

**COMP132: Advanced Programming**

**Programming Project Report**

**NBA Game Simulation**

**Ahmet Taha Ekim, 79234**

**Fall/2023**



**Part 1**

**General Demo Information:**

**1.1 Users:**

* User 1:

Nickname: tahaekim

Password: A@a210620

Email: aekm@gmail.com

Name: Ahmet

Surname: Ekim

Age: 28

Profile Photo Path: /Users/a.tahaekim/git/Comp132-Project/comp\_project/src/profile photos/a.jpg

* User 2:

Nickname: cemrecsr

Password: A@a210620

Email: cemre@gmail.com

Name: Cemre

Surname: Cesur

Age: 25

Profile Photo Path: /Users/a.tahaekim/git/Comp132-Project/comp\_project/src/profile photos/default.jpg

* User 3:

Nickname: nazimata

Password: Mertemin2003@

Email: nata21@ku.edu.tr

Name: Nazim

Surname: Ata

Age: 31

Profile Photo Path: /Users/a.tahaekim/git/Comp132-Project/comp\_project/src/profile photos/l.jpg

* User 4:

Nickname: hasan

Password: Hasan61#

Email: hasan@gmail.com

Name: Hasan

Surname: Gunes

Age: 35

Profile Photo Path: /Users/a.tahaekim/git/Comp132-Project/comp\_project/src/profile photos/k.jpg

* User 5:

Nickname: bymanyaq

Password: Hanbayram#61

Email: han@gmail.com

Name: Bedirhan

Surname: Bayram

Age: 20

Profile Photo Path: /Users/a.tahaekim/git/Comp132-Project/comp\_project/src/profile photos/m.jpg

* User 6:

Nickname: talha

Password: Talha#25

Email: talha21@ku.edu.tr

Name: Talha

Surname: Saritepe

Age: 20

Profile Photo Path: /Users/a.tahaekim/git/Comp132-Project/comp\_project/src/profile photos/c.jpgList of users

**Application usage information**

* 1. **How to Start Application:**

The application's initiation process involves executing the main method within the "TestLogin.java" file. The file is situated at the following location: "/Users/a.tahaekim/git/Comp132-Project/comp\_project/src/gui/TestLogin.java". Upon running the application, it will launch the graphical user interface, presenting a simple login window.

* 1. **Sign up/Login Guide:**

The TestLogin application comprises Login and Signup pages, each serving distinct purposes. Upon launching, the Login page prompts users to input their username and password, with an option to reveal the password through the "Show" button. Successful login directs users to their profile page, while invalid credentials trigger an error message. The Signup page facilitates new account creation. Users select a profile picture, input main details (username, email, password), and provide personal information (name, surname, age). The "Submit" button validates user input, ensuring the uniqueness of usernames and emails, and verifies the correctness of entered information. In case of validation errors, informative messages guide users. A "Go Back" button allows users to return to the Login page without creating an account.

* 1. **Profile Page:**

The UserProfilePage in the TestLogin application serves as the user's profile management interface. Upon login, the page displays the user's profile picture, username, and essential details such as name, surname, email, and age. Users can easily identify and navigate their personal information. The profile picture can be changed by clicking the "Change Photo" button, which opens a dialog for photo selection.

To modify personal information, users can click the "Modify Personal Info" button, initiating a dialog where they can update their email, age, and password. The application ensures data integrity by validating inputs and providing feedback messages, such as requesting a valid email address or ensuring the user is at least 12 years old.

The page also facilitates team-related actions. Users can select a team by clicking the "Select Team" button, leading to the TeamSelectionPage. Additionally, the "View Team" button allows users to inspect their selected team's details. For users ready to start a tournament, the "Start Season" button opens the TournamentPage with the selected teams.

Signing out is easy with the "Sign Out" button, returning users to the login screen. Overall, the UserProfilePage class serves as a central hub for managing user-related actions within the application, providing an intuitive interface for users to interact with their profiles and associated functionalities.

* 1. **Drafting Guide:**
     1. **Team Selection;**

The TeamSelectionPage interface serves as a user-friendly platform for selecting a basketball team before entering the drafting process. Users are presented with a scrollable grid containing logos of available teams. The user can select a team by clicking on the corresponding logo, with the selection being visually highlighted. The team logos are displayed in a grid layout with appropriate spacing for clarity.

The team logos are accompanied by a set of navigation buttons located at the bottom of the interface. The "Go Back" button allows users to return to the previous screen without selecting a team, providing flexibility in decision-making. On the other hand, the "Select Team" button finalizes the team selection process and proceeds to the drafting phase.

Upon clicking the "Select Team" button, the system records the selected team's logo path. The chosen team is then associated with the current user, enabling a seamless transition to the DraftingPlayers interface. A confirmation message is displayed, informing the user about the successful team selection.

* + 1. **Player Selection:**

Upon entering the DraftingPlayers interface, users are greeted with a comprehensive table displaying available players for the drafting process. At the top of the screen, essential information about the current user, their team, and the ongoing drafting round is presented. The detailed player table provides key statistics such as name, position, total rebounds, assists, blocks, steals, and points. Users can interact with the table to select players for drafting, facilitated by the "Select Player" button. To signify the completion of the drafting process, users can click the "Done" button. However, the system ensures roster conditions for all teams before finalization, prompting users to revisit and correct any discrepancies. The "Go Back" button allows users to return to the previous screen without completing the drafting process.

In terms of player selection, users can interact with the player table to view detailed information about each player. Selection is made by clicking on the desired player's line. The drafting process consists of multiple rounds, with a maximum of 15 rounds. Each round involves the user selecting a player for their team and the system randomly assigning players to other teams. The player for the user's team can be selected based on his preferred position, while for other teams players are randomly assigned in accordance with the team's positional requirements. There is no system here that guarantees the user to have players in 5 different positions in the first five drafting rounds. This is done because the user is aware of this requirement. The drafting process ends when the user clicks on the "Done" button. If each team meets the roster requirements, a confirmation message appears. Additionally, upon completion, a log file is created for each team, listing the players selected for each position. These log files are then saved in the “teamlogs” directory, providing a comprehensive record of drafting results.

* 1. **Match Making Guide:**

The "Match Making" module serves as a comprehensive tool for simulating basketball matches within a graphical user interface (GUI).

Upon launching the application, the GUI initializes the tournament, managing the flow of both regular season and playoffs. The layout includes various components such as home and away team logos, real-time score displays, a match table, and a dynamic result output area.

During regular season simulation, teams engage in randomly generated matches, with detailed results dynamically updating in the GUI. The "startTournament" method initiates the simulation in a separate thread. User controls, particularly the "Pause" and "Resume" buttons, offer flexibility in managing the simulation, allowing users to pause, view details, and resume the tournament according to their preferences. Pausing the simulation triggers the saving of regular season results and table data to files. The "View Team" button allows users to inspect detailed information about a selected team, temporarily pausing the tournament for a closer look.

Following the regular season, the top 8 teams enter playoff simulation. Quarterfinals, semifinals, and the final are dynamically displayed, showcasing results and advancing teams. The tournament concludes with the final match, and the winning team is declared, with all results being saved for future reference. Regular season match results are saved to "match\_results.txt," playoff results to "playoff\_results.txt," and table data to "table\_data.txt."

**Part 2**

**Project Design Description:**

**2.1** **Class relations:**

In terms of user-related classes, the "User" class serves as the base class, featuring attributes such as nickname, password, name, surname, age, email, and a reference to a "Team" object. The "Team" class, in turn, encapsulates team-related information, including the team's name, logo path, a roster (a HashMap mapping positions to player lists), and performance statistics like wins, losses, and ties.

Within the "Player" package, there is a hierarchy of player classes, each representing a specific basketball position. The base "Player" class includes common attributes such as name, position, points, total rebounds, assists, blocks, steals, and a score. Subclasses like "Center," "PointGuard," "PowerForward," "ShootingGuard," and "SmallForward" extend the base "Player" class, inheriting its properties while specifying the basketball position.

The use of inheritance in the "Player" package adheres to a type hierarchy based on player positions, providing a clear and organized structure. This design allows for the extension and customization of player attributes specific to each position, facilitating code reuse. The "Team" class employs composition to manage a roster of players, utilizing a HashMap to categorize players by position. Additionally, the "Team" class features methods for adding players, calculating match statistics, and obtaining a random position from a predefined list.

# 2.2 GUI Components:

The GUI components play a pivotal role in crafting visually appealing and interactive user interfaces. In the context of Swing, a powerful library for building graphical applications in Java, various components are employed to create windows, panels, buttons, text fields, and other elements that enhance user experience1.

Consider the JFrame, a fundamental component serving as the main window or frame for the application. It encapsulates other GUI components, creating a container for the graphical interface. JPanel, on the other hand, acts as a container within the JFrame, facilitating the organization and arrangement of other components.

Text entry is made possible through JTextField, allowing users to input data, such as usernames and passwords. For secure password entry, JPasswordField is used, ensuring that password characters are concealed. JButton provides clickable buttons that trigger specific actions upon user interaction. Also, JLabels are utilized to display descriptive text, guiding users through the interface. The JToggleButton introduces a dynamic element, enabling the user to toggle between showing and hiding the password, enhancing flexibility and security.

Event handling is a crucial aspect of GUI programming, and ActionListener interfaces are implemented to capture and respond to user actions. For instance, when a login button is clicked, an ActionListener checks the entered credentials against stored data, while a signup button navigates users to a new page for account creation. The visual appeal is further enriched by incorporating images using ImageIcon, and the layout managers, such as null layout and grid layout, contribute to the proper alignment and arrangement of components within the GUI.

**2.3 CSV File Processing Details and Scoring Algorithm:**

The provided CSV file containing NBA player statistics for the 2022-2023 season. The heart of the operation lies in the PlayerCreator class, which diligently reads the CSV file, parsing each line to extract crucial player details like name, team, position, and performance metrics by using BufferedReader.2 This class parses the player objects coming from the csv file according to their positions and determines which subclass of the player class they belong to. Also, while reading the file PlayerCreator undertake the duplicate entries by selecting TOTs.(Figure 1) Additionally, when the entire file is processed, it returns an arraylist with all the players.

metin, ekran görüntüsü, yazılım, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 1

The scoring logic is in the weightCalculator method, where a weighted value is computed for specific player statistics. This calculated score considers various factors, such as points, total rebounds, assists, blocks, and steals. Each category is assigned a specific weight (N) to reflect its relative importance in player performance assessment.(Figure 2) The algorithm then introduces an element of randomness by generating a random double within a specified range. In this way, when each player is created, the score is generated randomly.

**metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Figure 2

**2.4 Drafting Algorithm:**

The DraftingPlayers class implements a comprehensive player drafting algorithm for a basketball simulation game. The graphical user interface facilitates user interaction in selecting players for their team, while also automating the player selection process for other teams. The algorithm operates in rounds, with each round representing a turn for each participating team to make a player selection. The drafting rounds are capped at 15, ensuring a fair and dynamic drafting process.

For the user's team, player selection is user-driven, allowing the user to choose a player based on displayed information in a table. The algorithm ensures that the user's team adheres to roster conditions, rejecting selections that do not satisfy the requirement of having players for all five positions. In contrast, for other teams, the algorithm employs a randomized approach to player selection. The randomness is introduced differently in the first five rounds compared to subsequent rounds, adding variety to the drafting process. In the initial rounds, teams are assigned players based on their preferred positions, enhancing strategic diversity in team composition. As the drafting progresses beyond the fifth round, randomness is introduced without considering position preferences, further diversifying team compositions.(Figure 3)

ekran görüntüsü, metin, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 3

The algorithm continuously updates the graphical user interface, displaying real-time information about the drafting progress, selected players, and the current round. Additionally, a team log is generated, summarizing the drafted players for each team, and saved as a text file for future reference.

# 2.5 Regular Expressions:

In the user profile page, regular expressions (regex) are employed for validating and updating user information. Specifically, regex patterns are used to enforce certain constraints on user input, ensuring that the entered data adheres to predefined rules.3 The isValidPassword method utilizes a regex pattern to check if a new password meets the criteria of containing at least one letter, one digit, and one special character, with a minimum length of 8 characters. Similarly, the isValidEmail method employs a regex pattern to validate email addresses, verifying that they follow the standard format with proper characters and domain structure. These regex checks are essential for maintaining data integrity and security, providing a structured approach to input validation within the user profile page.

**2.6 Predefined Sample Images:**

The PhotoSelectionPage class represents a dialog for selecting a profile photo from a predefined set of sample images.4 The dialog features a grid layout containing buttons, each corresponding to a sample photo. Users can choose a photo by clicking on the associated button, and the selectedImagePath variable stores the path to the chosen image for later use. Upon selecting a photo, users can proceed to the next page or take appropriate actions, and a confirmation message is displayed. The dialog also includes a "Continue" button to facilitate the transition to subsequent pages. Additionally, if users attempt to proceed without selecting a photo, a warning message is displayed to prompt them to make a selection.

**References**

1. <https://help.eclipse.org/latest/index.jsp?topic=%2Forg.eclipse.wb.doc.user%2Fhtml%2Findex.html>
2. https://copyprogramming.com/howto/reading-csv-file-using-bufferedreader-resulting-in-reading-alternative-lines#reading-csv-file-using-bufferedreader-resulting-in-reading-alternative-lines
3. <https://www.w3schools.com/java/java_regex.asp>
4. <https://www.behance.net/awaisali13>
5. https://www.tutorialspoint.com/java/java\_documentation.htm