NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES ISLAMABAD
Object Oriented Programming Fall 2024
ASSIGNMENT # 03

Due Date: **18th Nov, 2024 (11:59PM)**

Instructions:

- Make sure that you read and understand each and every instruction
- Each question should have two files named as "23i-0001_Q1.h" and "23i-0001_Q1.cpp". Failing to do so will get you zero in that question.
- Keep a backup of your work always that will be helpful in preventing any mishap.
- Combine all your work in one .zip file.
- Declare functions outline.
- Name the .zip file as ROLL-NUM SECTION.zip (e.g. 23i-0001 B.zip).
- Submit the .zip file on Google Classroom within the deadline.
- Avoid last hour submissions
- Start early otherwise you will struggle with the assignment.
- You are not allowed to use any built-in functions.
- You must follow the submission instructions to the letter, as failing to do so will get you a zero in the assignment.
- All the submitted evaluation instruments will be checked for plagiarism. If found plagiarized, both the involved parties will be marked 0
- String or any other built-in datatypes or any built-in functions are strictly prohibited
- Make sure to make main() function of every question which demonstrate all functions mentioned in the question. Any function not shown in main() will not be marked during demo.

Fantasy Football League System
Object-Oriented Design Specification

System Design Team

1 Overview

This assignment involves building a comprehensive Fantasy Football League system by implementing several interconnected classes that demonstrate composition and aggregation relationships. The primary goal is to simulate a football season while managing teams and players in an organized, objectoriented manner.

Figure 1: Fantasy Football League

2 Restrictions and Assumptions

- You cannot use any hard-coded constants except those provided in the Constants class
- Any breach of these restrictions will result in significant penalties.

- All the work done needs to be shown via a simulation.

3 Core Classes and Relationships

3.1 Player Class

The Player class represents individual football players with the following attributes:

- Name (unique identifier)
- Age (must be 18 or higher)
- Position (goalkeeper, defender, midfielder, striker)
- Statistics (games played, goals scored, assists, tackles, interceptions, skill rating, etc.)

The Player class should provide methods

- Update and retrieve player statistics
- Reset statistics
- Get player information (name, position, etc.)

3.2 Team Class

The Team class owns a collection of Player objects. Each team has:

- Unique team number
- Team name (unique)
- Collection of players organized by position
- Team statistics (games played, wins, draws, losses, goals, etc.)
- Last five game results

Key features

- Teams maintain ownership of their players
- Players are organized by their positions within the team
- Team statistics are automatically updated based on game results • Teams track their last five game results

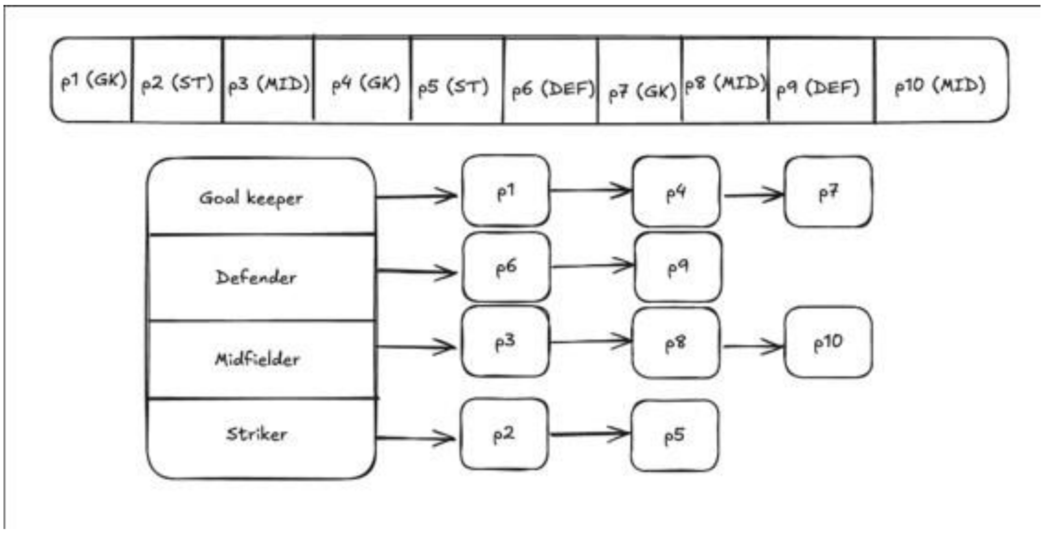


Figure 2: Player Saving overview

3.3 Season Class

The Season class manages Teams and their interactions. It includes:

- Collection of participating teams
- Dynamic game schedule
- Season leaderboard

Methods for

- Generating and managing the game schedule

- Handling game simulations
- Managing team rankings
- Delaying and rescheduling games

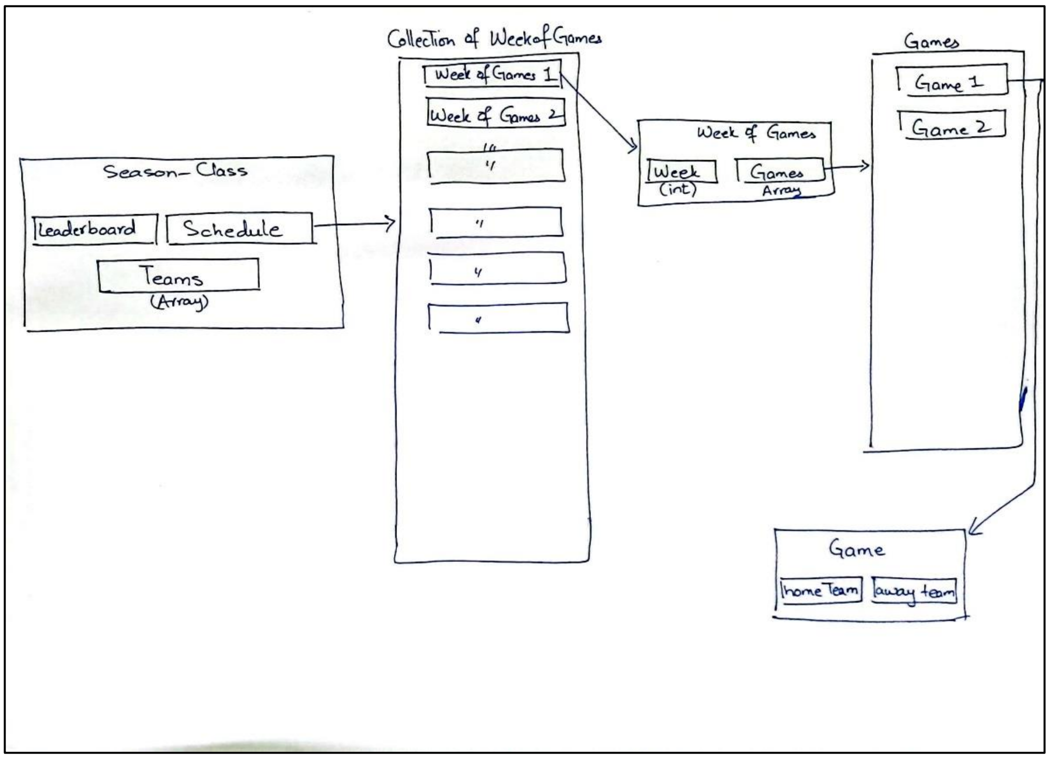


Figure 3: Flow of Season Make

sure to implement the all needed classes.

3.4 Award Class

The Award class works with the Season class to manage player achievements:

- Reference to the season
- Award criteria based on player statistics
- Number of recipients per team
- Player performance leaderboard

4 Key Relationships

The different classes will need to be interlinked using **composition** and **aggregation**. Identifying where what is needed is a problem you will need to evaluate.

5 Game Results and Statistics

Game results affect both team and player statistics:

- Win: 3 points
- Draw: 1 point • Loss: 0 points

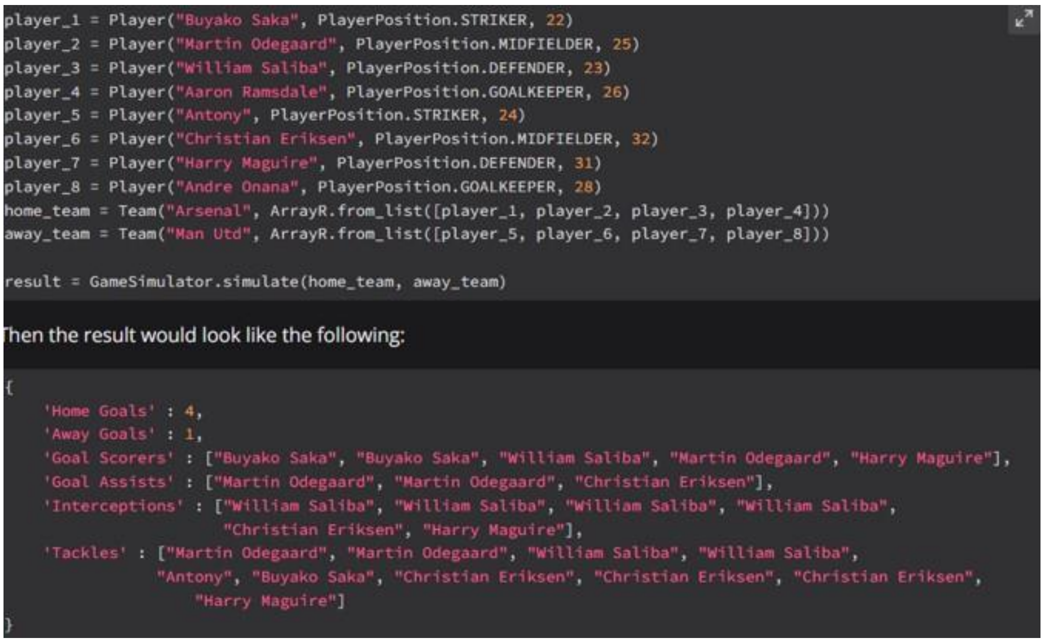
5.1 Team Statistics

- Games played
- Points
- Wins/Draws/Losses
- Goals (for/against/difference)
- Last five results

5.2 Player Statistics

- Games played
- Goals scored
- Assists made • Tackles
- Interceptions
- Skill ratings

This is an expected return value the goal scored, assists and so on may be randomized or done according to some logic you derive.



```
player_1 = Player("Buyako Saka", PlayerPosition.STRIKER, 22)
player_2 = Player("Martin Odegaard", PlayerPosition.MIDFIELDER, 25)
player_3 = Player("William Saliba", PlayerPosition.DEFENDER, 23)
player_4 = Player("Aaron Ramsdale", PlayerPosition.GOALKEEPER, 26)
player_5 = Player("Antony", PlayerPosition.STRIKER, 24)
player_6 = Player("Christian Eriksen", PlayerPosition.MIDFIELDER, 32)
player_7 = Player("Harry Maguire", PlayerPosition.DEFENDER, 31)
player_8 = Player("Andre Onana", PlayerPosition.GOALKEEPER, 28)
home_team = Team("Arsenal", ArrayR.from_list([player_1, player_2, player_3, player_4]))
away_team = Team("Man Utd", ArrayR.from_list([player_5, player_6, player_7, player_8]))

result = GameSimulator.simulate(home_team, away_team)
```

hen the result would look like the following:

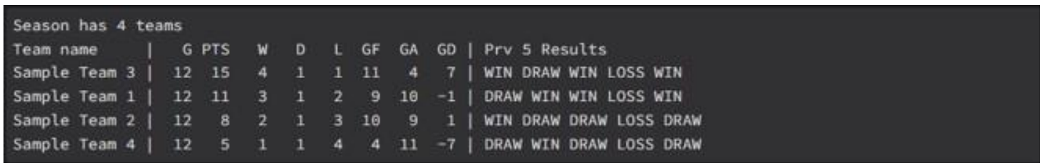
```
{
    'Home Goals' : 4,
    'Away Goals' : 1,
    'Goal Scorers' : ["Buyako Saka", "Buyako Saka", "William Saliba", "Martin Odegaard", "Harry Maguire"],
    'Goal Assists' : ["Martin Odegaard", "Martin Odegaard", "Christian Eriksen"],
    'Interceptions' : ["William Saliba", "William Saliba", "William Saliba", "William Saliba",
                       "Christian Eriksen", "Harry Maguire"],
    'Tackles' : ["Martin Odegaard", "Martin Odegaard", "William Saliba", "William Saliba",
                "Antony", "Buyako Saka", "Christian Eriksen", "Christian Eriksen", "Christian Eriksen",
                    "Harry Maguire"]
}
```

Figure 4: Expected return value for simulation function

6 Leaderboard Management

The season leaderboard should order teams by:

1. Points (descending)
2. Goal difference (descending)
3. Goals scored (descending)
4. Team name (ascending)



```
Season has 4 teams
Team name     |  G PTS  W  D  L  GF  GA  GD | Prv 5 Results
Sample Team 3 | 12  15  4  1  1  11   4   7 | WIN DRAW WIN LOSS WIN
Sample Team 1 | 12  11  3  1  2   9  10  -1 | DRAW WIN WIN LOSS WIN
Sample Team 2 | 12   8  2  1  3  10   9   1 | WIN DRAW DRAW LOSS DRAW
Sample Team 4 | 12   5  1  1  4   4  11  -7 | DRAW WIN DRAW LOSS DRAW
```

Figure 5: Leader Board

7 Schedule Management

The season schedule should:

- Support dynamic game scheduling
- Allow delaying games to later weeks
- Handle rescheduling conflicts
- Maintain game order integrity

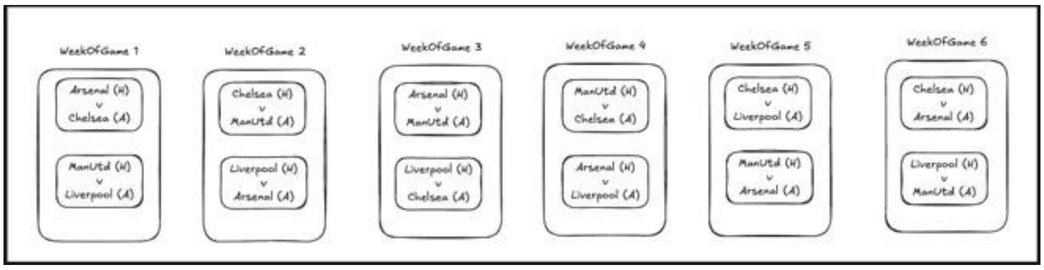

WeekOfGame 1
Arsenal (H) v Chelsea (A)
ManUtd (H) v Liverpool (A)
WeekOfGame 2
Chelsea (H) v ManUtd (A)
Liverpool (H) v Arsenal (A)
WeekOfGame 3
Arsenal (H) v ManUtd (A)
Liverpool (H) v Chelsea (A)
WeekOfGame 4
ManUtd (H) v Chelsea (A)
Arsenal (H) v Liverpool (A)
WeekOfGame 5
Chelsea (H) v Liverpool (A)
ManUtd (H) v Arsenal (A)
WeekOfGame 6
Chelsea (H) v Arsenal (A)
Liverpool (H) v ManUtd (A)

Figure 6: Caption

8 Error Handling

The system should handle:

- Invalid player ages
- Team size violations
- Schedule conflicts
- Invalid statistics updates
- Duplicate player/team names

9 Constants

The system uses several constant values:

- SEASON _LENGTH: Duration of the season
- TEAM MIN _PLAYERS: Minimum players per team

- TEAM MAX PLAYERS: Maximum players per team
- MAX NUM TEAMS: Maximum teams in a season