Author: Saivenket Patro K

Student ID: 220477422

Sports League Documentation

Documentation Overview

The Sports League Project is designed in a unique yet simple way to execute its problem in hand. The Documentation will help the reader understand the design principle and the implementation to simulate the English Premiere League Football.

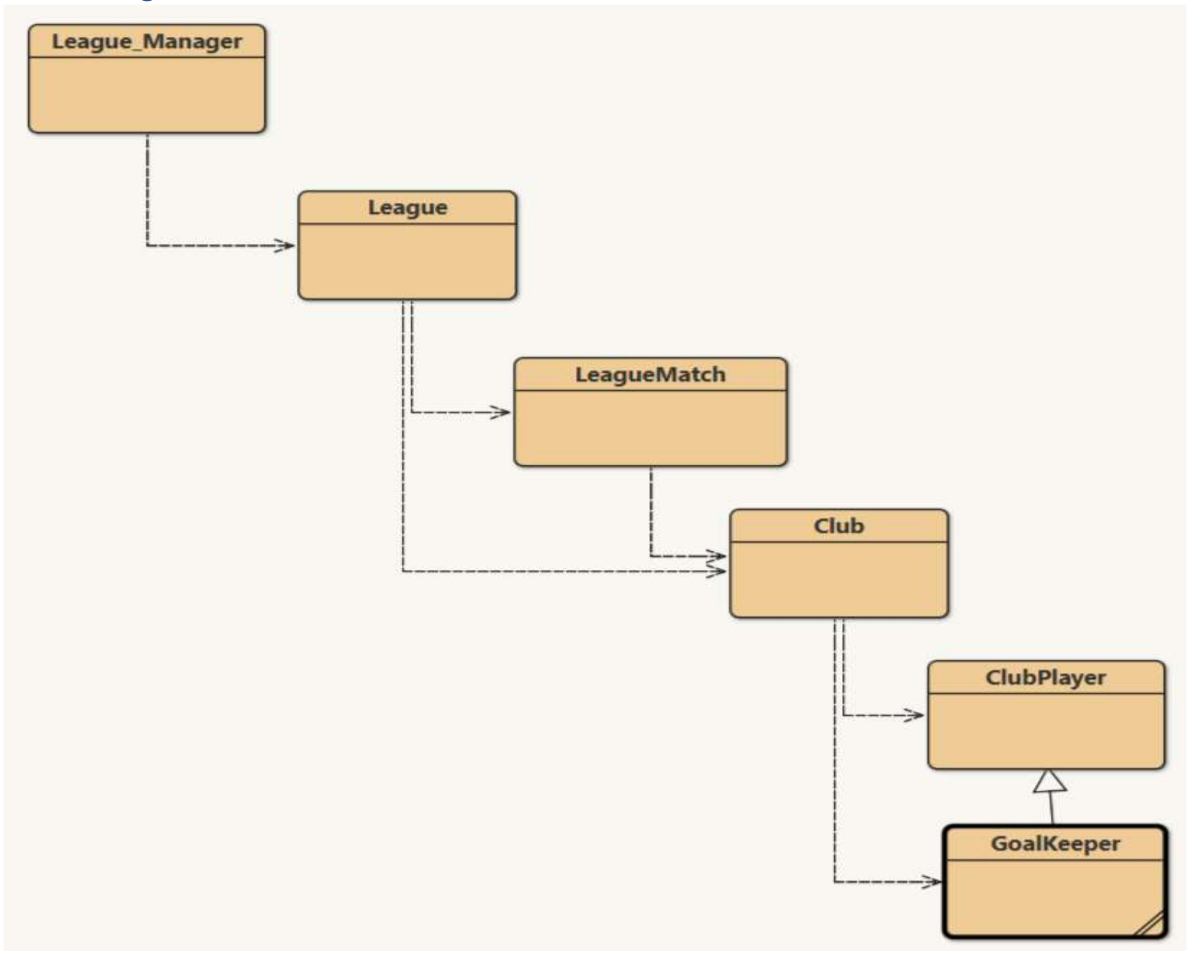
The documentation consists sections of:

1. Class Diagram
2. Class Diagram Overview
3. Design Principle
4. Project Folder Hierarchy
5. How To Run The Program
6. Explaining The Text Files For The Test Cases
7. Explaining The Test Case With Implementation

These sections will help in understanding the project.

Although, to keep in mind, the simulation has been demonstrated with constraints which will be highlighted in the Test Explanation Section.

Class Diagram



Class Diagram Overview

As the diagram states the flow,

The League_Manger is the driving class for the functioning of the overall project.

The League Class manages and handles the methods and functions of the LeagueMatch and Club class. It has a list of fixtures and clubs to store the information. The methods of this class are used to retrieve and compute information of the overall League.

The LeagueMatch Class helps in creating a fixture between two teams, where the teams are defined by the Club Class. It contains details about the fixture such as the time, venue, Home team and Away team, and their result. It also helps in stating results of a fixture.

The Club Class defines the overall characteristics of a team. It includes a Squad consisting of a Goal Keeper. It has information about the Club such as Goals Scored, Points Scored, etc. and also data about its players in the team. The Class uses the ClubPlayer Class to store its players and GoalKeeper to store its Squad GoalKeepers

The ClubPlayer class gives access to its personal information and its achievements attained during the match fixture.

The GoalKeeper Class is an extended Class of the ClubPlayer Class. It is defined as a Sub Class because it is a part of the squad but has an additional feature “Clean Sheets” as a Goal Keeper. While, it has the parameters such as goal scored, age, height, etc same as a ClubPlayer.

Design Principle

The Sports League project was designed to replicate as close to the Premiere League Football.

The classes have been given responsibilities as designed the real case scenario.

Let me elaborate,

The League consists of teams, and the designs different match fixtures between the clubs over a period in different locations. Hence, the League controls the data of Clubs and League Fixtures.

The League Fixture consists of two teams and has data about the clubs and the venue details.

The Club consists of its team and achievements of the squad. Hence the Club has access to its players and collects information about the club to project it to the League.

The Player in a Club is the smallest entity in the League and hence in the bottom of the class design. Where, a player can be defined by its personal characteristics and achievements.

Project Folder Hierarchy

Sports League (root folder)

- Java Class Files (root files)
- Sports League Documentation
- Doc
  - Java doc files for every class is present
- test_case
  - Fixture_Result.txt
  - Fixtures.txt
  - GoalKeeper CleanSheets.txt
  - Player Goals.txt
  - Squad.txt

How To Run The Program

The program is developed using BlueJ Version 5.0.2.

Step 1: Unzip the Project

Step 2: Open BlueJ and open the project folder Sports League. The Class diagram should be visible in the console.

Step3: Right Click on the League_Manager Class, and click on void main (String args []) option to run the program.

Step4: View the results in the output window

Explaining The Text Files For The Test Cases

All the txt files are designed in a certain format to run the program. If the user wishes to add or change the data please adhere to the format.

(Note: Ideally this is not the preferred method, for demonstration the format suffices)

1. Squad.txt - Squad size defines the number of players and is stated in the below line Following that is the Club Name and the next line is the player's name, following its year of birth and heigh in CMS. This combination of three is done for the size of the players.
   The size of the players and the players stated should be same for the demonstration. This txt file is used in Test Case B

2. Fixtures.txt – Match Id is the unique identifier for the match. It follows by time, location, home club and away club. Please be aware the name of the club has to be same as the Squad.txt. This txt file is used in Test Case C

3. Fixture_Results.txt – It starts with the unique match Id that is used in the fixtures.txt. In accordance to that, home and away, goals and result are to be defined.

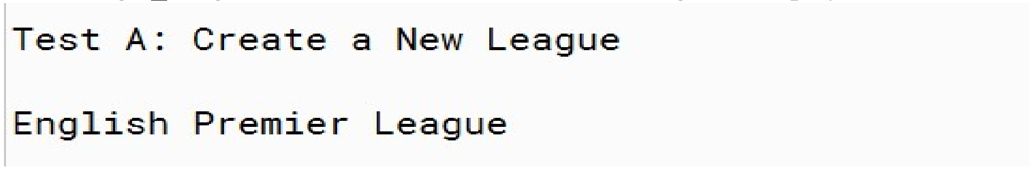
4. Player Goals.txt – the match id is defined here for the user to understand the fixture it corresponds to but not used in the program itself. Although, it is essential to stick to the format for seamless execution. Player name following the number of goals scored is the defined format

5. GoalKeeper CleanSheets.txt - the match id is defined here for the user to understand the fixture it corresponds to but not used in the program itself. Although, it is essential to stick to the format for seamless execution. Player name is sufficient for the program to understand.

Explaining The Test Case with Implementation

1. Test Case A

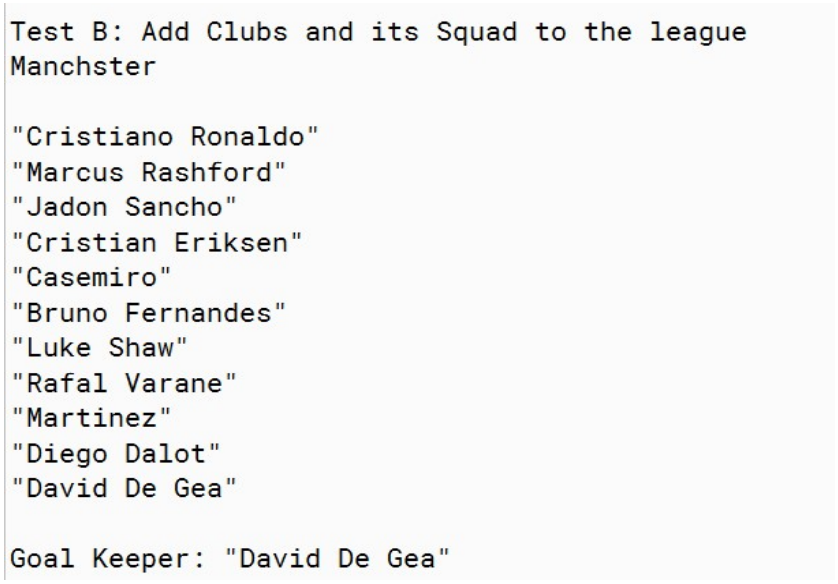
   It creates a new League object to define the English Premier League Football object.
   It uses the add_leagueName method to add the name of the league
   And, the get_LeagueName retrieves the name of the league to display

```
Test A: Create a New League

English Premier League
```

2. Test Case B

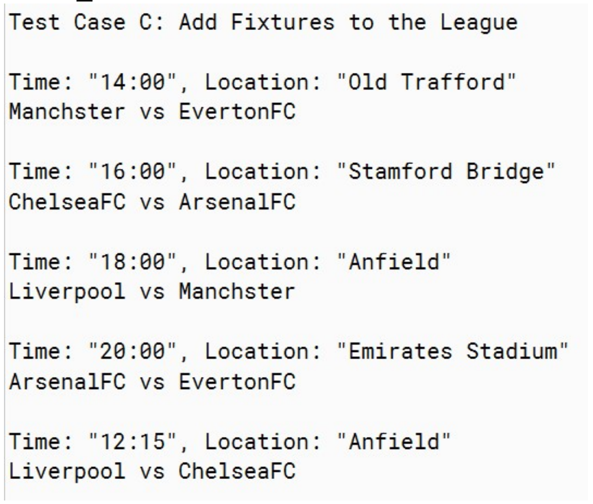
   It adds clubs and its player in this test case.
   We use the combination of FileReader with a Scanner Class to access a text file.
   The text file contains the club's name and its squad.
   Using the scanner object, we iterate over the text file lines to get the relevant information for the variables.
   Using the add_leagueClub method on the League object we add the clubs, where the method uses a Club object to create a club and store its club name. We use a for loop to iterate over the squad player of a particular size stated in the text file.
   The Players are added to the squad using the add_PlayerClub method.
   This method creates a ClubPlayer Object to create and store a new player.
   (Note: The Scanner object returns a String. Hence, for some variables the type has been changed from String to Integer to satisfy the method parameters)
   (Note: While the player is added to the squad, an internal check is done to validate if the player is not added in club before or present elsewhere using the check_clubPlayer method)
   (Exception: The date taken is a year of birth rather than date of birth)
   Individually, we also add the Goal Keeper using the add_goalkeeperClub method which creates a new GoalKeeper object to store the data

```
Test B: Add Clubs and its Squad to the league
Manchster

"Cristiano Ronaldo"
"Marcus Rashford"
"Jadon Sancho"
"Cristian Eriksen"
"Casemiro"
"Bruno Fernandes"
"Luke Shaw"
"Rafal Varane"
"Martinez"
"Diego Dalot"
"David De Gea"

Goal Keeper: "David De Gea"
```

3. Test Case C

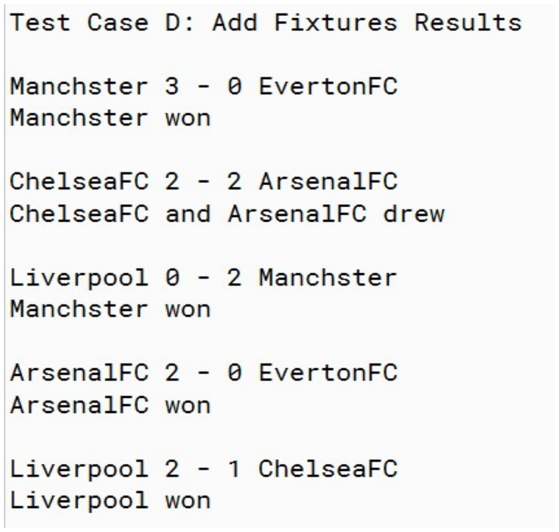
   This case explains the adding of fixtures to the league.
   Like Test Case B, we use the Scanner object to access the file.
   Use the while loop to retrieve the string of information from the txt file.
   Initiate the add_leagueFixture on the League object to add details of the league using a unique id called the match_id.
   The add_leagueFixture creates a new object of LeagueMatch and stores the information of the match.
   We then use the get_fixtureDetails to display a particular league fixture using the match_id.



```
Test Case C: Add Fixtures to the League

Time: "14:00", Location: "Old Trafford"
Manchster vs EvertonFC

Time: "16:00", Location: "Stamford Bridge"
ChelseaFC vs ArsenalFC

Time: "18:00", Location: "Anfield"
Liverpool vs Manchster

Time: "20:00", Location: "Emirates Stadium"
ArsenalFC vs EvertonFC

Time: "12:15", Location: "Anfield"
Liverpool vs ChelseaFC
```

4. Test Case D

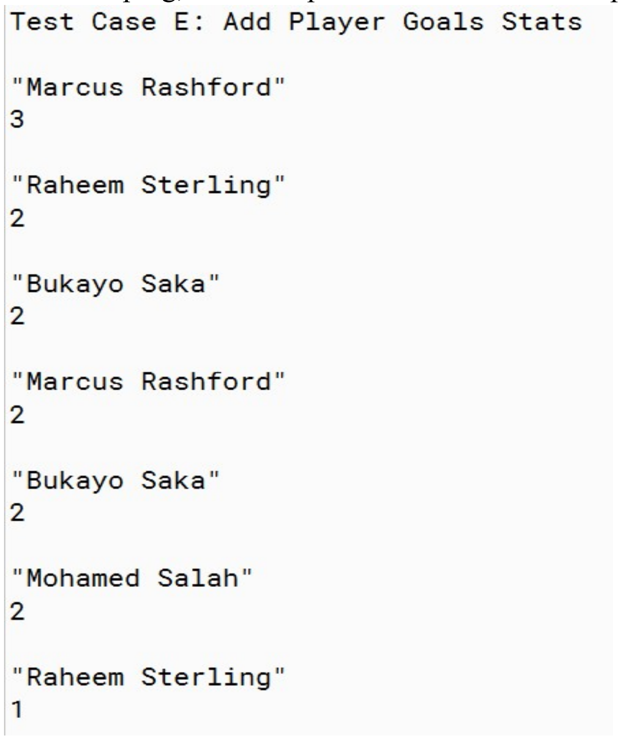
   This case adds results to the fixtures created in the latter case.
   Using the while loop, get information from the txt file.
   Assign results to the match fixture, using the add_resultFixture with the help of the match_id to identify the fixture.
   The method accesses the record_MatchStats method from the LeagueMatch class to assign values of the match.
   The get_resultDetails return the information of the results of a match using the match_id.
   This method uses get_finalScore method from the LeagueMatch Class to obtain the result



```
Test Case D: Add Fixtures Results

Manchster 3 - 0 EvertonFC
Manchster won

ChelseaFC 2 - 2 ArsenalFC
ChelseaFC and ArsenalFC drew

Liverpool 0 - 2 Manchster
Manchster won

ArsenalFC 2 - 0 EvertonFC
ArsenalFC won

Liverpool 2 - 1 ChelseaFC
Liverpool won
```

5. Test Case E

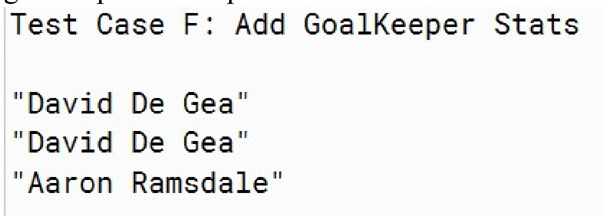
   This case tests adding the goals of a player.
   Using the while loop, add the goals with respect to the players using the add_goalPlayer method. Using the player's name, the method finds the player and adds the goals to its tally.
   While looping, there is a print statement which displays the players and its goals



```
Test Case E: Add Player Goals Stats

"Marcus Rashford"
3

"Raheem Sterling"
2

"Bukayo Saka"
2

"Marcus Rashford"
2

"Bukayo Saka"
2

"Mohamed Salah"
2

"Raheem Sterling"
1
```

6. Test Case F

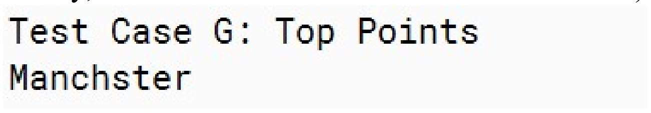
   Similar to test case E, we are adding the clean sheets of a match to the corresponding goalkeeper in that particular match fixture.



```
Test Case F: Add GoalKeeper Stats

"David De Gea"
"David De Gea"
"Aaron Ramsdale"
```

7. Test Case G

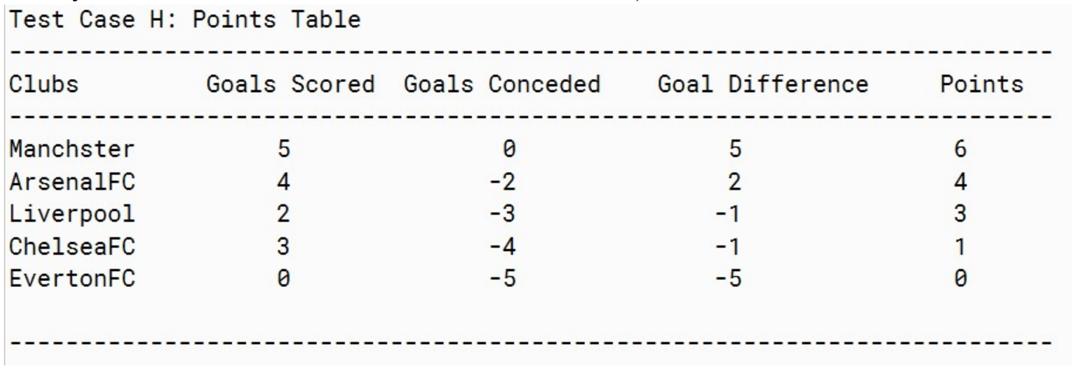
   This test case extracts the club's name having the highest points in the league.
   The get_topTeam_league method returns the highest points by a club, by sorting the club points in a descending order. Then taking the highest point element and searching through the clubs to find the club with the exact points.
   (Exception: This method only works when the points scored by each club is unique. Ideally, this is not a method used in the real scenario)



```
Test Case G: Top Points
Manchster
```

8. Test Case H

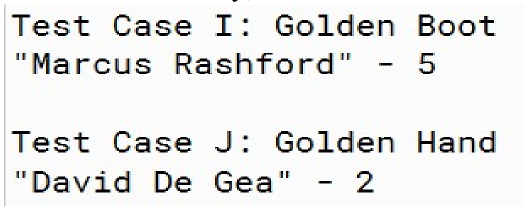
   The test case H, displays the points table of the league. Similar to the latter case, the points table is sorted in descending order and the club details are accessed with the unique points of the club.
   The get_leagueTable method does the above operation, whilst it displays the final league points table.
   (Exception: This method only works when the points scored by each club is unique. Ideally, this is not a method used in the real scenario)



```
Test Case H: Points Table
---------------------------------------------------------------------------
Clubs          Goals Scored  Goals Conceded    Goal Difference     Points
---------------------------------------------------------------------------
Manchster            5              0                5               6
ArsenalFC            4             -2                2               4
Liverpool            2             -3               -1               3
ChelseaFC            3             -4               -1               1
EvertonFC            0             -5               -5               0

---------------------------------------------------------------------------
```

9. Test Case I and Test Case J

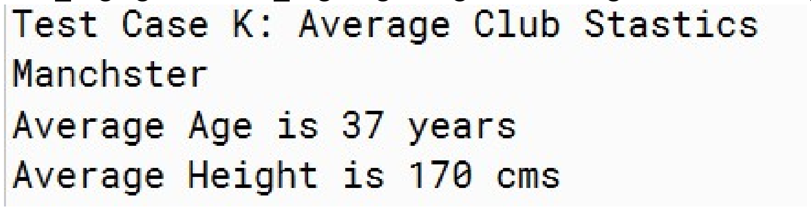
   These two cases are almost similar in operation but extracts two different outcomes. The get_goldenBoot method of the League Class accesses the club squad of all the clubs to evaluates the highest number of goals scored by a player. Upon sorting, the element is then searched to find the player's name with the corresponding goal. Similarly, goal keeper with highest clean sheet is displayed
   (Exception: This method only works when the player has unique number of goals and clean sheets. Ideally, this is not a method used in the real scenario)



```
Test Case I: Golden Boot
"Marcus Rashford" - 5

Test Case J: Golden Hand
"David De Gea" - 2
```

10. Test Case K

    This test case displays the average height and age of the player
    The get_avgClubSquad_stat method takes the club's name as the input to get the average data. The method calls upon the Club object consisting of the method club_avgAge and club_avgHeight to get the average of all ClubPlayer object



```
Test Case K: Average Club Stastics
Manchster
Average Age is 37 years
Average Height is 170 cms
```