simulation algorithms of SeasonSimulation and PlayoffSimulation, I used Runnable interface in order to implement `run()` method and use it with Thread (1). That way, I can determine when the simulation starts and I can create a pause-resume condition for the simulation.

Furthermore, I used WindowBuilder for and therefore, my GUI classes were subclass of JFrame.

- **GUI components:**

I used WindowBuilder for designing the GUI components. I choose to create a different class i.e., JFrame for different pages which have different purposes. Login, Register and *UserInfo* pages are mainly for creation, validation and presentation of users. After that point, other GUI components such as TeamSelect and Season have more complex methods inside in order to simulate the game. Also, the project has PLayerInfo and TeamInfo classes for user to observe information of during the match making.

- **CSV file processing details:**

For csv file processing, I created a class named PlayerSystem. This class has a `ArrayList<Player>()` class variable. There are methods for reading the provided csv file and removing duplicate players from the ArrayList. In order to read csv file and do some other operations on the player the game has, I create an object of this class and use its methods under the class TeamSelection, where I let the user pick a team and then choose players.

- **Your scoring algorithm:**

In each Player subclass, I created and initialized data value weights. Each one of the subclass has a `calculateScore()` method:

```
public void calculateScore() {  
    // TODO Auto-generated method stub  
    Random random = new Random();  
    int ptsValue = (int) (random.nextGaussian()*(pts/4) + pts);  
    int trbValue = (int) (random.nextGaussian()*(trb/4) + trb);  
    int astValue = (int) (random.nextGaussian()*(ast/4) + ast);  
    int blkValue = (int) (random.nextGaussian()*(blk/4) + blk);  
    int stlValue = (int) (random.nextGaussian()*(stl/4) + stl);  
  
    super.setScore((int) Math.round(ptsValue * ptsW + trbValue *  
trbW + astValue * astW + blkValue * blkW + stlValue * stlW));  
}
```

This method randomly calculates the score adding the data values by multiplying each of them with its weight. I used `random.nextGaussian()` in order to randomly get a normal value. I shifted the gaussian with the data value and scaled it with `dataValue/4`. That way, I



guaranteed that each random number is greater than 0, mean is data value and standard deviation is  $\text{dataValue}/4$ . Then, in team class, which has an ArrayList of players, I have a `calculateScore()` method which iterates over the ArrayList, calls the `calculateScore()` method of the player and adds them up. Finally sets the team score value as this final sum.

- **Your drafting method:**

The drafting method does not consist of only one part, and it is a little complex. It is described in detail as docstring in the code. But in order to summarize, I create a team list and shuffle it. One of them is user's team. Until it is user's team's turn, every team gets to select a player from the ArrayList called `availablePlayers`. When a player is selected, it is removed from the `availablePlayers`. Once it is user's team's turn, the loop terminates and waits for user to select a player from the GUI (`actionListener`). Once the user clicks on a player, the remaining of the teams pick a player and team list is reversed. After that, each time the user selects a player, the for loop starts and when it is user's team's turn, gets the selected player. After each loop, team list is reversed. That continues until each team picks 5 player. Additionally, there is a control mechanism for both user and computer to pick players with different positions each time. For computer, I implemented the method `private Player pickRandomPlayer(Team team)` and which picks a random player according to the aforementioned rule. For user, I update the JList I provide to user by removing the players that is picked and players that share the same position with user's team's players.

## References

---

1. **Interface Runnable**

<https://docs.oracle.com/javase/8/docs/api/java/lang/Runnable.html>