Informatics Institute of Technology

Department of Computing

(B.Eng.) in Software Engineering

Module: Object Oriented Programming

5COSC001W

Object Oriented Programming Coursework (Semester 1)

Task is to create an application using java which simulates the manipulation of a premier league championship.

Date of Submission: 04/01/2021

Module Leader – Mr. Guhanathan Poravi

Name : Oshadha Malith Goonathilake

UoW ID - w1762649

Student ID - 2018402

Group - E

Introduction

The task is to create a java-based program that simulates the manipulation of a premier league championship. Design and implement a PremierLeagueManager(for football) class that extends the LeagueManager interface. The LeagueManager interface must be built so that it can be expanded in the future to maintain not only a range of premier football league clubs but also academic clubs such as university sports clubs and school sports clubs.

Table of Contents

UML Diagrams

Class Diagram

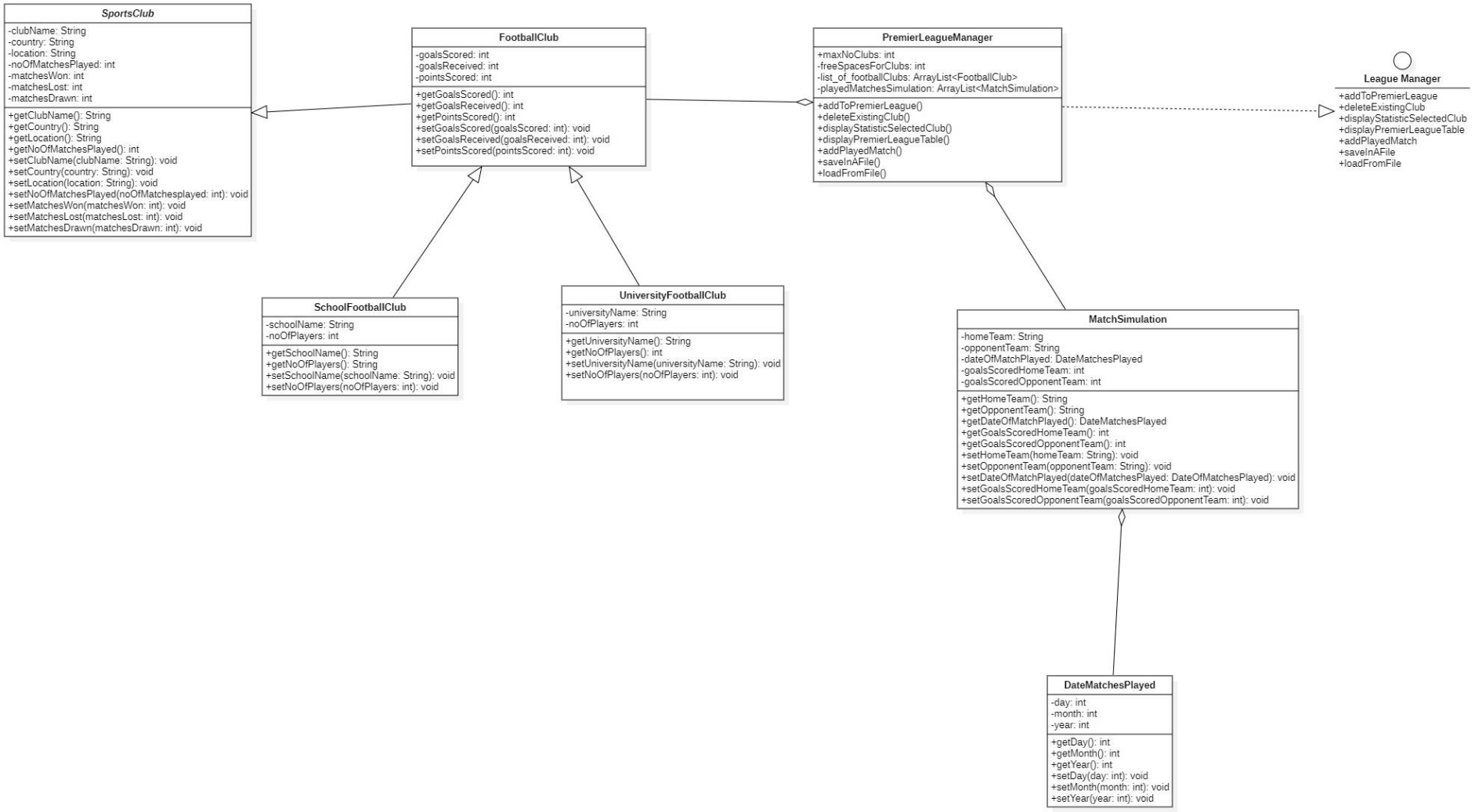


Figure 1: Class Diagram

Use Case Diagram CLI

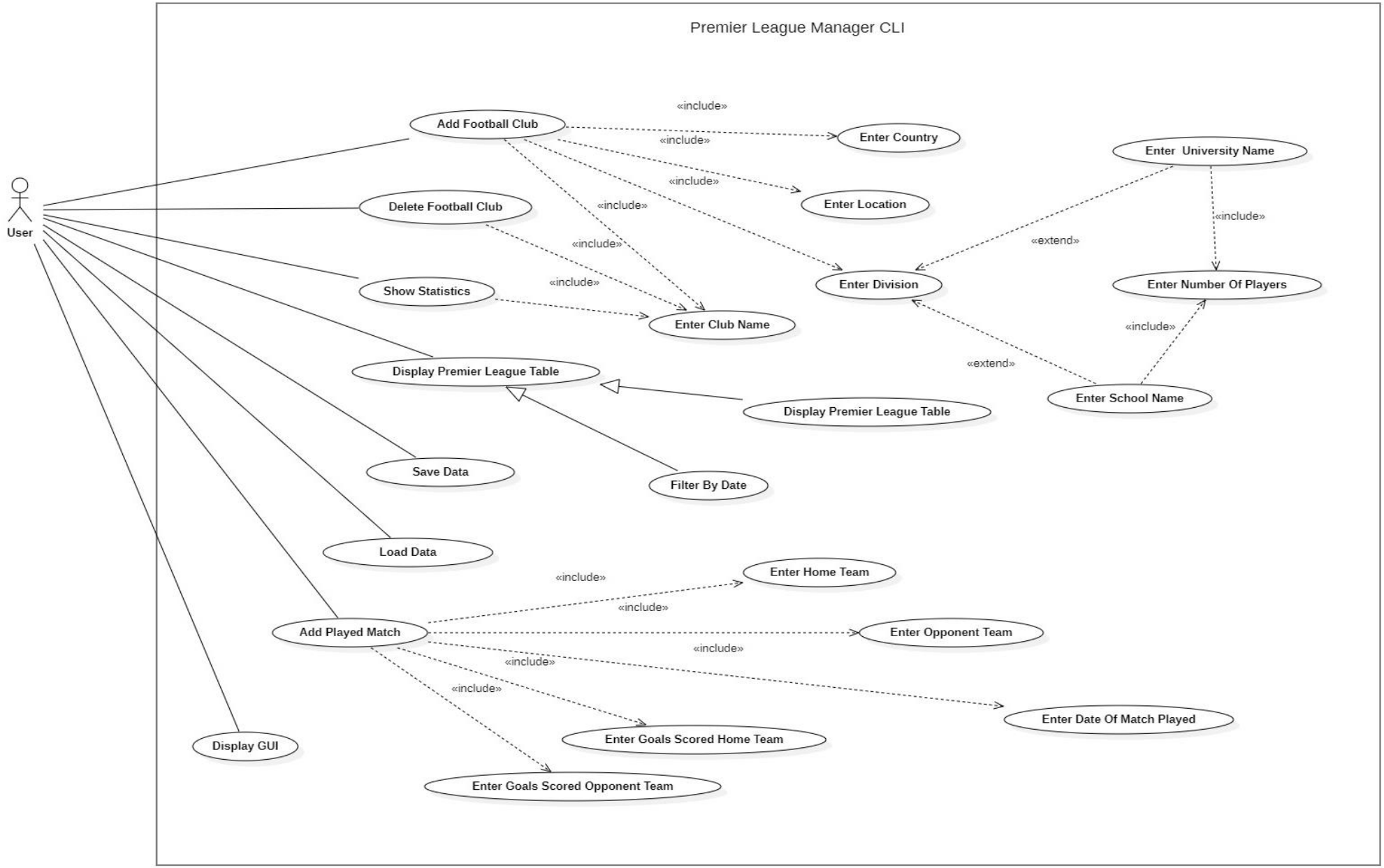


Figure 2: Use Case Diagram CLI

Use Case Diagram GUI

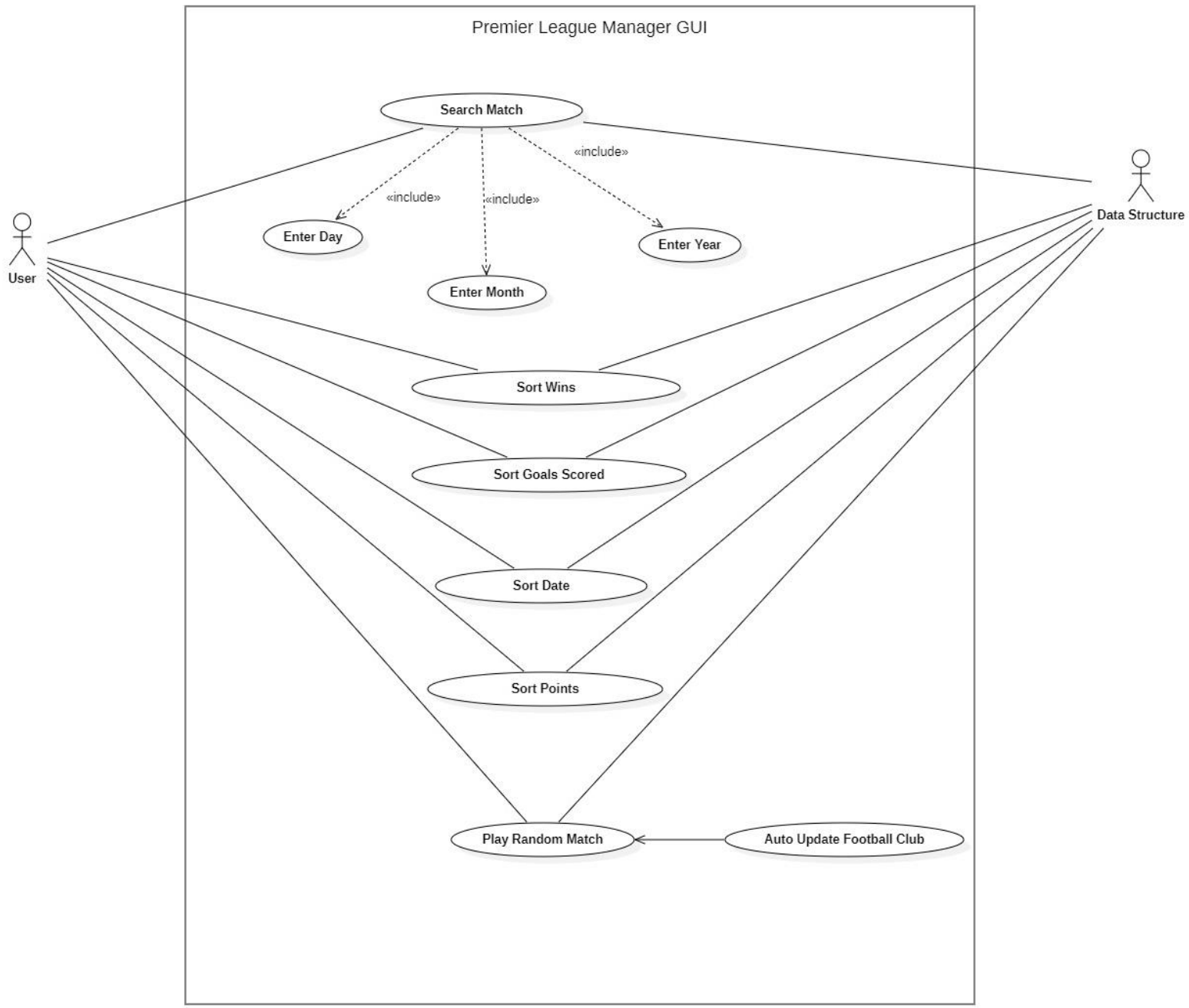


Figure 3 : Use Case Diagram GUI

Java Code for the Premier League Championship

Premier League championship backend

SportsClub

```
package controllers;

import java.io.Serializable;

public abstract class SportsClub implements Serializable {
    //attributes that a common sport club should contain
    private String clubName;
    private String country;//country of the sport club
    private String location;//location/city of the sport club
    private int noOfMatchesPlayed;
    private int matchesWon;
    private int matchesLost;
    private int matchesDrawn;

    //constructor for sports club
    public SportsClub(String clubName, String country, String location, int noOfMatchesPlayed,
                      int matchesWon, int matchesLost, int matchesDrawn) {
        this.clubName = clubName;
        this.country = country;
        this.location = location;
        this.noOfMatchesPlayed = noOfMatchesPlayed;
        this.matchesWon = matchesWon;
        this.matchesLost = matchesLost;
        this.matchesDrawn = matchesDrawn;
```

```
}

//getter method to get and displaying the name of the club
public String getClubName() {
   return clubName;
}

//setter method to set the name of the club entered by the user
public void setClubName(String clubName) {
   this.clubName = clubName;
}

//getter method to get and displaying the country of the club situated
public String getCountry() {
   return country;
}

//setter method to set the country of the club entered by the user
public void setCountry(String country) {
   this.country = country;
}

//getter method to get and displaying the location(city) of the club situated
public String getLocation() {
   return location;
}

//setter method to set the country of the location(city) entered by the user
```

```
public void setLocation(String location) {
    this.location = location;
}

//getter method to get and displaying the number of matches played
public int getNoOfMatchesPlayed() {
    return noOfMatchesPlayed;
}

//setter method to set the number of matches played
public void setNoOfMatchesPlayed(int noOfMatchesPlayed) {
    this.noOfMatchesPlayed = noOfMatchesPlayed;
}

//getter method to get and displaying the number of matches won
public int getMatchesWon() {
    return matchesWon;
}

//setter method to set the number of matches won which is entered by the user
public void setMatchesWon(int matchesWon) {
    this.matchesWon = matchesWon;
}

//getter method to get and displaying the number of matches lost
public int getMatchesLost() {
    return matchesLost;    }
```

```
    //setter method to set the number of matches lost which is entered by the user
    public void setMatchesLost(int matchesLost) {
        this.matchesLost = matchesLost;
    }

    //getter method to get and displaying the number of matches drawn
    public int getMatchesDrawn() {
        return matchesDrawn;
    }

    //setter method to set the number of matches drawn which is entered by the user
    public void setMatchesDrawn(int matchesDrawn) {
        this.matchesDrawn = matchesDrawn;
    }


    @Override
    public String toString() {
            return "SportsClub{" +"clubName=" + this.clubName + ", country=" + this.country + ",
            location=" + this.location +", noOfMatchesPlayed=" + this.noOfMatchesPlayed + ",
            matchesWon=" + this.matchesWon +", matchesLost=" + this.matchesLost +",
            matchesDrawn=" + this.matchesDrawn ;
    }
}
```

FootballClub

```
package controllers;

import java.io.Serializable;

public class FootballClub extends SportsClub implements
Comparable<FootballClub>,Serializable {

   //attributes that should contain in a football club

      private int goalsScored;

      private int goalsReceived;

      private int pointsScored;


      //constructor for football club

         public FootballClub(String clubName, String country, String location, int
         noOfMatchesPlayed, int matchesWon, int matchesLost, int matchesDrawn, int
         goalsScored, int goalsReceived, int pointsScored) {

            super(clubName, country, location, noOfMatchesPlayed, matchesWon, matchesLost,
            matchesDrawn);

            this.goalsScored = goalsScored;

            this.goalsReceived = goalsReceived;

             this.pointsScored = pointsScored;

   }


   //getter for get and display the goals scored by a football club

   public int getGoalsScored() {

      return goalsScored;

   }


   //setter method to set the goals scored by a football club entered by the user

   public void setGoalsScored(int goalsScored) {

      this.goalsScored = goalsScored;

   }
```

```
//getter for get and display the goals received for a football club
public int getGoalsReceived() {
    return goalsReceived;
}

//setter method to set the goals revived for a football club entered by the user
public void setGoalsReceived(int goalsReceived) {
    this.goalsReceived = goalsReceived;
}

//getter for get and display the points scored by a football club
public int getPointsScored() {
    return pointsScored;
}

//setter method to set the points scored by a football club entered by the user
public void setPointsScored(int pointsScored) {
    this.pointsScored = pointsScored;
}

//compare the points scored by a football club
@Override
public int compareTo(FootballClub footballClub) {
    if (this.pointsScored==footballClub.getPointsScored()){
        return this.goalsScored-footballClub.getGoalsReceived();
    }
    return this.getPointsScored()-footballClub.getPointsScored();
```

```
    }



    @Override
    public String toString() {
        return super.toString()+", goalsScored=" + this.goalsScored +", goalsReceived=" +
        this.goalsReceived + ", pointsScored=" + this.pointsScored ;
    }



}
```

LeagueManager

```
package controllers;

import java.io.IOException;

//Interface class with methods that have empty bodies

public interface LeagueManager {

    //methods that should be implemented in PremierLeagueManager class

    void addToPremierLeague(FootballClub footballClub);

    void deleteExistingClub(String clubName);

    void displayStatisticSelectedClub(String clubNameDisplay);

    void displayPremierLeagueTable();

    void addPlayedMatch(String homeTeamPlaying, String opponentTeamPlaying,
DateMatchesPlayed dateMatchPlaying, int goalsScoredHomeTeam, int
goalsScoredOpponentTeam);

    void saveInAFile() ;

    void loadFromFile() throws IOException;

}
```

PremierLeagueManager

```
package controllers;

import java.io.*;

import java.util.*;


//PremierLeagueManager class which is going to implement the classes which are in the interface LeagueManager

public class PremierLeagueManager implements LeagueManager, Serializable {

    public static final int maxNoClubs = 20;//variable showing maximum number of clubs that can play in the premier league

    private int freeSpacesForClubs = 20;//variable showing the free spaces available ing the list of football clubs.

    private List<FootballClub> list_of_footballClubs = new ArrayList<>();//arraylist which contain all the objects in sports club including football clubs

    private List<MatchSimulation> playedMatchesSimulation = new ArrayList<>();//arraylist which contain all the objects in match simulation class


    //Method that is used to create a new football club and add to the premier league manager

    @Override

    public void addToPremierLeague(FootballClub footballClub) {

        for (FootballClub footballClubNew : list_of_footballClubs) {//looping inside the list of football clubs

            if ((footballClub.getClubName().equals(footballClubNew.getClubName()))) {//if the user enters an already entered club, printing an error

                System.out.println("ERROR ! This Football club is already registered");

                System.out.println("\n");

                return;//printing the error message and return to the main menu
```

```
    }
}

//finding that the university entered by the user is already registered
if (footballClub instanceof UniversityFootballClub) {
    for (FootballClub footballClubNew : list_of_footballClubs) {
        if (footballClubNew instanceof UniversityFootballClub) {
            if ((((UniversityFootballClub)
            footballClubNew).getUniversityName()).equals((((UniversityFootballClub)
            footballClub).getUniversityName()))) {
            System.out.println("ERROR ! This UNIVERSITY IS ALREADY
            REGISTERED");
            System.out.println("\n");
            return;
          }
        }
    }
}



//finding that the school entered by the user is already registered
if (footballClub instanceof SchoolFootballClub) {
    for (FootballClub footballClubNew : list_of_footballClubs) {
        if (footballClubNew instanceof SchoolFootballClub) {
            if ((((SchoolFootballClub)
            footballClubNew).getSchoolName()).equals((((SchoolFootballClub)
            footballClub).getSchoolName()))) {
```

```
                    System.out.println("ERROR ! This SCHOOL IS ALREADY
                    REGISTERED");

                System.out.println("\n");

                return;

            }

        }

    }

}


if (freeSpacesForClubs == 0) {

    System.out.println("ERROR ! The Football club is Full");//if the spaces in the football
    club drops to zero printing an error message

} else {

    list_of_footballClubs.add(footballClub);//if there are no error adding football clubs to the
    arraylist

    freeSpacesForClubs -= footballClub instanceof UniversityFootballClub ? 1 : 1;//if the
    football club is a university football club reducing the space by one and else also one

    System.out.println("YOU HAVE SUCCESSFULLY ADDED A FOOTBALL
    CLUB...CHEERS !");

    //printing the number of free slots remaining

    System.out.println(freeSpacesForClubs > 0 ? ("Free Slots Remaining to add football
    clubs: " + freeSpacesForClubs) : "No More Spaces available to add a football club");

    System.out.println(list_of_footballClubs);

}

System.out.println("\n");


if (freeSpacesForClubs >= maxNoClubs) {//if the free slots became greater than or equal to
the maximum number of clubs printing an error message
```

```
        System.out.println("ERROR ! No spaces available to add any football club");
        System.out.println("\n");
    }

}

//Method that is used to delete an existing football club from premier league
@Override
public void deleteExistingClub(String clubName) {
        if (list_of_footballClubs.isEmpty()) {//printing an error message if the football club list is empty..so can't perform delete operation
        System.out.println("No Football clubs in the list,yet!");
    } else {
        boolean foundClub = false;//boolean value to find the club name
        for (FootballClub footballClub : list_of_footballClubs) {
        if (footballClub.getClubName().equals(clubName)) {//if the club name is inside the arraylist
                foundClub = true;//making the boolean value to true
        list_of_footballClubs.remove(footballClub);//removing the relevant club from the list of football clubs
                System.out.println("SAD NEWS !!!");
                System.out.printf("A %s has Left the Football Club List.%n", footballClub instanceof UniversityFootballClub ? "University Football Club" : "School Football Club");
                freeSpacesForClubs += footballClub instanceof UniversityFootballClub ? 1 : 1;//updating the free slots in the list of football clubs
                System.out.println("\n");
```

```
                    System.out.printf("Free Slots Remaining: %d%n", freeSpacesForClubs);
                //printing the remaining spaces

                System.out.println("\n");

                break;

            }

        }

        if (foundClub == false) {//if the club is not found printing an error message

            System.out.println("Invalid Club Name! Please Check & Try Again!");

            System.out.println("\n");

        }

    }

}


//Method that is used to display the statistics for a selected club

@Override

public void displayStatisticSelectedClub(String clubNameDisplay) {


    if (list_of_footballClubs.isEmpty()) {//printing an error message if the football club is
    empty

        System.out.println("No Football clubs in the list,yet!");

        System.out.println("\n");

    } else {

        boolean foundClub = false;//boolean value to find the club name


        for (FootballClub footballClub : list_of_footballClubs) {
```

```
        if (footballClub.getClubName().equals(clubNameDisplay)) {//if the club is in the list of
        football clubs

            foundClub = true;//making the boolean value tto true

            System.out.println("********************************** STATISTICS
**********************************");

            System.out.println("\n");

            if (footballClub instanceof UniversityFootballClub) {//if the football club is a
university football club printing the name of the university

                System.out.println("* Name of the University [U23 Division] \t: " +
((UniversityFootballClub) footballClub).getUniversityName());

            } else {

                //if the football club is a school football club printing the name of the school

                  System.out.println("* Name of the School [U18 Division]\t: " +
                ((SchoolFootballClub) footballClub).getSchoolName());

            }

            System.out.println("* Name of the Club \t\t\t: " +
footballClub.getClubName());//displaying the name of the club

            System.out.println("* Country of the Club \t\t\t: " +
footballClub.getCountry());//displaying the country of the club

            System.out.println("* Location of the Club \t\t\t: " +
footballClub.getLocation());//displaying the city of the club

            System.out.println("* Number Of Matches Played \t\t: " +
footballClub.getNoOfMatchesPlayed());//displaying the number of matches played by the club

            System.out.println("* Number of Matches Won \t\t: " +
footballClub.getMatchesWon());//displaying the number of matches won by the club

            System.out.println("* Number of Matches Lost \t\t: " +
footballClub.getMatchesLost());//displaying the number of matches lost by the club

            System.out.println("* Number of Matches Drawn \t\t: " +
footballClub.getMatchesDrawn());//displaying the number of matches drawn by the club
```

```
                System.out.println("* Goals Scored \t\t\t\t: " +
footballClub.getGoalsScored());//displaying the number of goals scored by the club

                System.out.println("* Goals Received \t\t\t: " +
footballClub.getGoalsReceived());//displaying the number of goals received by the club

                System.out.println("* Points Scored \t\t\t: " +
footballClub.getPointsScored());//displaying the points scored by the club

                System.out.println("\n");


System.out.println("**********************************************************************
*****************");

                System.out.println("\n");

                break;

            }

        }

        if (foundClub == false) {//if the football club is not found printing an error message

            System.out.println("Invalid Club Name! Please Check & Try Again!");

            System.out.println("\n");

        }

    }

}



  //Method that is used to display the premier league table in descending order of their points or
goal difference

  @Override

  public void displayPremierLeagueTable() {

    Scanner user_input = new Scanner(System.in);
```

```
System.out.println("1 => Display Premier League Table");

System.out.println("2 => Filter Matches played to a particular date");

System.out.println("\n");

System.out.println("Select [1 or 2] from above to proceed... : ");

int choice = user_input.nextInt();

System.out.println("\n");


if (choice == 1) {//if the user wants to show the premier league table

    Collections.sort(list_of_footballClubs, Collections.reverseOrder());//sort the arraylist of
    football clubs in descending order


    System.out.println("----------------------------------------------------------------------------------------
----------" +

            "----------------------------------------------------------------------------------------------------
----------");

    //headings of the table

     System.out.printf("| %-22s | %-15s | %-15s | %-15s | %-15s | %-15s | %-15s | %-15s | %-
    15s | %-15s | %-14s |", "ClubName", "Country", "Location",

    "NoOfMatchesPlayed", "MatchesWon", "MatchesLost", "MatchesDrawn",
    "GoalsScored", "GoalsReceived", "GoalsDifference", "PointsScored");

    System.out.println("\n");

    System.out.println("----------------------------------------------------------------------------------------
----------" +

            "----------------------------------------------------------------------------------------------------
----------");


    System.out.println("\n");
```

```
        for (FootballClub footballClub : list_of_footballClubs) {

            //values that are coming in the table

         System.out.printf("| %-22s | %-15s | %-15s | %-17s | %-15s | %-15s | %-15s | %-15s | %-15s |%-15s | %-14s |", footballClub.getClubName(), footballClub.getCountry(), footballClub.getLocation(), footballClub.getNoOfMatchesPlayed(), footballClub.getMatchesWon(), footballClub.getMatchesLost(),

        footballClub.getMatchesDrawn(), footballClub.getGoalsScored(), footballClub.getGoalsReceived(), (footballClub.getGoalsScored() - footballClub.getGoalsReceived()), footballClub.getPointsScored(), "|\n");

            System.out.println("\n");

            System.out.println("-------------------------------------------------------------------------------------------------------------------" +

                    "-------------------------------------------------------------------------------------------------------------------");

            System.out.println("\n");

        }

    } else if (choice == 2) {//if the user wants to filter the matches played by a specific date

        boolean dateFoundBoolean = false;//boolean vale to find the date of the match played


        System.out.println("Please Enter the Day of the match played: ");//taking the day of the match played

        int day = user_input.nextInt();


        System.out.println("Please Enter the Month of the match played: ");//taking the month of the match played

        int month = user_input.nextInt();
```

```
        System.out.println("Please Enter the Year of the match played : 2020");//taking the year
        of the match played
//          int year = user_input.nextInt();
        int year = 2020;
        System.out.println("\n");

        for (MatchSimulation matchSimulation : playedMatchesSimulation) {//looping inside the
        match simulation class from the arraylist
           //if the day,month and year is in the arraylist printing the statistics of the matches
           played
           if ((matchSimulation.getDateOfMatchPlayed().getDay() == day) &&
        (matchSimulation.getDateOfMatchPlayed().getMonth() == month)
                   && (matchSimulation.getDateOfMatchPlayed().getYear() == year)) {
              System.out.println("-------------------------------------------------------------------------------
-----------------------");

              System.out.printf("| %-22s | %-22s | %-15s | %-25s |", "HomeClubName",
              "OpponentClubName", "HomeClubGoalsScored", "OpponentClubGoalsScored");
              System.out.println("\n");
              System.out.println("-------------------------------------------------------------------------------
-----------------------");
              System.out.println("\n");

              //displaying the values of the table which is sorted to a specific date
             System.out.printf("| %-22s | %-22s | %-19s | %-25s | ",
        matchSimulation.getHomeTeam(),

        matchSimulation.getOpponentTeam(),matchSimulation.getGoalsScoredHomeTeam(),
        matchSimulation.getGoalsScoredOpponentTeam(), "|\n");
```

```
                System.out.println("\n");
                System.out.println("--------------------------------------------------------------------------------
------------------------");
                System.out.println("\n");
                dateFoundBoolean = true;//making the boolean value to true as the date is correct
            }


        }
        if (dateFoundBoolean == false) {//if the date is not found printing and error message
            System.out.println("ERROR ! Invalid Date or You have entered a wrong date...");
            System.out.println("\n");
        }
    } else {
        //if the user inputs anything else 1 and 2 options printing an error message
        System.out.println("ERROR ! Wrong Input...Try Again...");
        System.out.println("\n");
    }


}


//Method that is used to add a played match with its score and its date
@Override
public void addPlayedMatch(String homeTeamPlaying, String opponentTeamPlaying,
DateMatchesPlayed dateMatchesPlayed,
                    int goalsScoredHomeTeam, int goalsScoredOpponentTeam) {
```

```
//check that home team and the opponent team is equal
if (homeTeamPlaying.equals(opponentTeamPlaying)) {
   System.out.println("ERROR ! Home Team and Opponent Team cannot be the same");
   System.out.println("\n");
}

boolean homeClubFound = false;//to find the home club entered by the user
boolean opponentClubFound = false;//to find the opponent club entered by the user

boolean isClubUniversity = false;//to find the club entered by the user belongs to which division

FootballClub homeClub = null;//taking a variable to set the relevant attributes related to that particular football club(home club)

for (FootballClub footballClub : list_of_footballClubs) {
    if (footballClub.getClubName().equals(homeTeamPlaying)) {//if the home club entered by the user is in the football club arraylist
      if (footballClub instanceof UniversityFootballClub) {//and of the home club os a university football club
         isClubUniversity = true;//making the boolean value to true as the home club is a university football club.

      }

      //else if the football club entered by the user is a school football club
```

```
        homeClub = footballClub;//take the specific club name entered by the user and the
    relevant features of that club name into the home club variable


        homeClubFound = true;//as the home club is found making the boolean value to true


    }
}
    FootballClub opponentClub = null;//taking a variable to set the relevant attributes related
    to that particular football club(opponent club)


for (FootballClub footballClub : list_of_footballClubs) {
     if ((footballClub.getClubName().equals(opponentTeamPlaying))) {//if the opponent club
    entered by the user is in the list of football clubs
        if (isClubUniversity == true) {//making the boolean value to true as it a university
        football club
            if (footballClub instanceof UniversityFootballClub) {


                isClubUniversity = true;
            }
        }


        opponentClub = footballClub;
        opponentClubFound = true;
    }


}
```

```
if (homeClubFound == false) {//if the home club entered by the user is not found printing an error message

    System.out.println("ERROR ! This Home Team is not registered...Please register it first !!!");

    System.out.println("\n");


}

     if (opponentClubFound == false) {//if the opponent club entered by the user is not found printing an error message

    System.out.println("ERROR ! This Opponent Team is not registered on selected divisions...Please register it first !!!");

    System.out.println("\n");


}

     if (homeClubFound == true && opponentClubFound == true) {//if the home club and the opponent club entered by the user, both are found adding the elements to the arraylist and setting it to the match simulation class

    MatchSimulation matchSimulation = new MatchSimulation(homeTeamPlaying, opponentTeamPlaying, dateMatchesPlayed, goalsScoredHomeTeam, goalsScoredOpponentTeam);

    playedMatchesSimulation.add(matchSimulation);

    System.out.println(playedMatchesSimulation);



    //printing an error message because one club can play maximum of 38 matches only for the season.

    if (homeClub.getNoOfMatchesPlayed() == 38) {
```

```
 System.out.println("ERROR ! MAXIMUM AMOUNT OF MATCHES PLAYED BY A single CLUB SHOULD NOT EXCEED 38 [Home club has exceeded the maximum amount]");

    System.out.println("\n");

}

if (opponentClub.getNoOfMatchesPlayed() == 38) {

System.out.println("ERROR ! MAXIMUM AMOUNT OF MATCHES PLAYED BY A single CLUB SHOULD NOT EXCEED 38 [Opponent club has exceeded the maximum amount]");

    System.out.println("\n");

}



homeClub.setNoOfMatchesPlayed(homeClub.getNoOfMatchesPlayed() + 1);//increase the number of matches played by one

homeClub.setGoalsScored(homeClub.getGoalsScored() + goalsScoredHomeTeam);//updating the goals scored the home team

homeClub.setGoalsReceived(homeClub.getGoalsReceived() + goalsScoredOpponentTeam);//updating the goals received by the home team



opponentClub.setNoOfMatchesPlayed(opponentClub.getNoOfMatchesPlayed() + 1);//increase the number of matches played by one

opponentClub.setGoalsScored(opponentClub.getGoalsScored() + goalsScoredOpponentTeam);//updating the goals scored the opponent team

opponentClub.setGoalsReceived(opponentClub.getGoalsReceived() + goalsScoredHomeTeam);//updating the goals received by the opponent team
```

```
if (goalsScoredHomeTeam > goalsScoredOpponentTeam) {//if the goals scored by home team is greater than the goals scored by the opponent team

    homeClub.setPointsScored(homeClub.getPointsScored() + 3);//increasing the points of the home team by 3

    homeClub.setMatchesWon(homeClub.getMatchesWon() + 1);//increasing the number of matches won by the home team by one

    opponentClub.setMatchesLost(opponentClub.getMatchesLost() + 1);//increasing the number of matches lost by the opponent team by one

    System.out.println("HOME CLUB HAS WON THE MATCH...");

    System.out.println("\n");


}

if (goalsScoredHomeTeam < goalsScoredOpponentTeam) {//if the goals scored by opponent team is greater than the goals scored by the home team

    opponentClub.setPointsScored(opponentClub.getPointsScored() + 3);//increasing the points of the opponent team by 3

    opponentClub.setMatchesWon(opponentClub.getMatchesWon() + 1);//increasing the number of matches won by the opponent team by one

    homeClub.setMatchesLost(homeClub.getMatchesLost() + 1);//increasing the number of matches lost by the home team by one

    System.out.println("OPPONENT CLUB HAS WON THE MATCH...");

    System.out.println("\n");


}

if (goalsScoredHomeTeam == goalsScoredOpponentTeam) {//if the goals scored by the home team and the opponent team is equal

    homeClub.setPointsScored(homeClub.getPointsScored() + 1);//increasing the number of points scored by the home club by one
```

```
            opponentClub.setPointsScored(opponentClub.getPointsScored() + 1);//increasing the
            number og points scored by the opponent club by one

            homeClub.setMatchesDrawn(homeClub.getMatchesDrawn() + 1);//increasing the
            number of matches drawn by the home club by one

            opponentClub.setMatchesDrawn(opponentClub.getMatchesDrawn() + 1);//increasing
            the number of matches drawn by the opponent club by one

            System.out.println("MATCH HAS BEEN DRAWN...");

            System.out.println("\n");

        }

    }

}



//Method that is used to save the the information entered by the user into a text file

@Override

public void saveInAFile() {


    try {

        //creating text file of football clubs

        FileOutputStream fileOutputStreamPremierLeague1 = new
        FileOutputStream("footballClubPremierLeague.txt");

        ObjectOutputStream objectOutputStreamPremierLeague1 = new
        ObjectOutputStream(fileOutputStreamPremierLeague1);



        //creating text file of matches played

        FileOutputStream fileOutputStreamPremierLeague2 = new
        FileOutputStream("matchSimulation.txt");
```

```
ObjectOutputStream objectOutputStreamPremierLeague2 = new
ObjectOutputStream(fileOutputStreamPremierLeague2);


//writing objects into the text file which are in the football clubs
for (FootballClub footballClub : list_of_footballClubs) {
   objectOutputStreamPremierLeague1.writeObject(footballClub);
}
//flush the object output stream
objectOutputStreamPremierLeague1.flush();
//close the fileoutputstream and objectoutputstream
fileOutputStreamPremierLeague1.close();
objectOutputStreamPremierLeague1.close();


//writing objects into the text file which the matches are played
for (MatchSimulation matchSimulation : playedMatchesSimulation) {
   objectOutputStreamPremierLeague2.writeObject(matchSimulation);
}
//flush the object output stream
objectOutputStreamPremierLeague2.flush();
//close the fileoutputstream and objectoutputstream
fileOutputStreamPremierLeague2.close();
objectOutputStreamPremierLeague2.close();


System.out.println("DATA SAVED SUCCESSFULLY...");
System.out.println("\n");
```

```
        //show any errors there are errors
    } catch (Exception exception) {
        System.out.println("ERROR in Saving !");
        System.out.println("\n");
    }
}


@Override
public void loadFromFile() throws IOException {

    try {
        //Creating a stream to read the objects in the text file
        FileInputStream fileInputStream1 = new
        FileInputStream("footballClubPremierLeague.txt");
        ObjectInputStream objectInputStream1 = new ObjectInputStream(fileInputStream1);

        while (true) {
            FootballClub footballClub = (FootballClub) objectInputStream1.readObject();

            list_of_footballClubs.add(footballClub);
          freeSpacesForClubs -= footballClub instanceof UniversityFootballClub ? 1 : 1;//if the
        football club is a university football club reducing the space by one and else also one

        }
} catch (ClassNotFoundException classNotFoundException) {//exception for class not found
```

```
    System.out.println("ERROR ! Class not found Exception has occurred");
    System.out.println("\n");
} catch (FileNotFoundException fileNotFoundException) {
    System.out.println("ERROR ! File not found Exception has occurred");
    System.out.println("\n");
} catch (EOFException eofException) {//exception for end of file
    System.out.println("==============================");
    System.out.println("FILE HAS BEEN READ COMPLETELY");
    System.out.println("==============================");
    System.out.println("\n");
}
if (list_of_footballClubs.size() > 1) {
    System.out.println("DATA LOADED SUCCESSFULLY OF FOOTBALL CLUBS");
    System.out.println("\n");
}
     new FileOutputStream("footballClubPremierLeague.txt").close();//flushing the text file
    after reading


try {
    //Creating a stream to read the objects in the text file
    FileInputStream fileInputStream2 = new FileInputStream("matchSimulation.txt");
    ObjectInputStream objectInputStream2 = new ObjectInputStream(fileInputStream2);


    while (true) {
            MatchSimulation matchSimulation = (MatchSimulation)
           objectInputStream2.readObject();
```

```
                playedMatchesSimulation.add(matchSimulation);
            }

        } catch (ClassNotFoundException classNotFoundException) {//exception for class not found
            System.out.println("ERROR ! Class not found Exception has occurred");
            System.out.println("\n");
        } catch (FileNotFoundException fileNotFoundException) {
            System.out.println("ERROR ! File not found Exception has occurred");
            System.out.println("\n");
        } catch (EOFException eofException) {//exception for end of file
            System.out.println("==============================");
            System.out.println("FILE HAS BEEN READ COMPLETELY");
            System.out.println("==============================");
            System.out.println("\n");
        }
        if (playedMatchesSimulation.size() > 1) {
            System.out.println("DATA LOADED SUCCESSFULLY OF MATCHES PLAYED");
            System.out.println("\n");
        }
        new FileOutputStream("matchSimulation.txt").close();//flushing the text file after reading
    }

}
```

SchoolFootballClub

```
package controllers;

import java.io.Serializable;

public class SchoolFootballClub extends FootballClub implements Serializable {
    //attributes that a school football club should contain
    private String schoolName;
    private int noOfPlayers;//number of players in the school football club

    //constructor for school football club
    public SchoolFootballClub( String schoolName, int noOfPlayers,String clubName,
                               String country, String location,int noOfMatchesPlayed,
                               int matchesWon, int matchesLost,int matchesDrawn,
                               int goalsScored, int goalsReceived,
                              int pointsScored) {
        super(clubName, country, location, noOfMatchesPlayed, matchesWon, matchesLost,
        matchesDrawn, goalsScored, goalsReceived, pointsScored);
         this.schoolName = schoolName;
        this.noOfPlayers = noOfPlayers;
    }

    //getter method to get and display the school name
    public String getSchoolName() {
       return schoolName;
    }

    //setter method to set the school name entered by the user
    public void setSchoolName(String schoolName) {
```

```
        this.schoolName = schoolName;
    }

    //getter method to get and display the number of player
    public int getNoOfPlayers() {
        return noOfPlayers;
    }

    //setter method to set the number of players entered by the user
    public void setNoOfPlayers(int noOfPlayers) {
        this.noOfPlayers = noOfPlayers;
    }

    @Override
    public String toString() {
            return super.toString() +"schoolName=" + this.schoolName +", noOfPlayers=" +
           this.noOfPlayers +"}";
    }
}
```

UniversityFootballClub

```
package controllers;

import java.io.Serializable;

public class UniversityFootballClub extends FootballClub implements Serializable {
    //attributes that a university football club should contain
    private String universityName;
    private int noOfPlayers;//number of players in the university football club

    //constructor for university football club
    public UniversityFootballClub(String universityName, int noOfPlayers,String clubName,
                                  String country, String location,int noOfMatchesPlayed,
                                   int matchesWon, int matchesLost,int matchesDrawn,
                                  int goalsScored, int goalsReceived,
                                  int pointsScored) {
        super(clubName, country, location, noOfMatchesPlayed, matchesWon, matchesLost,
        matchesDrawn,goalsScored, goalsReceived, pointsScored);
        this.universityName = universityName;
        this.noOfPlayers = noOfPlayers;
    }

    //getter method to get and display the university name
    public String getUniversityName() {
        return universityName;
    }

    //setter method to set the number of players which is entered by the user
    public void setUniversityName(String universityName) {
```

```
        this.universityName = universityName;
    }

    //getter method to get and display the number of players
    public int getNoOfPlayers() {
        return noOfPlayers;
    }

    //setter method to set the number of players which is entered by the user
    public void setNoOfPlayers(int noOfPlayers) {
        this.noOfPlayers = noOfPlayers;
    }

    @Override
    public String toString() {
            return super.toString() +"universityName='" + this.universityName  +", noOfPlayers=" +
            this.noOfPlayers +"}";
    }
}
```

MatchSimulation

```
package controllers;


import java.io.Serializable;


public class MatchSimulation implements Comparable<MatchSimulation>,Serializable {
    private String homeTeam;
    private String opponentTeam;
    private DateMatchesPlayed dateOfMatchPlayed;
    private int goalsScoredHomeTeam;
    private int goalsScoredOpponentTeam;



    public MatchSimulation(String homeTeam, String opponentTeam, DateMatchesPlayed
    dateOfMatchPlayed,int goalsScoredHomeTeam, int goalsScoredOpponentTeam) {
        this.homeTeam = homeTeam;
        this.opponentTeam = opponentTeam;
        this.dateOfMatchPlayed = dateOfMatchPlayed;
        this.goalsScoredHomeTeam = goalsScoredHomeTeam;
        this.goalsScoredOpponentTeam = goalsScoredOpponentTeam;
    }


    public String getHomeTeam() {
        return homeTeam;
    }


    public void setHomeTeam(String homeTeam) {
        this.homeTeam = homeTeam;
    }
```

```
public String getOpponentTeam() {
    return opponentTeam;
}

public void setOpponentTeam(String opponentTeam) {
    this.opponentTeam = opponentTeam;
}

public DateMatchesPlayed getDateOfMatchPlayed() {
    return dateOfMatchPlayed;
}

public void setDateOfMatchPlayed(DateMatchesPlayed dateOfMatchPlayed) {
    this.dateOfMatchPlayed = dateOfMatchPlayed;
}

public int getGoalsScoredHomeTeam() {
    return goalsScoredHomeTeam;
}

public void setGoalsScoredHomeTeam(int goalsScoredHomeTeam) {
    this.goalsScoredHomeTeam = goalsScoredHomeTeam;
}

public int getGoalsScoredOpponentTeam() {
    return goalsScoredOpponentTeam;
}
```

```
    public void setGoalsScoredOpponentTeam(int goalsScoredOpponentTeam) {
        this.goalsScoredOpponentTeam = goalsScoredOpponentTeam;
    }



    @Override
    public String toString() {
        return "MatchSimulation{" +"homeTeam=" + this.homeTeam +", opponentTeam=" +
this.opponentTeam  + ", dateOfMatchPlaying=" + this.dateOfMatchPlayed +",
goalsScoredHomeTeam=" + this.goalsScoredHomeTeam + ", goalsScoredOpponentTeam=" +
this.goalsScoredOpponentTeam +'}';
    }


    @Override
    public int compareTo(MatchSimulation matchSimulation) {
        if
(this.dateOfMatchPlayed.getMonth()==matchSimulation.getDateOfMatchPlayed().getMonth()){
            return this.dateOfMatchPlayed.getDay()-
            matchSimulation.getDateOfMatchPlayed().getDay();
        }
            return this.dateOfMatchPlayed.getMonth()-
            matchSimulation.getDateOfMatchPlayed().getMonth();
    }
}
```

DateMatchesPlayed

```
package controllers;

import java.io.Serializable;

import java.util.Scanner;



public class DateMatchesPlayed implements Serializable {

    private int day;

    private int month;

    private int year;


    private static Scanner dateValidation = new Scanner(System.in);

    public DateMatchesPlayed(int day, int month, int year) {


        try{

            if(day>0 && day<=31){

                this.day = day;

            }else{

                System.out.print("Please Enter valid date which played the match: ");

                setDay(dateValidation.nextInt());

                System.out.println("\n");

            }

        }catch (Exception e){

            System.out.print("Please enter valid date which played the match: ");

            setDay(dateValidation.nextInt());

            System.out.println("\n");

        }
```

```
try{
   if(month>0 &&month<=12){
      this.month = month;
   }else{
      System.out.print("Please Enter valid month which played the match:  ");
      setMonth(dateValidation.nextInt());
      System.out.println("\n");

   }
}catch (Exception e){
   System.out.print("Please enter valid month which played the match: ");
   setMonth(dateValidation.nextInt());
   System.out.println("\n");

}

try{
   if(year==2020){
      this.year = year;
   }else{
      System.out.print("Please Enter valid year which played the match: ");
      setYear(dateValidation.nextInt());
      System.out.println("\n");

   }
}catch (Exception e){
   System.out.print("Please enter valid year which played the match: ");
```

```
        setYear(dateValidation.nextInt());
        System.out.println("\n");

    }
}

public int getDay() {
    return day;
}

public void setDay(int day) {
    this.day=day;

}

public int getMonth() {
    return month;
}

public void setMonth(int month) {
    this.month=month;

}

public int getYear() {
    return year;
}
```

```
    public void setYear(int year) {
        this.year=year;
    }

    @Override
    public String toString() {
        return "Date{" +"day=" + this.day +", month=" + this.month +", year=" + this.year +'}';
    }
}
```

ConsoleSystem

```
package controllers;

import java.io.File;

import java.io.IOException;

import java.io.Serializable;

import java.net.URISyntaxException;

import java.util.Scanner;


public class ConsoleSystem implements Serializable {

    static LeagueManager premierLeagueManager = new PremierLeagueManager();

    final static Scanner User_Input = new Scanner(System.in);//scanner for the user inputs


    public static void main(String[] args) throws IOException, URISyntaxException {


        try {

            // Sleep for 5 Seconds

            System.out.println("\n");

            System.out.println(".....");

            System.out.println("YOU ARE ENTERING TO THE PREMIER LEAGUE ");

            System.out.println("\n");

            System.out.println("server is getting ready !!! PLEASE WAIT ...");

            System.out.println(".....");

            System.out.println("\n");

            Thread.sleep(5000);

            System.out.println("**********HERE WE GO**********");

            System.out.println("\n");

        }catch(InterruptedException interruptedException) {

            System.out.println(interruptedException);
```

```
}

    //run the playframework and angular in two cmds at the start of the premier league
    championship
ProcessBuilder processBuilderPlayFrameWork=new ProcessBuilder();
processBuilderPlayFrameWork.command("cmd.exe","/c","start sbt run");
processBuilderPlayFrameWork.directory(new File("../premier-league-manager"));

ProcessBuilder processBuilderAngular=new ProcessBuilder();
processBuilderAngular.command("cmd.exe","/c","start ng serve");
processBuilderAngular.directory(new File("../premier-league-manager-frontend"));

try {
    //start the cmds to run the playframework and angular projects
    processBuilderPlayFrameWork.start();
    processBuilderAngular.start();
}catch (Exception exception){
    System.out.println(exception);
}

premierLeagueManager.loadFromFile();//load from the file

mainMenu:
while (true) {
    displayMenu();//display the menu
    System.out.println("Enter a number from above to Proceed ...");//ask the user to decide a
    choice from the menu
    int choice = User_Input.nextInt();
    System.out.println("\n");
```

```
switch (choice) {
    case 1:
        addToPremierLeague();//call the method to add a new football club
        break;
    case 2:
        deleteExistingClub();//call the method to delete an existing club
        break;
    case 3:
    displayStatisticSelectedClub();//call the method to display statistics of a particular club
        break;
    case 4:
        displayPremierLeagueTable();//call the method to display the premier league table
        break;
    case 5:
        addPlayedMatch();//call the method to add a played match
        break;
    case 6:
        saveInAFile();//call the method to save the details in text file
        break;
    case 7:
        premierLeagueGUI();//call the method to open the premier league gui
        break;
    case 8:
            System.out.println("Thank you for choosing the system, Have a pleasant
            Day");//Exit from the menu
        break mainMenu;
    default:
```

```
                System.out.println("<<<<You selected an Invalid option. Please Try Again
                !>>>>");//invalid option selected from the menu

            continue mainMenu;

        }


    }

}


private static void displayMenu() {

    //Display the menu


    System.out.println("----------------------------------------/*\\----------------------------------------
---");

    System.out.println("=================WELCOME TO THE FOOTBALL
PREMIERE LEAGUE CHAMPIONSHIP=================");

    System.out.println("\n");

    System.out.println(".....................................................");

    System.out.println("1. Add a club to the premier League Manager");

    System.out.println("2. Delete an existing club from the premier League");

    System.out.println("3. Display Statistics for a selected club");

    System.out.println("4. Display Premier League Table");

    System.out.println("5. Add a played match");

    System.out.println("6. Save Into a File");

    System.out.println("7. Open Premier League GUI");

    System.out.println("8. Exit");

    System.out.println(".....................................................");

    System.out.println("----------------------------------------\\*/----------------------------------------
--");


    System.out.println("\n");
```

```
    }


    //method to add a new club to the premier league
    private static void addToPremierLeague() {

        FootballClub footballClub;//initializing the football club


        //initializing the variables
        int totalMatchesPlayed = 0;
        int noOfMatchesWon = 0;
        int noOfMatchesLost = 0;
        int noOfMatchesDraw = 0;
        int goalsScored = 0;
        int goalsReceived = 0;
        int pointsScored = 0;


            User_Input.nextLine();//as a football club can have spaces between the name of the club,here used a nextLine(), if this nextline() is not there the name
             //of the club will not be taken, it will skip to take the country of the club.


        mainLoopAdd:
        while (true) {

System.out.println("++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++");
            System.out.println("\t\tYOU ARE GOING TO ADD FOOTBALL CLUBS TO THE PREMIER LEAGUE BASED ON YOUR UNIVERSITY AND SCHOOL");
```

```
System.out.println("++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++");
        System.out.println("\n");

        //enter the name of the football club
        System.out.println("Enter the name of the Club : ");
        String clubName = User_Input.nextLine().toLowerCase();
        System.out.println("\n");

        //if the user enters only a space to the club name will take the club name again
        while (clubName.equals("")) {
           System.out.println("ERROR ! Enter clubName Again : ");
           clubName = User_Input.nextLine().toLowerCase();
           System.out.println("\n");
        }

        //if the user enters another character except strings will take the club name again
        while (!clubName.matches("[a-zA-Z]+\\s?[a-zA-Z]+\\s?[a-zA-Z]*$")) {
        System.out.println("Enter a String value for the Club Name...Enter the name of the Club :
        ");
           clubName = User_Input.nextLine().toLowerCase();
           System.out.println("\n");
        }

        //enter the name of the country of the club
        System.out.println("Enter the country of the Club : ");
        String country = User_Input.nextLine().toLowerCase();
        System.out.println("\n");
```

```
//if the user enters a space to the country of the club, country will be taken again
while (country.equals("")) {
   System.out.println("ERROR ! Enter Country Again : ");
   country = User_Input.nextLine().toLowerCase();
   System.out.println("\n");
}

//if the user enters any other character except a string, country will be taken again as a user input
while (!country.matches("[a-zA-Z]+\\s?[a-zA-Z]+\\s?[a-zA-Z]*$")) {
        System.out.println("Enter a String value for the Country...Enter the name of the Club : ");
        country = User_Input.nextLine().toLowerCase();
        System.out.println("\n");
}

//taking the location of the club
System.out.println("Enter the location(city) of the Club : ");
String location = User_Input.nextLine().toLowerCase();
System.out.println("\n");

//if the user enters a space to the location,taking the location again
while (location.equals("")) {
   System.out.println("ERROR ! Enter Location Again : ");
   location = User_Input.nextLine().toLowerCase();
   System.out.println("\n");
}
```

```
//if the user enters any other character except a string to the location, taking the location again

while (!location.matches("[a-zA-Z]+\\s?[a-zA-Z]+\\s?[a-zA-Z]*$")) {

 System.out.println("Enter a String value for the City of the club...Enter the name of the Club : ");

   location = User_Input.nextLine().toLowerCase();

   System.out.println("\n");

}


//choice what division the club should play

footballClubChoice:

while (true) {

        System.out.println("1. (U23 players) => University Players");//U23 for University players

        System.out.println("2. (U18 players) => School Players");//U18 for school players

        System.out.println("\n");


   //choice of the division

System.out.println("Do you want to proceed with University Football club or School Football club [Enter the number only (1 or 2)] : ");

   int footballClubChoice = User_Input.nextInt();

   System.out.println("\n");


   if (footballClubChoice == 1) {//user chosen university as the division

        System.out.println("<<<You have chosen UNIVERSITY FOOTBALL CLUB>>>");

      System.out.println("\n");


        User_Input.nextLine();//prevent of skipping the name of the university to the number of players
```

```
    System.out.println("Enter the name of the UNIVERSITY : ");//name of the
    university
String universityName = User_Input.nextLine().toLowerCase();
System.out.println("\n");

//if the user enters space to the university name,taking the user input again
while (universityName.equals("")) {
    System.out.println("ERROR ! Enter University Name Again...");
    universityName = User_Input.nextLine().toLowerCase();
    System.out.println("\n");
}

    //if the user enters any other character except a string to the university name,
    taking it again
while (!universityName.matches("[a-zA-Z]+\\s?[a-zA-Z]+\\s?[a-zA-Z]*$")) {
    System.out.println("Enter a String value for the University Name...Enter the name
    of the Club : ");
    universityName = User_Input.nextLine().toLowerCase();
    System.out.println("\n");
}

//enter the number of players in the university, with the reserved players
    System.out.println("Enter the number of players in the university with reserved
    played [total number of players] : ");
int universityNoPlayers = User_Input.nextInt();
System.out.println("\n");

//if the user enters the number of players which is less than 0, enter an error message
if (universityNoPlayers < 0) {
    System.out.println("ERROR ! No of players can't be a negative value...");
```

```
            System.out.println("\n");
            continue footballClubChoice;
        }

        //set the values to the university division
            footballClub = new UniversityFootballClub(universityName,
            universityNoPlayers, clubName, country, location, totalMatchesPlayed,
            noOfMatchesWon, noOfMatchesLost, noOfMatchesDraw, goalsScored,
            goalsReceived, pointsScored);

             premierLeagueManager.addToPremierLeague(footballClub);//call the add
            method from the premier league manager

            break mainLoopAdd;//break the loop after setting the values to the university
            division

    } else if (footballClubChoice == 2) {//if the chosen the school division
        System.out.println("<<<You have chosen SCHOOL FOOTBALL CLUB>>>");
        System.out.println("\n");

        User_Input.nextLine();
            System.out.println("Enter the name of the SCHOOL : ");//take the name of the
            school
        String schoolName = User_Input.nextLine().toLowerCase();
        System.out.println("\n");

        //if the user enters a space to the school name, taking it again
        while (schoolName.equals("")) {
            System.out.println("ERROR ! Enter School Name Again : ");
            schoolName = User_Input.nextLine().toLowerCase();
```

```
    System.out.println("\n");
}

//if the user enters any other character except a string, taking the school name again
while (!schoolName.matches("[a-zA-Z]+\\s?[a-zA-Z]+\\s?[a-zA-Z]*$")) {
    System.out.println("Enter a String value for the School Name...Enter the name of
    the Club : ");
    schoolName = User_Input.nextLine().toLowerCase();
    System.out.println("\n");
}

//enter the number of players in the school
    System.out.println("Enter the number of players in the school with reserved
    played [total number of players]  : ");
int schoolNoPlayers = User_Input.nextInt();
System.out.println("\n");

//if the user enters the number of players less than 0, printing an error message
if (schoolNoPlayers < 0) {
    System.out.println("No of players can't be a negative value...");
    System.out.println("\n");
    continue footballClubChoice;
}

//setting the school football club after getting the values
    footballClub = new SchoolFootballClub(schoolName, schoolNoPlayers,
    clubName, country, location, totalMatchesPlayed, noOfMatchesWon,
    noOfMatchesLost, noOfMatchesDraw, goalsScored, goalsReceived,
    pointsScored);
```

```
                premierLeagueManager.addToPremierLeague(footballClub);//calling the add to
                premier league method


                break mainLoopAdd;//break the loop after setting the values to the school division
            } else {
                //printing an error message if the user selects any other number in the division
                selection
                System.out.println("In this age you can't play football any of the age category
                mentioned above");
                System.out.println("\n");
                break mainLoopAdd;
            }
        }
    }
}


private static void deleteExistingClub() {

    User_Input.nextLine();
    mainLoopDelete:
    while (true) {
        System.out.println("Enter the CLUB NAME you want to delete : ");//enter the name of
        the club to be deleted
        String deleteClub = User_Input.nextLine().toLowerCase();
        System.out.println("\n");

        //if the user enters a space to the name of the club, taking the name of the club again
        while (deleteClub.equals("")) {
            System.out.println("ERROR ! Enter Club Name Again : ");
```

```
            deleteClub = User_Input.nextLine().toLowerCase();
            System.out.println("\n");
        }

        //if the user enters any other character except string, taking the club name again
        while (!deleteClub.matches("[a-zA-Z]+\\s?[a-zA-Z]+\\s?[a-zA-Z]*$")) {
                System.out.println("Enter a String value for the Club Name...Enter the name of
                the Club : ");
               deleteClub = User_Input.nextLine().toLowerCase();
               System.out.println("\n");
        }

        premierLeagueManager.deleteExistingClub(deleteClub);//calling the delete method again
        break mainLoopDelete;
    }

}

private static void displayStatisticSelectedClub() {

    User_Input.nextLine();
    mainLoopDisplayStats:
    while (true) {
        System.out.println("Enter the name of the Football Club : ");//taking thee name of the
        football club to display the stattistics
        String clubNameDisplay = User_Input.nextLine().toLowerCase();
        System.out.println("\n");

        //if the user enters a space to the club name, taking the club name again
```

```
        while (clubNameDisplay.equals("")) {
            System.out.println("ERROR ! Enter Club Name Again : ");
            clubNameDisplay = User_Input.nextLine().toLowerCase();
            System.out.println("\n");
        }

        //if the user enters any other character except string, taking the name again
        while (!clubNameDisplay.matches("[a-zA-Z]+\\s?[a-zA-Z]+\\s?[a-zA-Z]*$")) {
        System.out.println("Enter a String value for the Club Name...Enter the name of the Club :
        ");
            clubNameDisplay = User_Input.nextLine().toLowerCase();
            System.out.println("\n");
        }

        premierLeagueManager.displayStatisticSelectedClub(clubNameDisplay);//calling the
        display statistics methods from the premier league
        break mainLoopDisplayStats;
    }

}

private static void displayPremierLeagueTable() {

        premierLeagueManager.displayPremierLeagueTable();//calling the premier league table
        from the premier league manager
}

private static void addPlayedMatch() {
    mainLoopAddPlayedMatch:
```

```
        while (true) {
            User_Input.nextLine();
            System.out.println("Enter home team playing the premier league [Club name] : ");//taking the name of the home team to play a match
            String homeTeamPlaying = User_Input.nextLine().toLowerCase();
            System.out.println("\n");

            //if the user enters a space to the name of the home team, take the home team again
            while (homeTeamPlaying.equals("")) {
               System.out.println("ERROR ! Enter Home Club Again : ");
               homeTeamPlaying = User_Input.nextLine().toLowerCase();
               System.out.println("\n");
            }

            //if the user enters a character except a string,take the home team again
            while (!homeTeamPlaying.matches("[a-zA-Z]+\\s?[a-zA-Z]+\\s?[a-zA-Z]*$")) {
                    System.out.println("Enter a String value for the Home Team...Enter the name of the Club : ");
                     homeTeamPlaying = User_Input.nextLine().toLowerCase();
                     System.out.println("\n");
            }

            //taking the name of the opponent team
            System.out.println("Enter the opponent team playing the premier league [Club name]: ");
            String opponentTeamPlaying = User_Input.nextLine().toLowerCase();
            System.out.println("\n");

            //if the user enters a space to the name of the opponent team, take the opponent team again
```

```
while (opponentTeamPlaying.equals("")) {
   System.out.println("ERROR ! Enter Opponent Team Again : ");
   opponentTeamPlaying = User_Input.nextLine().toLowerCase();
   System.out.println("\n");
}

//if the user enters a character except a string,take the opponent team again
while (!opponentTeamPlaying.matches("[a-zA-Z]+\\s?[a-zA-Z]+\\s?[a-zA-Z]*$")) {
        System.out.println("Enter a String value for the Opponent Team...Enter the name
        of the Club : ");
        opponentTeamPlaying = User_Input.nextLine().toLowerCase();
        System.out.println("\n");
}

//take the day of the match played
System.out.println("Please Enter the Day of the match played: ");
int day = User_Input.nextInt();

//taking the month of the match played
System.out.println("Please Enter the Month of the match played: ");
int month = User_Input.nextInt();

//year of the match played
System.out.println("Year of the match played : 2020");
int year=2020;
System.out.println("\n");

System.out.println("Enter the number of goals scored by the Home Team: ");//goals
scored by the home team
```

```
int goalsScoredHomeTeam = User_Input.nextInt();
System.out.println("\n");

//if the user enters the goals scored by the home team as less than zero, printing an error message
if (goalsScoredHomeTeam < 0) {
        System.out.println("ERROR ! Goals Scored by the home team can't be a negative value...Re-Enter it again : ");
       System.out.println("\n");
      continue mainLoopAddPlayedMatch;
}

//taking the goals scored by the opponent team
System.out.println("Enter the number of goals scored by the Opponent Team : ");
int goalsScoredOpponentTeam = User_Input.nextInt();
System.out.println("\n");

//if the user enters the goals scored by the opponent team as less than zero, printing an error message
if (goalsScoredOpponentTeam < 0) {
System.out.println("ERROR ! Goals Scored by the opponent team can't be a negative value...Re-Enter it again : ");
   System.out.println("\n");
   continue mainLoopAddPlayedMatch;
}

//setting the the date to the date constructor
DateMatchesPlayed dateMatchPlaying = new DateMatchesPlayed(day, month, year);
```

```
        //calling the add played match from the premier league
        premierLeagueManager.addPlayedMatch(homeTeamPlaying, opponentTeamPlaying,
        dateMatchPlaying, goalsScoredHomeTeam, goalsScoredOpponentTeam);
        break mainLoopAddPlayedMatch;
    }
}

private static void saveInAFile()  {
        premierLeagueManager.saveInAFile();//calling the saving method from the premier
        league manager
}

private static void premierLeagueGUI() throws URISyntaxException, IOException {

    //open localhost:9000 and localhost:4200
    ProcessBuilder processBuilderPlayFramework=new ProcessBuilder();
        processBuilderPlayFramework.command("cmd.exe","/c","start microsoft-
        edge:http://localhost:9000");

    ProcessBuilder processBuilderAngular=new ProcessBuilder();
        processBuilderAngular.command("cmd.exe","/c","start microsoft-
        edge:http://localhost:4200");



    try {
        //start the cmds to run the playframework and angular projects
        processBuilderPlayFramework.start();
```

```
            processBuilderAngular.start();
        }catch (Exception exception){
            System.out.println(exception);
        }
}
}
```

HomeController

```
package controllers;

import com.fasterxml.jackson.databind.JsonNode;

import play.libs.Json;

import play.mvc.*;

import java.io.*;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;


/**

 * This controller contains an action to handle HTTP requests

 * to the application's home page.

 */

public class HomeController extends Controller {


    /**

     * An action that renders an HTML page with a welcome message.

     * The configuration in the <code>routes</code> file means that

     * this method will be called when the application receives a

     * <code>GET</code> request with a path of <code>/</code>.

     */

    //convert football club arraylist to json format

    public Result getFootballClubsToJson() {

        List<FootballClub> footballClubs = footballClubs_readFromFile();
```

```
    JsonNode jsonFootballClubs = Json.toJson(footballClubs);

    return ok(jsonFootballClubs);

}

private List<FootballClub> footballClubs_readFromFile() {
    List<FootballClub> list_of_footballClubs = new ArrayList<>();

    try {
        //Creating a stream to read the objects in the text file
        FileInputStream fileInputStream1 = new
        FileInputStream("footballClubPremierLeague.txt");
        ObjectInputStream objectInputStream1 = new ObjectInputStream(fileInputStream1);

        //read to end of the file and add to the arraylist
        while (true) {
            FootballClub footballClub = (FootballClub) objectInputStream1.readObject();

            list_of_footballClubs.add(footballClub);
        }

    } catch (ClassNotFoundException classNotFoundException) {//exception for class not
    found
```

```
            System.out.println("ERROR ! Class not found Exception has occurred");
            System.out.println("\n");
        } catch (FileNotFoundException fileNotFoundException) {
            System.out.println("ERROR ! File not found Exception has occurred");
            System.out.println("\n");
        } catch (EOFException eofException) {//exception for end of file
            System.out.println("ERROR ! End of File Exception has occurred");
            System.out.println("\n");
        } catch (IOException ioException) {
            ioException.printStackTrace();
        }
        Collections.sort(list_of_footballClubs,Collections.reverseOrder());
        return list_of_footballClubs;
    }

    /*Match Simulation*/

    //convert matchsimulation arraylist to json
    public Result getMatchesPlayedToJson() {
        List<MatchSimulation> matchSimulation = matchesPlayed_readFromFile();

        JsonNode jsonMatchesPlayed = Json.toJson(matchSimulation);

        return ok(jsonMatchesPlayed);
```

```
}

private List<MatchSimulation> matchesPlayed_readFromFile() {
   List<MatchSimulation> playedMatchSimulation = new ArrayList<>();

   try {
      //Creating a stream to read the objects in the text file
      FileInputStream fileInputStream = new FileInputStream("matchSimulation.txt");
      ObjectInputStream objectInputStream = new ObjectInputStream(fileInputStream);

      //read to end of the file and add to the arraylist
      while (true) {
       MatchSimulation matchSimulation = (MatchSimulation)
      objectInputStream.readObject();

         playedMatchSimulation.add(matchSimulation);
      }

   } catch (ClassNotFoundException classNotFoundException) {//exception for class not
   found
      System.out.println("ERROR ! Class not found Exception has occurred");
      System.out.println("\n");
   } catch (FileNotFoundException fileNotFoundException) {
      System.out.println("ERROR ! File not found Exception has occurred");
      System.out.println("\n");
```

```
        } catch (EOFException eofException) {//exception for end of file
            System.out.println("ERROR ! End of File Exception has occurred");
            System.out.println("\n");
        } catch (IOException ioException) {
            ioException.printStackTrace();
        }
        return playedMatchSimulation;
    }
}
```

RandomMatchController

```
package controllers;


import com.fasterxml.jackson.databind.JsonNode;

import play.libs.Json;

import play.mvc.*;


import java.io.*;

import java.util.ArrayList;

import java.util.List;

import java.util.Random;


/**

 * This controller contains an action to handle HTTP requests

 * to the application's home page.

 */

public class RandomMatchAngularController extends Controller {

    /**

     * An action that renders an HTML page with a welcome message.

     * The configuration in the <code>routes</code> file means that

     * this method will be called when the application receives a

     * <code>GET</code> request with a path of <code>/</code>.

     */

    //convert randommatch arraylist to json

    public Result getRandomMatchToJson() {
```

```
        List<MatchSimulation> randomMatches = randomMatches_readFromFile();

        JsonNode jsonRandomMatchSimulation = Json.toJson(randomMatches);

        return ok(jsonRandomMatchSimulation);


    }

    private List<MatchSimulation> randomMatches_readFromFile() {
        List<FootballClub> list_of_footballClubs = new ArrayList<>();
        List<MatchSimulation> randomMatches = new ArrayList<>();

        try {
            //Creating a stream to read the objects in the text file
            FileInputStream fileInputStream1 = new
FileInputStream("footballClubPremierLeague.txt");
            ObjectInputStream objectInputStream1 = new ObjectInputStream(fileInputStream1);

            //read to end of the file and add to the arraylist
            while (true) {
                FootballClub footballClub = (FootballClub) objectInputStream1.readObject();

                list_of_footballClubs.add(footballClub);
            }
```

```
} catch (ClassNotFoundException classNotFoundException) {//exception for class not found
    System.out.println("ERROR ! Class not found Exception has occurred");
    System.out.println("\n");
} catch (FileNotFoundException fileNotFoundException) {
    System.out.println("ERROR ! File not found Exception has occurred");
    System.out.println("\n");
} catch (EOFException eofException) {//exception for end of file
    System.out.println("ERROR ! End of File Exception has occurred");
    System.out.println("\n");
} catch (IOException ioException) {
    ioException.printStackTrace();
}
//random home team and opponent team
Random randomHomeTeam = new Random();
Random randomOpponentTeam = new Random();

//random date for random match
Random date=new Random();

//random goals score by the home team
int goalsScoredHomeTeam = randomHomeTeam.nextInt(11);
//random goals scored by the opponent team
int goalsScoredOpponentTeam = randomOpponentTeam.nextInt(11);
```

```
//generate random dates
int dayRandom= date.nextInt(31)+1;
int monthRandom=date.nextInt(12)+1;
int yearRandom=2020;

//set the random dates for the date constructor
   DateMatchesPlayed dateMatchesPlayed =new
   DateMatchesPlayed(dayRandom,monthRandom,yearRandom);

clubSameLoop:
while (true) {

   //generate random home team name
   int randomGenerateHomeTeam =
   randomHomeTeam.nextInt(list_of_footballClubs.size());
   FootballClub randomElementHomeTeam =
   list_of_footballClubs.get(randomGenerateHomeTeam);

   //generate random opponent team name
   int randomGenerateOpponentTeam =
randomOpponentTeam.nextInt(list_of_footballClubs.size());
   FootballClub randomElementOpponentTeam =
list_of_footballClubs.get(randomGenerateOpponentTeam);

   //random home team should be equal to the home team in the football club list
```

```
        if
(!(randomElementHomeTeam.getClubName().equals(randomElementOpponentTeam.getClubNa
me()))) {

            //check whether the random home team and opponent team is university sports club

                    if ((randomElementHomeTeam instanceof UniversityFootballClub &&
        randomElementOpponentTeam instanceof UniversityFootballClub) ||

                    //check whether the random home team and opponent team is school sports club

        randomElementHomeTeam instanceof SchoolFootballClub &&
        randomElementOpponentTeam instanceof SchoolFootballClub) {




            //set the values to the match simulation constructor

         MatchSimulation matchSimulation=new
        MatchSimulation(randomElementHomeTeam.getClubName(),randomElementOpponent
        Team.getClubName(),
        dateMatchesPlayed,goalsScoredHomeTeam,goalsScoredOpponentTeam);

            randomMatches.add(matchSimulation);

            //System.out.println(randomMatches);


            boolean homeClubFound = false;//to find the home club entered by the user

            boolean opponentClubFound = false;//to find the opponent club entered by the user


            boolean isClubUniversity = false;//to find the club entered by the user belongs to
            which division

            FootballClub homeClub = null;//taking a variable to set the relevant attributes
            related to that particular football club(home club)
```

```
for (FootballClub footballClub : list_of_footballClubs) {

    if
(footballClub.getClubName().equals(randomElementHomeTeam.getClubName())) {//if the
home club entered by the user is in the football club arraylist

        if (footballClub instanceof UniversityFootballClub) {//and of the home club os
a university football club

            isClubUniversity = true;//making the boolean value to true as the home club
is a university football club.



        }



        //else if the football club entered by the user is a school football club

            homeClub = footballClub;//take the specific club name entered by the user
        and the relevant features of that club name into the home club variable



        homeClubFound = true;//as the home club is found making the boolean value
        to true

    }

}

    FootballClub opponentClub = null;//taking a variable to set the relevant attributes
    related to that particular football club(opponent club)



for (FootballClub footballClub : list_of_footballClubs) {

    if
((footballClub.getClubName().equals(randomElementOpponentTeam.getClubName()))) {//if the
opponent club entered by the user is in the list of football clubs

        if (isClubUniversity == true) {//making the boolean value to true as it a
university football club

            if (footballClub instanceof UniversityFootballClub) {
```

```
            isClubUniversity=true;
          }
        }

        opponentClub=footballClub;
        opponentClubFound=true;
    }

}
```

if (homeClubFound == true && opponentClubFound == true) {//if the home club and the opponent club entered by the user, both are found adding the elements to the arraylist and setting it to the match simulation class

```
    homeClub.setNoOfMatchesPlayed(homeClub.getNoOfMatchesPlayed() +
    1);//increase the number of matches played by one

    homeClub.setGoalsScored(homeClub.getGoalsScored() +
    goalsScoredHomeTeam);//updating the goals scored the home team

    homeClub.setGoalsReceived(homeClub.getGoalsReceived() +
    goalsScoredOpponentTeam);//updating the goals received by the home team

    opponentClub.setNoOfMatchesPlayed(opponentClub.getNoOfMatchesPlayed() +
    1);//increase the number of matches played by one

    opponentClub.setGoalsScored(opponentClub.getGoalsScored() +
    goalsScoredOpponentTeam);//updating the goals scored the opponent team

    opponentClub.setGoalsReceived(opponentClub.getGoalsReceived() +
    goalsScoredHomeTeam);//updating the goals received by the opponent team
```

```
if (goalsScoredHomeTeam > goalsScoredOpponentTeam) {//if the goals scored by home team is greater than the goals scored by the opponent team

    homeClub.setPointsScored(homeClub.getPointsScored() + 3);//increasing the points of the home team by 3

    homeClub.setMatchesWon(homeClub.getMatchesWon() + 1);//increasing the number of matches won by the home team by one

    opponentClub.setMatchesLost(opponentClub.getMatchesLost() + 1);//increasing the number of matches lost by the opponent team by one

  }

  if (goalsScoredHomeTeam < goalsScoredOpponentTeam) {//if the goals scored by opponent team is greater than the goals scored by the home team

    opponentClub.setPointsScored(opponentClub.getPointsScored() + 3);//increasing the points of the opponent team by 3

    opponentClub.setMatchesWon(opponentClub.getMatchesWon() + 1);//increasing the number of matches won by the opponent team by one

    homeClub.setMatchesLost(homeClub.getMatchesLost() + 1);//increasing the number of matches lost by the home team by one

  }

  if (goalsScoredHomeTeam == goalsScoredOpponentTeam) {//if the goals scored by the home team and the opponent team is equal

  homeClub.setPointsScored(homeClub.getPointsScored() + 1);//increasing the number of points scored by the home club by one

    opponentClub.setPointsScored(opponentClub.getPointsScored() + 1);//increasing the number og points scored by the opponent club by one

    homeClub.setMatchesDrawn(homeClub.getMatchesDrawn() + 1);//increasing the number of matches drawn by the home club by one

    opponentClub.setMatchesDrawn(opponentClub.getMatchesDrawn() + 1);//increasing the number of matches drawn by the opponent club by one

  }

}
```

```
                break clubSameLoop;
            } else {
                //continue the loop until the home team and the opponent team is not equal
                continue clubSameLoop;
            }
        } else {
                //continue the loop until the home team and the opponent team is in the same
                division to play the match
            continue clubSameLoop;
        }
    }
    return randomMatches;
  }
}
```

SortByDateAngular

```
package controllers;


import com.fasterxml.jackson.databind.JsonNode;
import play.libs.Json;
import play.mvc.*;
import java.io.*;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;


public class SortByDateController extends Controller {


    //convert the match simulation array to json
    public Result getSortByDateToJson() {
        List<MatchSimulation> matchSimulations = matchesPlayed_readFromFile();


        JsonNode jsonRandomMatchSimulation = Json.toJson(matchSimulations);


        return ok(jsonRandomMatchSimulation);



    }
    private List<MatchSimulation> matchesPlayed_readFromFile() {
        List<MatchSimulation> playedMatchSimulation = new ArrayList<>();


        try {
            //Creating a stream to read the objects in the text file
```

```
        FileInputStream fileInputStream = new FileInputStream("matchSimulation.txt");
        ObjectInputStream objectInputStream = new ObjectInputStream(fileInputStream);

        //read to end of the file and add to the arraylist
        while (true) {
         MatchSimulation matchSimulation = (MatchSimulation)
        objectInputStream.readObject();

           playedMatchSimulation.add(matchSimulation);
        }

            } catch (ClassNotFoundException classNotFoundException) {//exception for class
        not found
        System.out.println("ERROR ! Class not found Exception has occurred");
        System.out.println("\n");
      } catch (FileNotFoundException fileNotFoundException) {
        System.out.println("ERROR ! File not found Exception has occurred");
        System.out.println("\n");
      } catch (EOFException eofException) {//exception for end of file
        System.out.println("ERROR ! End of File Exception has occurred");
        System.out.println("\n");
      } catch (IOException ioException) {
        ioException.printStackTrace();
      }
      Collections.sort(playedMatchSimulation);
      return playedMatchSimulation;
    }

}
```

Premier League championship frontend

app.component.html

```
<!DOCTYPE html>

<html><!-- Start the html tag -->

   <head><!-- Start the head tag -->

      <title>Premiere League Manager</title><!-- title of the html page -->

   </head><!-- End of the head tag -->


   <body><!-- Start the body tag -->

      <ul>

         <!-- buttons which perform sort wins,goals scored,points in descending order,button
         which sorts the dates in ascending order,button which play a random match-->

         <li> <button [disabled]="!footballClubs || footballClubs.length===0?true:false"
         (click)="sortNoOfWins()" class="sortWinsButton" >Sort by Number of
         Wins</button> 

         </li>

         <li> <button [disabled]="!footballClubs || footballClubs.length===0?true:false"
         (click)="sortGoalsScored()" class="sortGoalsScoredButton">Sort by Number of Goals
         Scored</button> 

         </li>

         <li> <button [disabled]="!footballClubs || footballClubs.length < 2?true:false"
         class="randomMatchButton" (click)="randomMatch()">Random Match</button> 

         </li>

         <li> <button [disabled]="!sortByDateAscendingOrder ||
         sortByDateAscendingOrder.length===0?true:false" class="sortByDateButton"
         (click)="sortByDate()" id="sortByDateButton">Sort by Date</button>

         </li>
```

```
 <li> <button [disabled]="!footballClubs || footballClubs.length===0?true:false"
(click)="sortByPoints()" class="sortByPoints">Sort by Points</button>

</li>


<!-- Calender -->

<li style="float: right;" class="calender">

   <div class="calender">

        <label for="Calender" style="font-size: 20px;color:darkcyan;background-color:
       black;"><b>Calender:</b></label>

     <input type="date" id="calender" name="calender">

   </div>

</li>

<li style="float: left;" class="clock">

   <!-- Digital clock -->

   <div class="clock"> {{clock}} </div>

</li>

</ul>

<br>

<h1>PREMIERE LEAGUE MANAGER</h1><!-- heading of the page -->


<div class="search"><!-- division of the search bar and search button -->

   <!-- day to search -->

    <input maxlength="4" size="4" style="border-top-left-radius: 15px;border-bottom-left-
   radius:15px;padding-top: 5px;padding-bottom: 5px;" #search type="text" id="day"
   name="Day"  placeholder="Day" [(ngModel)]="day"/>

   <!-- month to search -->
```

```
 <input  maxlength="4" size="4" style="padding-top: 5px;padding-bottom: 5px;" #search
type="text" id="month" name="Month"  placeholder="Month" [(ngModel)]="month"/>

<!-- year to search -->

 <input  maxlength="4" size="4" style="border-top-right-radius: 15px;border-bottom-
right-radius:15px;padding-top: 5px;padding-bottom: 5px;"   #search type="text"
id="year" name="Year"  placeholder="Year" [(ngModel)]="year"/>

<!-- button to perform search action -->

<!-- if the matches played array is empty disable the button -->

 <button [disabled]="!matchSimulation || matchSimulation.length===0?true:false"
class="searchButton" (click)="searchButton()">Search</button>

</div>

<br>

<!-- modal which displays the dates which was searched by the user -->

 <div [style.visibility]="searchDatePopUpBox ? 'visible' : 'hidden'"
class="modal_filterByDate">

<div class="modal-content_filterByDate">

 <h1 style="margin-left: 220px;">FILTER BY DATE</h1><!-- Heading of the modal
box -->

  <!-- Close button in the filter by date modal -->

  <button class="closePopUpFilterDate"
(click)="closePopUpFilterByDate()">X</button>

  <br>

<!-- Table showing the results of after searching -->

<table>

 <tr>

    <!-- Headings of the table-->

    <th class="tablehead" *ngFor="let head of headings_filterByDate">{{head}}</th>

</tr>
```

```
    <!-- results came after searching -->
    <tr *ngFor="let matchesPlayed of tempSearchArray">
      <td *ngIf="dateFound==true">{{matchesPlayed.homeTeam}}</td>
      <td  *ngIf="dateFound==true">{{matchesPlayed.opponentTeam}}</td>
          <td  *ngIf="dateFound==true">Day : {{matchesPlayed.dateMatchPlayed.day}},
        Month : {{matchesPlayed.dateMatchPlayed.month}}, Year :
        {{matchesPlayed.dateMatchPlayed.year}}</td>
      <td  *ngIf="dateFound==true">{{matchesPlayed.goalsScoredHomeTeam}}</td>
      <td  *ngIf="dateFound==true">{{matchesPlayed.goalsScoredOpponentTeam}}</td>
    </tr>
</table>
    </div>
</div>
<!-- Table which shows all the football clubs -->
<!-- getting the tale id to refresh after adding a random match -->
<table id="tableFootballClubs">
<tr>
    <!-- heading of the football club table -->
    <th class="tablehead" *ngFor="let head of headings_footballClub">{{head}}</th>
</tr>

<!-- values of the football club table -->
<tr *ngFor="let clubs of footballClubs" > <!-- looping inside the football clubs array -->
    <td>{{clubs.clubName}}</td>
    <td>{{clubs.country}}</td>
    <td>{{clubs.location}}</td>
```

```
        <td>{{clubs.noOfMatchesPlayed}}</td>
        <td>{{clubs.matchesWon}}</td>
        <td>{{clubs.matchesLost}}</td>
        <td>{{clubs.matchesDrawn}}</td>
       <td>{{clubs.goalsScored}}</td>
       <td>{{clubs.goalsReceived}}</td>
        <td>{{clubs.pointsScored}}</td>
        <td>{{clubs.universityName}}</td>
        <td>{{clubs.schoolName}}</td>
        <td>{{clubs.noOfPlayers}}</td>
    </tr>
</table>
    <!-- Modal which displays the random matches playing -->
        <div  [style.visibility]="popUpRandomMatch ? 'visible' : 'hidden'"
       class="modal_randomMatches">
       <div class="modal-content_randomMatches" >
          <!-- Heading of the modal which plays the random matches -->
          <h1 style="margin-left: 220px;">RANDOM MATCH</h1>
          <!-- table which displays the random matches -->
          <table>
             <tr>
                <!-- heading of the random match modal table -->
                 <th class="tablehead" *ngFor="let head of
                headingRandomMatch">{{head}}</th>
             </tr>
```

```
        <!-- values of the random match playing table-->

        <tr *ngFor="let randomMatchValues of randomMatches" >

          <td>{{randomMatchValues.homeTeam}}</td>

          <td>{{randomMatchValues.opponentTeam}}</td>

                <td>Day : {{randomMatchValues.dateOfMatchPlayed.day}}, Month :
                {{randomMatchValues.dateOfMatchPlayed.month}}, Year :
                {{randomMatchValues.dateOfMatchPlayed.year}}</td>

          <td>{{randomMatchValues.goalsScoredHomeTeam}}</td>

          <td>{{randomMatchValues.goalsScoredOpponentTeam}}</td>

        </tr>

      </table>

    <!-- close the random match modal -->

    <button class="closePopUpRandomMatch"
  (click)="closePopUpRandomMatch()">OK</button>

  </div>

</div>

<!-- Modal which displays the dates sorted in ascending order -->

    <div [style.visibility]="sortByDatePopUpBox ? 'visible' : 'hidden'"
  class="modal_sortByDate">

  <div class="modal-content_sortByDate" >

    <!-- heading of the sort date modal -->

    <h1 style="margin-left: 220px;">SORT BY DATE</h1>

    <!-- close the dates sorts modal -->

    <button class="closePopUpSortDate" (click)="closePopUpsortByDate()">X</button>

    <br>

    <!-- table which displays the clubs which were played and sorted in scendong order by
  the date-->
```

```
        <table>
          <tr>
            <!-- headings of the table -->
            <th class="tablehead" *ngFor="let head of headings_filterByDate">{{head}}</th>
          </tr>
          <!-- values of the table -->
          <tr *ngFor="let sortByDate of sortByDateAscendingOrder" ><!-- loop inside the array
          of sortDateAscendingOrder-->
            <td>{{sortByDate.homeTeam}}</td>
            <td>{{sortByDate.opponentTeam}}</td>
                <td>Day : {{sortByDate.dateOfMatchPlayed.day}}, Month :
                {{sortByDate.dateOfMatchPlayed.month}}, Year :
                {{sortByDate.dateOfMatchPlayed.year}}</td>
            <td>{{sortByDate.goalsScoredHomeTeam}}</td>
            <td>{{sortByDate.goalsScoredOpponentTeam}}</td>
          </tr>
        </table>
          </div>
        </div>
<!-- Javascript -->
<script>
   var tableOfFootballCLubs=document.getElementById("tableFootballClubs");//getting the
   table id
    tableOfFootballCLubs.refresh();//refresh the table after playing a random match
 </script>
</body>
</html>
```

app.component.css

```
/* heading of the premier league manager */
h1{
  margin-left: 170px;
  font-family: 'Monospace      ';
  font-weight: bold;
  font-size: 50px;
}

/* body of the  web page*/
body{
  background-color: #E0FFFF;
  background-image: url(Images/premier-league-logo.jpg);
  background-repeat: no-repeat;
  background-size: auto;
  width:100%;
  height: 900px;
}

/* search field alignment */
.search{
    padding-left: 75%;
    margin-top: -3px;
    padding-bottom: 50px;
}
/* search button design and alignment */
.searchButton{
```

```
    transition-duration: 0.4s;
    border: 2px solid #4CAF50;
    margin-left: 10px;
    padding-top: 5px;
    padding-bottom: 5px;
    cursor: grab;
    border-radius: 25px;
}

/* search button hover */
.searchButton:hover {
    background-color: #4CAF50;
    color: white;
  }

/* table of football clubs */
table{
    width: 100%;
    line-height: 250%;
    overflow:auto;
    border-spacing: 0px;
}

/* table heading of football clubs */
.tablehead{
    background-color:  #00cc99;
}
```

```
/* table border */
table, th, td {
  border: 1px solid black;
  padding: 5px;
}

/* stripes in the table rows */
tr:nth-child(even) {background-color: #f2f2f2;}/*stripped rows*/
tr:nth-child(odd) {background-color:    #ccd9ff;}/*stripped rows*/

/* sort wins in descending order button, design and alignment */
.sortWinsButton{
    transition-duration: 0.4s;
    border: 2px solid #f44336;
    padding-top: 10px;
    padding-bottom: 10px;
    margin: 25px;
    cursor: grab;
}

/*  sort wins in descending order button, hover  */
.sortWinsButton:hover {
    background-color: #f44336;
    color: white;
  }


 /*  sort goals scored in descending order button, design and alignment  */
```

```
.sortGoalsScoredButton{
    transition-duration: 0.4s;
    border: 2px solid #f44336;
    padding-top: 10px;
    padding-bottom: 10px;
    margin: 25px;
    cursor: grab;
}

/*  sort goals scored in descending order button, hover  */
.sortGoalsScoredButton:hover {
    background-color: #f44336;
    color: white;
 }

 /*  random matches playing button, design and alignment  */
.randomMatchButton{
    transition-duration: 0.4s;
    border: 2px solid #f44336;
    padding-top: 10px;
    padding-bottom: 10px;
    margin: 25px;
    cursor: grab;
}


 /*  random matches playing button, hover  */
 .randomMatchButton:hover {
```

```
    background-color: #f44336;
    color: white;
  }

/*  sort date in descending order button, design and alignment  */
.sortByDateButton{
    transition-duration: 0.4s;
    border: 2px solid #f44336;
    padding-top: 10px;
    padding-bottom: 10px;
    margin: 25px;
    cursor: grab;
  }

  /*  sort date in descending order button, hover  */
  .sortByDateButton:hover {
    background-color: #f44336;
    color: white;
  }

  /*  sort points in descending order button, design and alignment  */
  .sortByPoints{
   transition-duration: 0.4s;
   border: 2px solid #f44336;
   padding-top: 10px;
   padding-bottom: 10px;
   margin: 28px;
   cursor: grab;
```

```
}

/*  sort points in descending order button, hover  */
.sortByPoints:hover {
  background-color: #f44336;
  color: white;
}

 /* The Modal which displays random matches */
.modal_randomMatches {
    /* display: none; Hidden by default */
    position: fixed; /* Stay in place */
    z-index: 1; /* plce on top */
    padding-top: 100px; /* Location of the box */
    left: 0;
    top: 0;
    width: 100%; /* width size*/
    height: 100%; /* height size*/
    overflow: auto; /* Enable scroll if needed */

  }

  /* Modal Content which displays random matches */
  .modal-content_randomMatches {
    background-color: #00d1cc; /* background color */
    margin: auto;
    padding: 20px;
    border: 1px solid #888;
```

```
    padding-bottom: 150px;
  }

 /* close the random match modal */
  .closePopUpRandomMatch{
    transition-duration: 0.4s;
    border: 2px solid #f44336;
    padding-top: 10px;
    padding-bottom: 10px;
    padding-right: 10px;
    padding-left: 10px;
    margin: 0px;
    float: right;
    margin-left: 850px;
    margin-top: 50px;
    cursor: grab;
}

/* close the random match modal button hover */
.closePopUpRandomMatch:hover {
    background-color: #f44336;
    color: rgb(70, 53, 53);
 }

/* The Modal which displays date sorted in descending order */
.modal_sortByDate {
    position: absolute; /* Stay in fixed place */
    z-index: 1; /* place on top */
```

```
  padding-top: 100px; /* Location of the box */
  left: 0;
  top: 0;
  width: 100%; /* width size */
  height: 100%; /* height size*/
  overflow: auto;/*If the height exceeds the default size insert a scrool bar*/
 }

 /* Modal Content which displays date sorted in descending order*/
 .modal-content_sortByDate {
  background-color:   #009eab;
  margin: auto;
  padding: 20px;
  border: 1px solid #888;
  padding-bottom: 80px;
 }

/* close modal of date sorted in descending order*/
 .closePopUpSortDate{
  transition-duration: 0.4s;
  border: 2px solid #f44336;
  padding-top: 10px;
  padding-bottom: 10px;
  padding-right: 10px;
  padding-left: 10px;
  margin: 0px;
  float: right;
  margin-left: 850px;
```

```
    margin-top: -40px;
    cursor: grab;
}

 /* close modal of date sorted in descending order*/
.closePopUpSortDate:hover {
    background-color: #f44336;
    color: rgb(70, 53, 53);
 }

 /* The Modal which displays dates which was filtered after searching */
.modal_filterByDate {
  position: absolute; /* Stay fixed */
  z-index: 1; /* plce on top */
  padding-top: 100px; /* Location of the box */
  left: 0;
  top: 0;
  width: 100%; /* width size*/
  height: 100%; /* height size*/
  overflow: auto;/*If the height exceeds the default size insert a scrool bar*/

}

/* Modal Content which displays dates which was filtered after searching  */
.modal-content_filterByDate {
  background-color: #4682B4;
  margin: auto;
  padding: 20px;
```

```
  border: 1px solid #888;
  padding-bottom: 150px;

}

/* close the modal which displays dates which was filtered after searching button,design and alignment */
.closePopUpFilterDate{
  transition-duration: 0.4s;
  border: 2px solid #f44336;
  padding-top: 10px;
  padding-bottom: 10px;
  padding-right: 10px;
  padding-left: 10px;
  margin: 0px;
  float: right;
  margin-left: 850px;
  margin-top: -40px;
  cursor: grab;
}

/* close the modal which displays dates which was filtered after searching button,hover */
.closePopUpFilterDate:hover {
  background-color: #f44336;
  color: rgb(70, 53, 53);
}

/* calender design and alignment */
.calender{
```

```
  margin-left: 1150px;
  margin-top: -40px;
  font-family: Orbitron;

}

/* clock design and alignment */
.clock{
  margin-top: -10px;
  color:darkcyan;
  font-size: 20px;
  font-family: Orbitron;
  letter-spacing: 7px;
  font-weight: bold;
  background-color: black;
  width: 130%;
  text-align: center;
  margin-left: -250px;
}
/* header of the page */
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #333;
  padding-bottom: -5px;
}
```

```
li {
  float: left;
}
```

app.component.ts

```
import { Component} from '@angular/core';
import { FootballClubs } from './frontendClasses/FootballClubs';
import { FreeapiService } from './apiService/apiServices.service';
import {RandomMatches} from './frontendClasses/randomMatches';
import {SortByDate} from './frontendClasses/SortByDate';
import { MatchSimulation } from './frontendClasses/MatchSimulation';



@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  popUpRandomMatch = false;//random match popup set to false
  sortByDatePopUpBox = false;//sort date pop up set to false
  searchDatePopUpBox = false;//search date pop up set to false
  title = 'Premier League Manager';//title of the angular project

  footballClubs:FootballClubs[];//football club array
  matchSimulation:MatchSimulation[];//matches played array
  randomMatches:RandomMatches[];//random match playing array
  sortByDateAscendingOrder:SortByDate[];//sort date in ascending order array

  //day,month and year which contain the user input values in the search field
  day:number;
```

```
  month:number;
  year:number;

//search field elements
dateFound=false;
alertBox=false;
homeTeam:string;
opponentTeam:string;
dateOfMatchPlayed:{day:number,month:number,year:number};
goalsScoredHomeTeam:number;
goalsScoredOpponentTeam:number;
tempSearchArray=new Array();

//random match elements
homeTeamRandomMatchFound=false;
opponentTeamRandomMatchFound=false;
homeTeamRandomMatch:FootballClubs;
opponentTeamRandomMatch:FootballClubs;

//headings of the tables
public headings_footballClub=["Club Name", "Country", "Location","No Of Matches Played",
"Matches Won", "Matches Lost", "Matches Drawn", "Goals Scored", "Goals Received", "Points
Scored","University Name","School Name","No Of Players"];
public headings_filterByDate=["Home Team", "Opponent Team", "Date Of Match
Played","Goals Scored Home Team", "Goals Scored Opponent Team"];
public headingRandomMatch=["Home Team", "Opponent Team", "Date Of Match
Played","Goals Scored Home Team", "Goals Scored Opponent Team"];

//clock and clock handle
clock=""
```

```
clockHandle;
  constructor(private apiService:FreeapiService) {

  }

  ngOnInit(){

    //get the football clubs from the http://localhost:9000
    this.apiService.getFootballClubs()
    .subscribe(
      data =>{
        this.footballClubs = data;

        }
    );

    //get the football clubs from the http://localhost:9000/sortByDate
    this.apiService.getSortByDate()
    .subscribe(
      data =>{
          this.sortByDateAscendingOrder = data;

      }
    );

    //get the football clubs from the http://localhost:9000/matchesPlayed
    this.apiService.getMatchSimulation()
    .subscribe(
```

```
    data =>{
      this.matchSimulation = data;

    }
  );

  //code for the clock is taken from =>
 //https://stackblitz.com/edit/angular-clock-1-q2tuyq?file=src%2Fapp%2Fapp.component.html
  this.clockHandle = setInterval(()=>{
     this.clock = new Date().toLocaleString();},1000);
 }



//search function
searchButton(){

//empty the array in begining of each loop
this.tempSearchArray=[];

 //find if the date entered by the user is in the array
  for(let matchSimulationSearch of this.matchSimulation){
        if((this.day==matchSimulationSearch.dateOfMatchPlayed.day) &&
        (this.month==matchSimulationSearch.dateOfMatchPlayed.month) &&
        (this.year==matchSimulationSearch.dateOfMatchPlayed.year)){
       this.homeTeam=matchSimulationSearch.homeTeam;
       this.opponentTeam=matchSimulationSearch.opponentTeam;
       this.dateOfMatchPlayed={day:
matchSimulationSearch.dateOfMatchPlayed.day,month:matchSimulationSearch.dateOfMatchPla
yed.month,year:matchSimulationSearch.dateOfMatchPlayed.year}
       this.goalsScoredHomeTeam=matchSimulationSearch.goalsScoredHomeTeam;
```

```
        this.goalsScoredOpponentTeam=matchSimulationSearch.goalsScoredOpponentTeam;
        //if the date is in the array displaying the modal
        this.searchDatePopUpBox=true;
        //if the date is in the array making the datefound boolean value to true
        this.dateFound=true;
        //if the date is in the array pushing the relavent information to the temporary array

          this.tempSearchArray.push({homeTeam:matchSimulationSearch.homeTeam,opponentTe
          am:matchSimulationSearch.opponentTeam,dateMatchPlayed:matchSimulationSearch.dat
          eOfMatchPlayed,goalsScoredHomeTeam:matchSimulationSearch.goalsScoredHomeTea
          m,goalsScoredOpponentTeam:matchSimulationSearch.goalsScoredOpponentTeam});
      }
    }

  //if the length of the temporary array is 0 making the day,month and year text filds to null and
displaying an alert box
  if(this.tempSearchArray.length==0){
  for(let matchSimulationSearch of this.matchSimulation){
    //displaying an error message if one component of the date is found
    if((this.day==matchSimulationSearch.dateOfMatchPlayed.day) ||
(this.month==matchSimulationSearch.dateOfMatchPlayed.month) ||
(this.year==matchSimulationSearch.dateOfMatchPlayed.year)){
      this.day=null;
      this.month=null;
      this.year=null;
      alert("ERROR ! DATE CANNOT BE FOUND !!!");
      break;
    //displaying an error message if all the components of the date are not found
   }if(!((this.day==matchSimulationSearch.dateOfMatchPlayed.day) &&
(this.month==matchSimulationSearch.dateOfMatchPlayed.month) &&
(this.year==matchSimulationSearch.dateOfMatchPlayed.year))){
```

```
    this.day=null;
    this.month=null;
    this.year=null;
    alert("ERROR ! DATE CANNOT BE FOUND !!!");
     break;
   }
   break;
  }
}
}

//close the filter by date modal after the button click
closePopUpFilterByDate(){
  this.searchDatePopUpBox=false;

}

  // sort number of wins in descending order
   compareWins(object_1, object_2, key){
    const obj_1 = object_1[key];
    const obj_2 = object_2[key];

    if (obj_1 > obj_2) {//if the object_1 is greater than object_2 shift the relavent row upper
      return -1
    }

    return 0//else do nothing
  }
```

```
  //button click action for sort number of wins
  sortNoOfWins(){
    this.footballClubs.sort((a,b)=>{//sort the matches won column by calling the above compare wins method
      return this.compareWins(a, b, 'matchesWon')

    }
    )
  }

//sort goals scored in descending order
compareGoalsScored(object_1, object_2, key){
  const obj_1 = object_1[key];
  const obj_2 = object_2[key];

  if (obj_1 > obj_2) {//if the object_1 is greater than object_2 shift the relavent row upper
    return -1
  }

  return 0//else do nothing
}

  //button click action for sort number of goals scored
sortGoalsScored(){

  this.footballClubs.sort((a,b)=>{//sort the goals scored column by calling the above compare goals scored method
```

```
    return this.compareGoalsScored(a, b, 'goalsScored')

  }
  )
}
//sort points scored in descending order
comparedPointsScored(object1, object2, key){
  const obj1 = object1[key];
  const obj2 = object2[key];

  if (obj1 > obj2) {//if the object_1 is greater than object_2 shift the relavent row upper
    return -1
  }

  return 0//else do nothing
}

sortByPoints(){

  this.footballClubs.sort((a,b)=>{//sort the points scored column by calling the above compare points scored method
    return this.comparedPointsScored(a, b, 'pointsScored')

  }
  )
}

//random match button click action method
randomMatch(){
```

```
  //call the api after every button click(new random match will be generated)
  this.apiService.getRandomMatches()
  .subscribe(
    data =>{
        this.randomMatches = data;
    }

  );
//show the random match modal
this.popUpRandomMatch=true;

}

//close the random match modal
closePopUpRandomMatch(){

this.popUpRandomMatch=false;

//if the club name is in the match simulation home team make the boolean home team random match to true and set the football club to the home team random match boolean value
for(let footballClubsRandomMatch of this.footballClubs){
  for(let matchSimulationRandomMatch of this.randomMatches){

if(footballClubsRandomMatch.clubName.includes(matchSimulationRandomMatch.homeTeam))
{
   this.homeTeamRandomMatchFound=true;
    this.homeTeamRandomMatch=footballClubsRandomMatch;
```

```
 }
 }
}

//if the club name is in the match simulation opponent team make the boolean opponent team random match to true and set the football club to the opponent team random match boolean value
for(let footballClubsRandomMatch of this.footballClubs){
  for(let matchSimulationRandomMatch of this.randomMatches){

if(footballClubsRandomMatch.clubName.includes(matchSimulationRandomMatch.opponentTeam)){
   this.opponentTeamRandomMatchFound=true;
    this.opponentTeamRandomMatch=footballClubsRandomMatch;

 }
 }
 }

//if both home team random match and opponent team random match found true
if (this.homeTeamRandomMatchFound==true && this.opponentTeamRandomMatchFound==true){
    for(let matchSimulationRandomMatch of this.randomMatches){
      //increase the number of matches played by one

this.homeTeamRandomMatch.noOfMatchesPlayed=this.homeTeamRandomMatch.noOfMatchesPlayed+1;
      //updating the goals scored of the home team

this.homeTeamRandomMatch.goalsScored=this.homeTeamRandomMatch.goalsScored+matchSimulationRandomMatch.goalsScoredHomeTeam;
```

```
    //updating the goals received by the home team

this.homeTeamRandomMatch.goalsReceived=this.homeTeamRandomMatch.goalsReceived+ma
tchSimulationRandomMatch.goalsScoredOpponentTeam;


    //increase the number of matches played by one

this.opponentTeamRandomMatch.noOfMatchesPlayed=this.opponentTeamRandomMatch.noOf
MatchesPlayed+1;
    //updating the goals scored of the opponent team

this.opponentTeamRandomMatch.goalsScored=this.opponentTeamRandomMatch.goalsScored+
matchSimulationRandomMatch.goalsScoredOpponentTeam;
    //updating the goals received by the opponent team

this.opponentTeamRandomMatch.goalsReceived=this.opponentTeamRandomMatch.goalsRecei
ved+matchSimulationRandomMatch.goalsScoredHomeTeam;


    //if the goals scored by home team is greater than the goals scored by the opponent team
   if(matchSimulationRandomMatch.goalsScoredHomeTeam >
matchSimulationRandomMatch.goalsScoredOpponentTeam){
    //increasing the points of the home team by 3
    this.homeTeamRandomMatch.pointsScored=this.homeTeamRandomMatch.pointsScored+3;
    //increasing the number of matches won by the home team by one
    this.homeTeamRandomMatch.matchesWon=this.homeTeamRandomMatch.matchesWon+1;
    //increasing the number of matches lost by the opponent team by one

this.opponentTeamRandomMatch.matchesLost=this.opponentTeamRandomMatch.matchesLost
+1;


   }
```

```
    //if the goals scored by opponent team is greater than the goals scored by the home team
    if(matchSimulationRandomMatch.goalsScoredHomeTeam <
matchSimulationRandomMatch.goalsScoredOpponentTeam){
      //increasing the points of the opponent team by 3

this.opponentTeamRandomMatch.pointsScored=this.opponentTeamRandomMatch.pointsScored
+3;
      //increasing the number of matches won by the opponent team by one

this.opponentTeamRandomMatch.matchesWon=this.opponentTeamRandomMatch.matchesWon
+1;
     //increasing the number of matches lost by the home team by one
      this.homeTeamRandomMatch.matchesLost=this.homeTeamRandomMatch.matchesLost+1;
    }


    //if the goals scored by the home team and the opponent team are equal
    if(matchSimulationRandomMatch.goalsScoredHomeTeam ==
matchSimulationRandomMatch.goalsScoredOpponentTeam){
      //increasing the number of points scored by the home club by one
      this.homeTeamRandomMatch.pointsScored=this.homeTeamRandomMatch.pointsScored+1;
      //increasing the number of points scored by the opponent club by one

this.opponentTeamRandomMatch.pointsScored=this.opponentTeamRandomMatch.pointsScored
+1;
      //increasing the number of matches drawn by the home club by one

this.homeTeamRandomMatch.matchesDrawn=this.homeTeamRandomMatch.matchesDrawn+1;
      //increasing the number of matches drawn by the opponent club by one

this.opponentTeamRandomMatch.matchesDrawn=this.opponentTeamRandomMatch.matchesDr
awn+1;
```

```
    }

}
}

//push the relavent data to the match simulation array
for(let matchSimulationRandomMatch of this.randomMatches){

        this.matchSimulation.push({homeTeam:matchSimulationRandomMatch.homeTeam,opponentTeam:matchSimulationRandomMatch.opponentTeam,dateOfMatchPlayed:matchSimulationRandomMatch.dateOfMatchPlayed,goalsScoredHomeTeam:matchSimulationRandomMatch.goalsScoredHomeTeam,

        goalsScoredOpponentTeam:matchSimulationRandomMatch.goalsScoredOpponentTeam});

     }

//push the relavent data to the sortdate ascending order array
for(let matchSimulationRandomMatch of this.randomMatches){

this.sortByDateAscendingOrder.push({homeTeam:matchSimulationRandomMatch.homeTeam,opponentTeam:matchSimulationRandomMatch.opponentTeam,dateOfMatchPlayed:matchSimulationRandomMatch.dateOfMatchPlayed,goalsScoredHomeTeam:matchSimulationRandomMatch.goalsScoredHomeTeam,
   goalsScoredOpponentTeam:matchSimulationRandomMatch.goalsScoredOpponentTeam});

 }
}

//sort date in ascending order button click action
sortByDate(){
```

```
    //first sort the date by day
        this.sortByDateAscendingOrder.sort((a, b) => (a.dateOfMatchPlayed.day <
        b.dateOfMatchPlayed.day ? -1 : 1));
    //second sort the date by month
        this.sortByDateAscendingOrder.sort((a, b) => (a.dateOfMatchPlayed.month <
        b.dateOfMatchPlayed.month ? -1 : 1));

  //open the modal sort date pop up box
  this.sortByDatePopUpBox=true;
}

//close the sort date pop up box after a button click
closePopUpsortByDate(){
  this.sortByDatePopUpBox=false;
}
}
```

apiService = > apiServices.service.ts

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';

@Injectable()
 export class FreeapiService {

 constructor(private httpclient:HttpClient) { }

 //call the localhost:9000 from the api as a json format
 getFootballClubs():Observable<any>{
  return this.httpclient.get("http://localhost:9000");
 }

 //call the localhost:9000/matchesPlayed from the api as a json format
  getMatchSimulation():Observable<any>{
    return this.httpclient.get("http://localhost:9000/matchesPlayed");
 }

  //call the localhost:9000/randomMatch from the api as a json format
  getRandomMatches():Observable<any>{
    return this.httpclient.get("http://localhost:9000/randomMatch");
  }

  //call the localhost:9000/sortByDate from the api as a json format
  getSortByDate():Observable<any>{
```

```
    return this.httpclient.get("http://localhost:9000/sortByDate");
  }
}
```

frontendClasses => FootballClubs.ts

```
//get the required details from the web(API) to display the football clubs
export class FootballClubs{
  clubName:string;
  country:string;
  location:string;
  noOfMatchesPlayed:number;
  matchesWon:number;
  matchesLost:number;
  matchesDrawn:number;
  goalsScored:number;
  goalsReceived:number;
  pointsScored:number;
  universityName:string;
  schoolName:string;
  noOfPlayers:number;
}
```

frontendClasses => MatchSimulation.ts

```
//get the required details from the web(API) to display the matches played
export class MatchSimulation{
  homeTeam:string;
  opponentTeam:string;
  dateOfMatchPlayed:{day:number,month:number,year:number};
```

```
    goalsScoredHomeTeam:number;
    goalsScoredOpponentTeam:number;

}
```

frontendClasses => RandomMatches.ts

```
//get the required details from the web(API) to display the random matches played
export class RandomMatches{
    homeTeam:string;
    opponentTeam:string;
    dateOfMatchPlayed:{day:number,month:number,year:number};
    goalsScoredHomeTeam:number;
    goalsScoredOpponentTeam:number;

}
```

frontendClasses => SortByDate.ts

```
//get the required details from the web(API) to display matches sorted by date
export class SortByDate{
    homeTeam:string;
    opponentTeam:string;
    dateOfMatchPlayed:{day:number,month:number,year:number};
    goalsScoredHomeTeam:number;
    goalsScoredOpponentTeam:number;

}
```

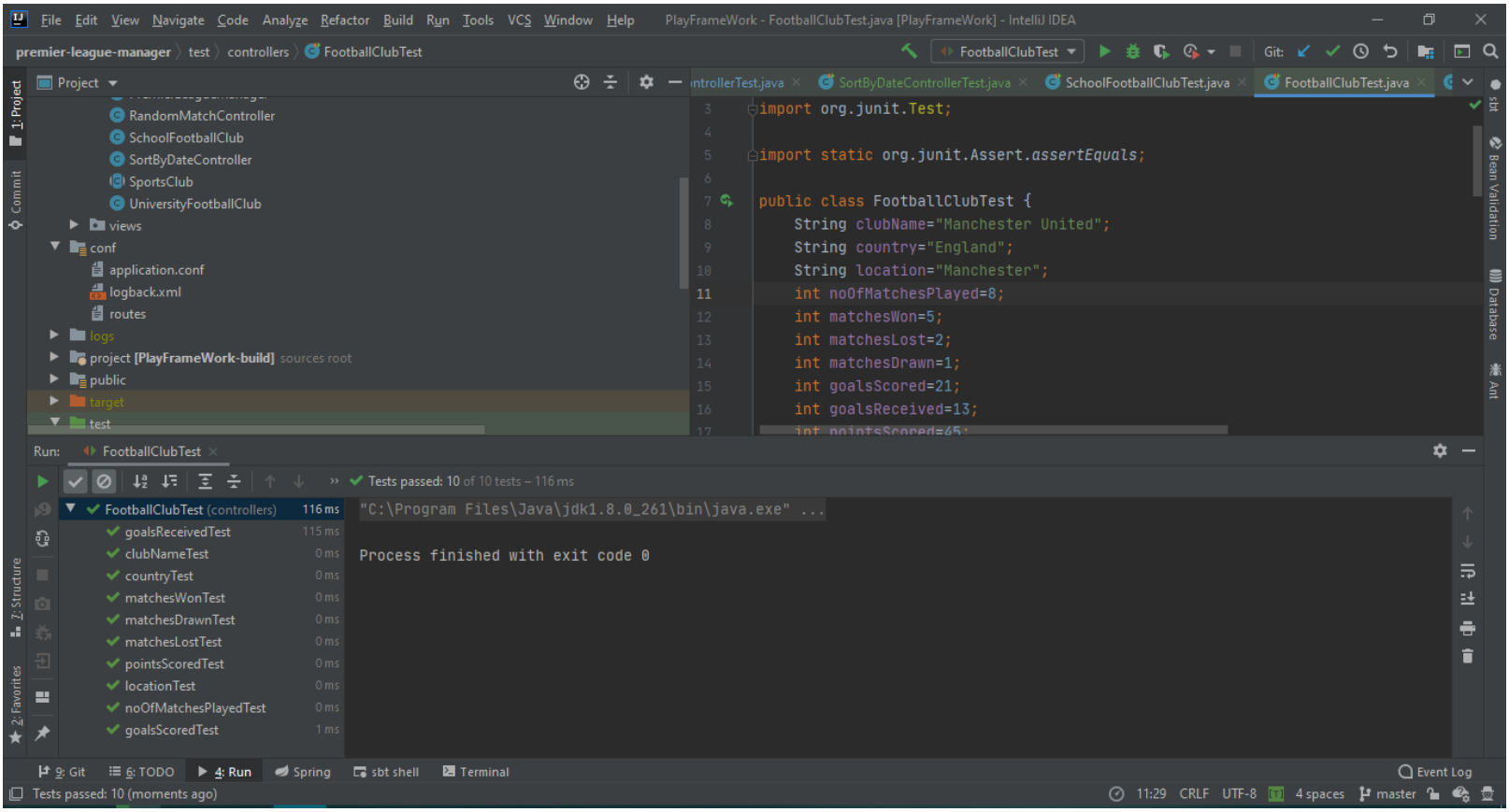
Unit Testing and Screenshots of the output

FootballClubTest

```
package controllers;

import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class FootballClubTest {
    String clubName="Manchester United";
    String country="England";
    String location="Manchester";
    int noOfMatchesPlayed=8;
    int matchesWon=5;
    int matchesLost=2;
    int matchesDrawn=1;
    int goalsScored=21;
    int goalsReceived=13;
    int pointsScored=45;

    SportsClub footballClubs=new
    FootballClub(clubName,country,location,noOfMatchesPlayed,matchesWon,
    matchesLost,matchesDrawn,goalsScored,goalsReceived,pointsScored);
    @Test
    public void clubNameTest() {
        assertEquals(clubName,footballClubs.getClubName());
    }
```

```
@Test
public void countryTest() {
   assertEquals(country,footballClubs.getCountry());
}

@Test
public void locationTest() {
   assertEquals(location,footballClubs.getLocation());
}

@Test
public void noOfMatchesPlayedTest() {
   assertEquals(noOfMatchesPlayed,footballClubs.getNoOfMatchesPlayed());
}

@Test
public void matchesWonTest() {
   assertEquals(matchesWon, footballClubs.getMatchesWon());
}

@Test
public void matchesLostTest() {
   assertEquals(matchesLost,footballClubs.getMatchesLost());
}

@Test
public void matchesDrawnTest() {
```

```
        assertEquals(matchesDrawn,footballClubs.getMatchesDrawn());
    }

    @Test
    public void goalsScoredTest() {
        assertEquals(goalsScored,((FootballClub) footballClubs).getGoalsScored());
    }

    @Test
    public void goalsReceivedTest() {
        assertEquals(goalsReceived,((FootballClub) footballClubs).getGoalsReceived());
    }

    @Test
    public void pointsScoredTest() {
        assertEquals(pointsScored,((FootballClub) footballClubs).getPointsScored());
    }

}
```


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
PlayFrameWork - FootballClubTest.java [PlayFrameWork] - IntelliJ IDEA
premier-league-manager test controllers FootballClubTest
Project
RandomMatchController
SchoolFootballClub
SortByDateController
SportsClub
UniversityFootballClub
views
conf
application.conf
logback.xml
routes
logs
project [PlayFrameWork-build] sources root
public
target
test
SortByDateControllerTest.java
SchoolFootballClubTest.java
FootballClubTest.java
import org.junit.Test;
import static org.junit.Assert.assertEquals;
public class FootballClubTest {
String clubName="Manchester United";
String country="England";
String location="Manchester";
int noOfMatchesPlayed=8;
int matchesWon=5;
int matchesLost=2;
int matchesDrawn=1;
int goalsScored=21;
int goalsReceived=13;
Run: FootballClubTest
Tests passed: 10 of 10 tests – 116 ms
FootballClubTest (controllers) 116 ms
goalsReceivedTest 115 ms
clubNameTest 0 ms
countryTest 0 ms
matchesWonTest 0 ms
matchesDrawnTest 0 ms
matchesLostTest 0 ms
pointsScoredTest 0 ms
locationTest 0 ms
noOfMatchesPlayedTest 0 ms
goalsScoredTest 1 ms
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ...
Process finished with exit code 0
9: Git 6: TODO 4: Run Spring sbt shell Terminal
Event Log
Tests passed: 10 (moments ago)
11:29 CRLF UTF-8 4 spaces master

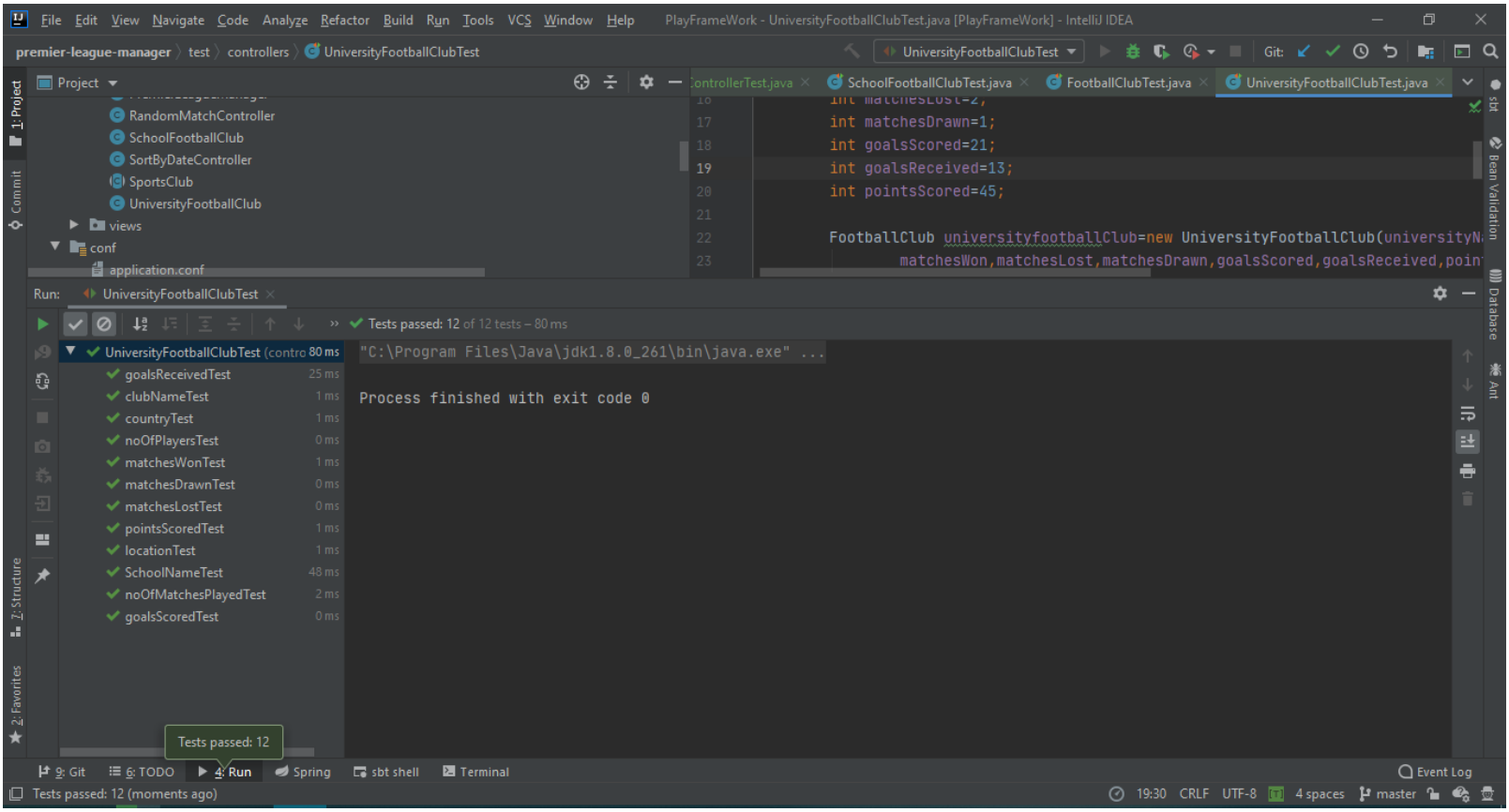
UniversityFootballClubTest

```
package controllers;

import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class UniversityFootballClubTest {

    String universityName="IIT";
    int noOfPlayers=15;//number of players in the school football club
    String clubName="Manchester United";
    String country="England";
    String location="Manchester";
    int noOfMatchesPlayed=8;
    int matchesWon=5;
    int matchesLost=2;
    int matchesDrawn=1;
    int goalsScored=21;
    int goalsReceived=13;
    int pointsScored=45;

    FootballClub universityfootballClub=new
    UniversityFootballClub(universityName,noOfPlayers,clubName,country,location,noOfMatc
    hesPlayed,matchesWon,matchesLost,matchesDrawn,goalsScored,goalsReceived,pointsScor
    ed);
```

```
@Test
public void SchoolNameTest() {
   assertEquals(universityName,((UniversityFootballClub)
   universityfootballClub).getUniversityName());

}

@Test
public void noOfPlayersTest() {
    assertEquals(noOfPlayers,((UniversityFootballClub)
   universityfootballClub).getNoOfPlayers());

}

@Test
public void clubNameTest() {
   assertEquals(clubName,universityfootballClub.getClubName());
}

@Test
public void countryTest() {
   assertEquals(country,universityfootballClub.getCountry());
}

@Test
public void locationTest() {
   assertEquals(location,universityfootballClub.getLocation());
}
```

```
@Test
public void noOfMatchesPlayedTest() {
   assertEquals(noOfMatchesPlayed,universityfootballClub.getNoOfMatchesPlayed());
}

@Test
public void matchesWonTest() {
   assertEquals(matchesWon, universityfootballClub.getMatchesWon());
}

@Test
public void matchesLostTest() {
   assertEquals(matchesLost,universityfootballClub.getMatchesLost());
}

@Test
public void matchesDrawnTest() {
   assertEquals(matchesDrawn,universityfootballClub.getMatchesDrawn());
}

@Test
public void goalsScoredTest() {
   assertEquals(goalsScored,universityfootballClub.getGoalsScored());
}

@Test
public void goalsReceivedTest() {
   assertEquals(goalsReceived,universityfootballClub.getGoalsReceived());      }
```

```
    @Test
    public void pointsScoredTest() {
        assertEquals(pointsScored,universityfootballClub.getPointsScored());
    }

}
```

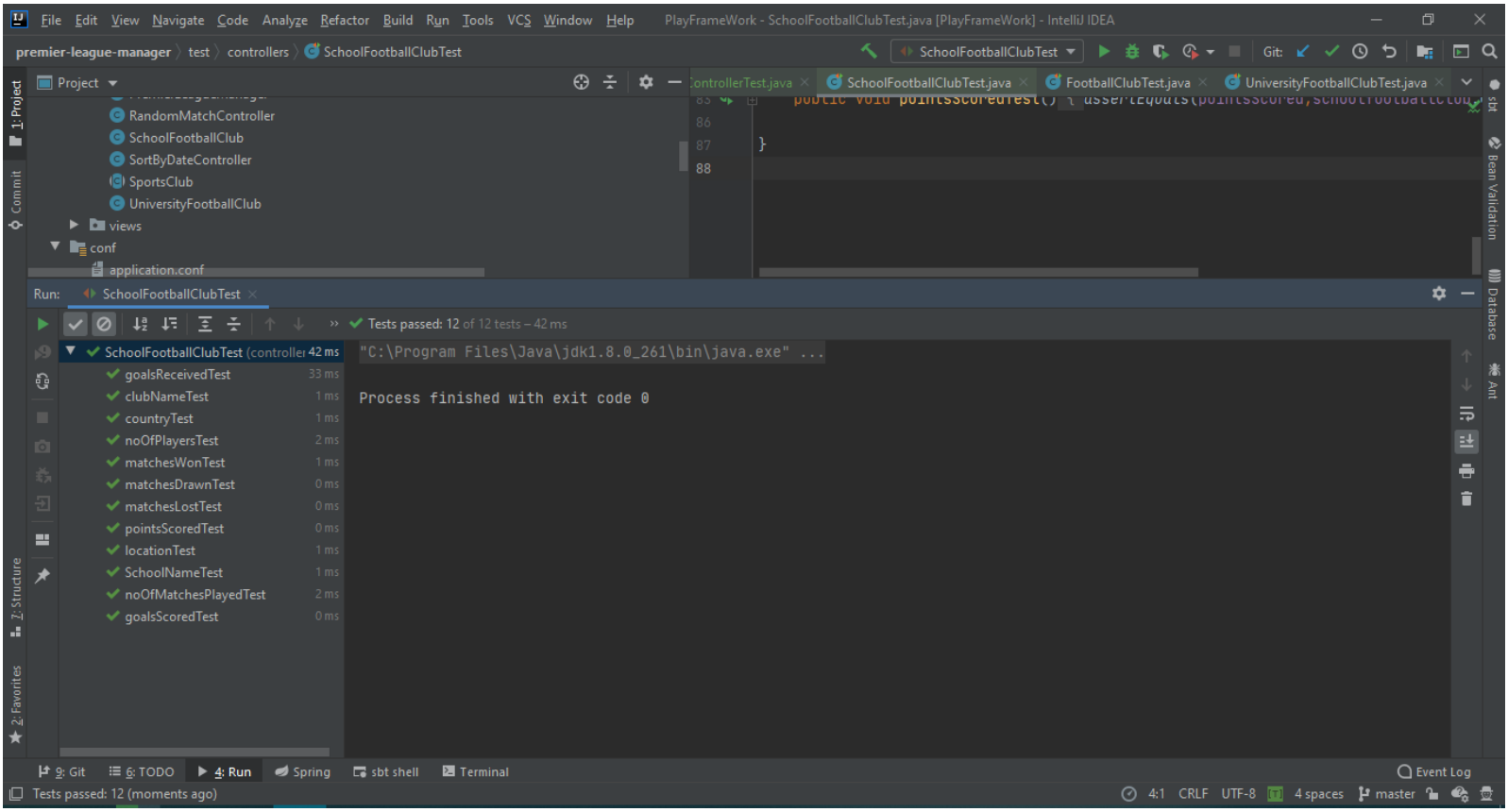


SchoolFootballClubTest

```
package controllers;


import org.junit.Test;


import static org.junit.Assert.assertEquals;


public class SchoolFootballClubTest {
   String schoolName="Vidura College";
   int noOfPlayers=15;//number of players in the school football club
   String clubName="Manchester United";
   String country="England";
   String location="Manchester";
   int noOfMatchesPlayed=8;
   int matchesWon=5;
   int matchesLost=2;
   int matchesDrawn=1;
   int goalsScored=21;
   int goalsReceived=13;
   int pointsScored=45;


   FootballClub schoolfootballClub=new
   SchoolFootballClub(schoolName,noOfPlayers,clubName,country,location,noOfMatchesPlayed,
   matchesWon,matchesLost,matchesDrawn,goalsScored,goalsReceived,pointsScored);


   @Test
   public void SchoolNameTest() {
      assertEquals(schoolName,((SchoolFootballClub) schoolfootballClub).getSchoolName());
```

```
}

@Test
public void noOfPlayersTest() {
   assertEquals(noOfPlayers,((SchoolFootballClub) schoolfootballClub).getNoOfPlayers());

}

@Test
public void clubNameTest() {
   assertEquals(clubName,schoolfootballClub.getClubName());
}

@Test
public void countryTest() {
   assertEquals(country,schoolfootballClub.getCountry());
}

@Test
public void locationTest() {
   assertEquals(location,schoolfootballClub.getLocation());
}

@Test
public void noOfMatchesPlayedTest() {
   assertEquals(noOfMatchesPlayed,schoolfootballClub.getNoOfMatchesPlayed());
}

@Test
```

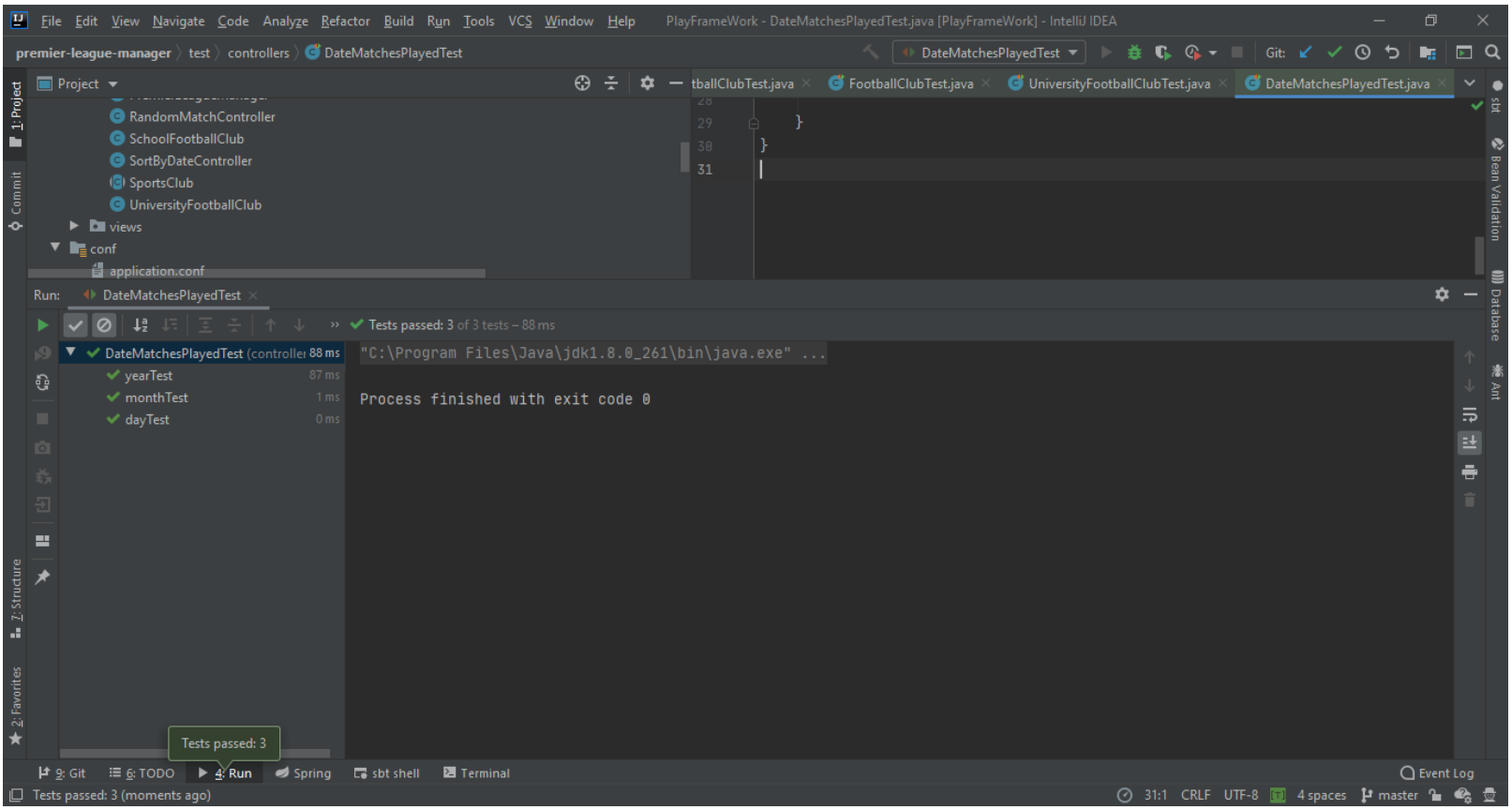
```
    public void matchesWonTest() {
        assertEquals(matchesWon, schoolfootballClub.getMatchesWon());
    }

    @Test
    public void matchesLostTest() {
        assertEquals(matchesLost,schoolfootballClub.getMatchesLost());
    }

    @Test
    public void matchesDrawnTest() {
        assertEquals(matchesDrawn,schoolfootballClub.getMatchesDrawn());
    }

    @Test
    public void goalsScoredTest() {
        assertEquals(goalsScored,schoolfootballClub.getGoalsScored());
    }

    @Test
    public void goalsReceivedTest() {
        assertEquals(goalsReceived,schoolfootballClub.getGoalsReceived());
    }

    @Test
    public void pointsScoredTest() {
        assertEquals(pointsScored,schoolfootballClub.getPointsScored());
    }
}
```


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
PlayFrameWork - SchoolFootballClubTest.java [PlayFrameWork] - IntelliJ IDEA
premier-league-manager test controllers SchoolFootballClubTest
SchoolFootballClubTest
Git:
Project
RandomMatchController
SchoolFootballClub
SortByDateController
SportsClub
UniversityFootballClub
views
conf
application.conf
SchoolFootballClubTest.java
FootballClubTest.java
UniversityFootballClubTest.java
86
87 }
88
Run: SchoolFootballClubTest
Tests passed: 12 of 12 tests – 42 ms
SchoolFootballClubTest (controller 42 ms
goalsReceivedTest 33 ms
clubNameTest 1 ms
countryTest 1 ms
noOfPlayersTest 2 ms
matchesWonTest 1 ms
matchesDrawnTest 0 ms
matchesLostTest 0 ms
pointsScoredTest 0 ms
locationTest 1 ms
SchoolNameTest 1 ms
noOfMatchesPlayedTest 2 ms
goalsScoredTest 0 ms
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ...
Process finished with exit code 0
9: Git
6: TODO
4: Run
Spring
sbt shell
Terminal
Event Log
Tests passed: 12 (moments ago)
4:1 CRLF UTF-8 4 spaces master

DateMatchesPlayedTest

```
package controllers;

import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class DateMatchesPlayedTest {
    int day=5;
    int month=3;
    int year=2020;

    DateMatchesPlayed dateMatchesPlayed=new DateMatchesPlayed(day,month,year);


    @Test
    public void dayTest() {
        assertEquals(day,dateMatchesPlayed.getDay());
    }

    @Test
    public void monthTest() {
        assertEquals(month,dateMatchesPlayed.getMonth());

    }
```

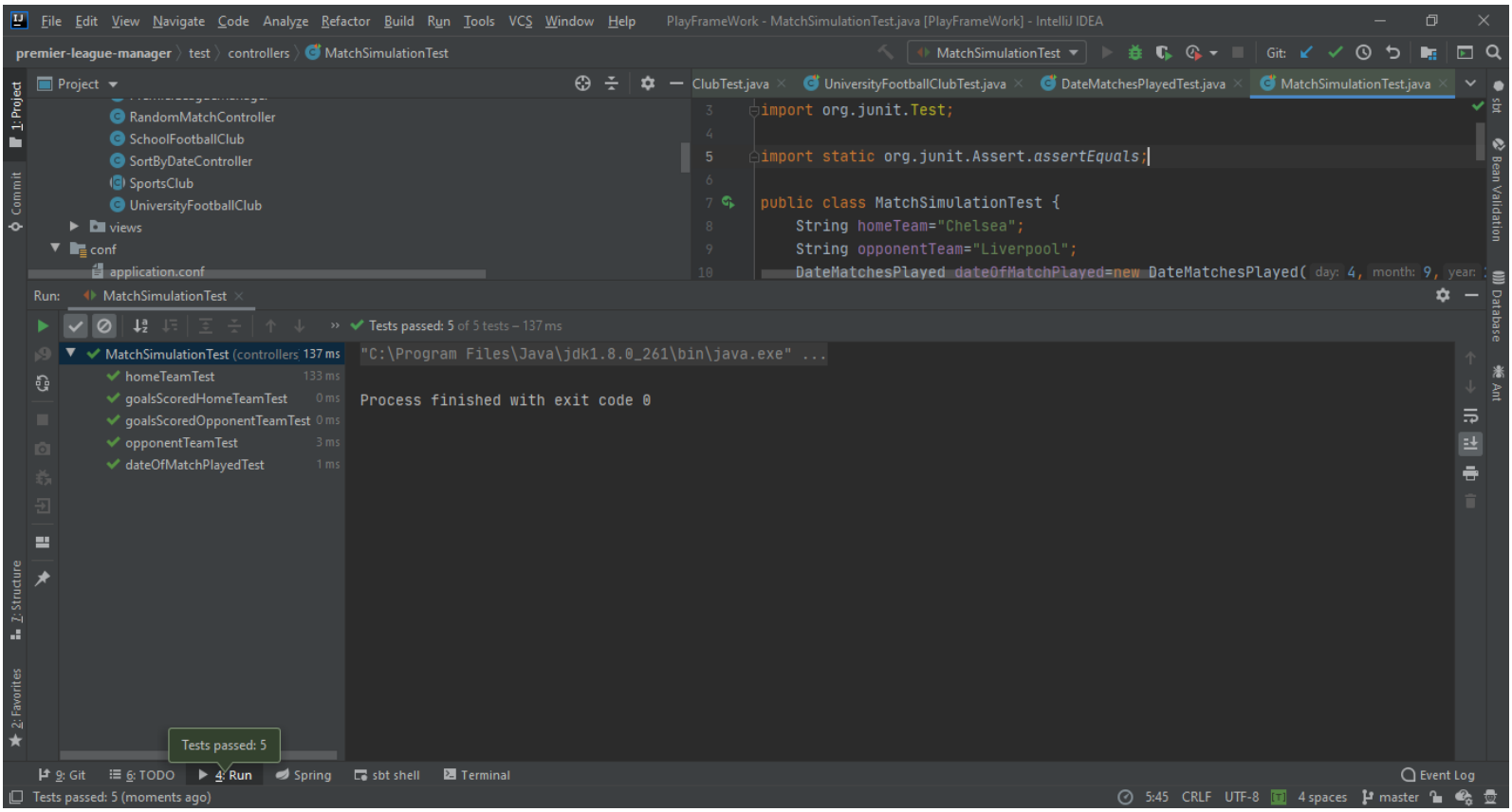
```
    @Test
    public void yearTest() {
        assertEquals(year,dateMatchesPlayed.getYear());

    }
}
```



MatchSimulationTest

```
package controllers;

import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class MatchSimulationTest {
    String homeTeam="Chelsea";
    String opponentTeam="Liverpool";
    DateMatchesPlayed dateOfMatchPlayed=new DateMatchesPlayed(4,9,2020);
    int goalsScoredHomeTeam=7;
    int goalsScoredOpponentTeam=2;

    MatchSimulation matchSimulation=new
MatchSimulation(homeTeam,opponentTeam,dateOfMatchPlayed,goalsScoredHomeTeam,goals
ScoredOpponentTeam);

    @Test
    public void homeTeamTest() {
        assertEquals(homeTeam,matchSimulation.getHomeTeam());
    }

    @Test
    public void opponentTeamTest() {
        assertEquals(opponentTeam,matchSimulation.getOpponentTeam());
    }
```

```
    @Test
    public void dateOfMatchPlayedTest() {
        assertEquals(dateOfMatchPlayed,matchSimulation.getDateOfMatchPlayed());

    }

    @Test
    public void goalsScoredHomeTeamTest() {
        assertEquals(goalsScoredHomeTeam,matchSimulation.getGoalsScoredHomeTeam());
    }

    @Test
    public void goalsScoredOpponentTeamTest() {

assertEquals(goalsScoredOpponentTeam,matchSimulation.getGoalsScoredOpponentTeam());

    }
}
```

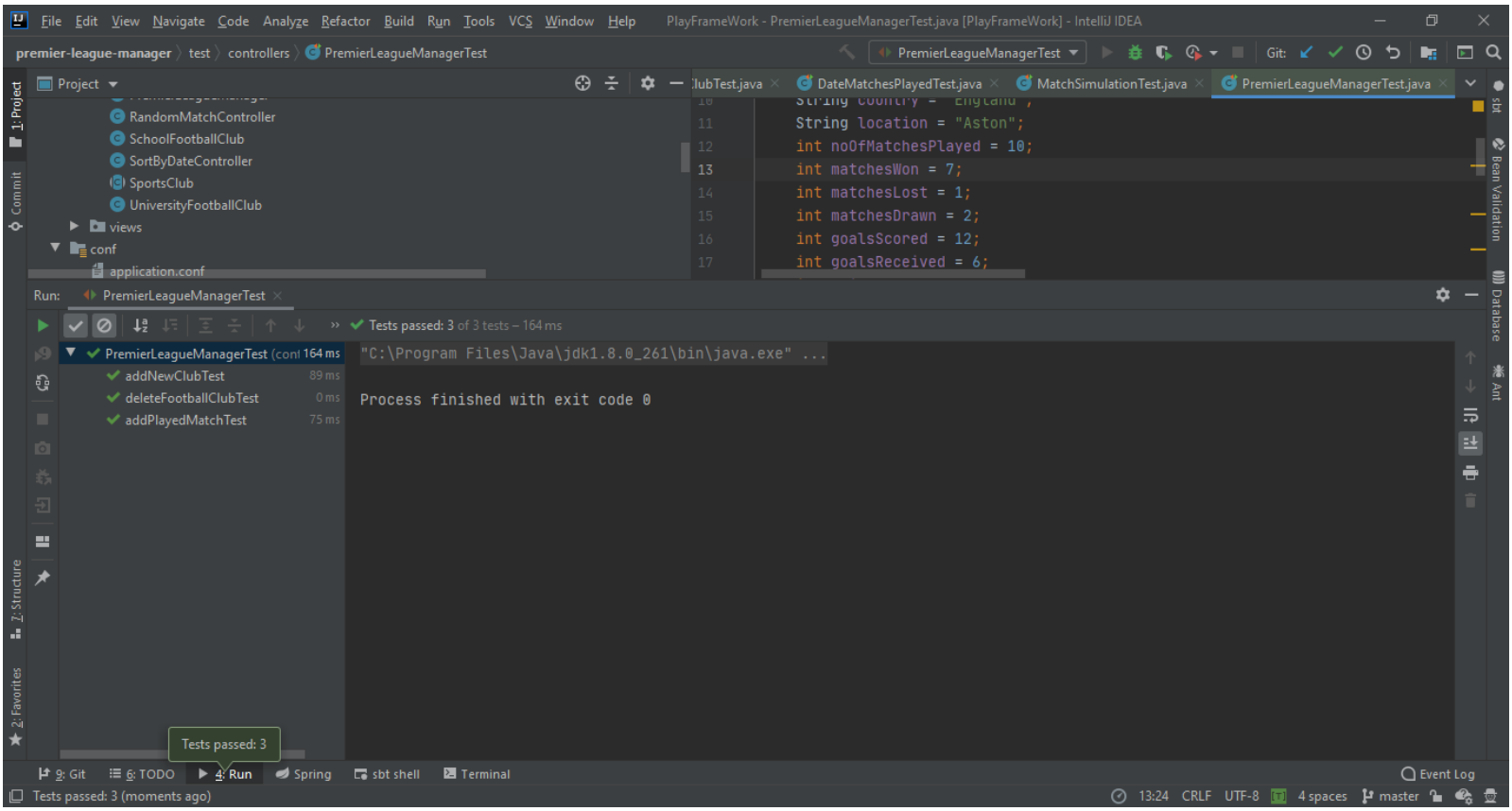

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
PlayFrameWork - MatchSimulationTest.java [PlayFrameWork] - IntelliJ IDEA
premier-league-manager test controllers MatchSimulationTest
Project
RandomMatchController
SchoolFootballClub
SortByDateController
SportsClub
UniversityFootballClub
views
conf
application.conf
ClubTest.java
UniversityFootballClubTest.java
DateMatchesPlayedTest.java
MatchSimulationTest.java
import org.junit.Test;
import static org.junit.Assert.assertEquals;
public class MatchSimulationTest {
String homeTeam="Chelsea";
String opponentTeam="Liverpool";
DateMatchesPlayed dateOfMatchPlayed=new DateMatchesPlayed(day: 4, month: 9, year:
Run: MatchSimulationTest
Tests passed: 5 of 5 tests – 137 ms
MatchSimulationTest (controllers 137 ms
homeTeamTest 133 ms
goalsScoredHomeTeamTest 0 ms
goalsScoredOpponentTeamTest 0 ms
opponentTeamTest 3 ms
dateOfMatchPlayedTest 1 ms
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ...
Process finished with exit code 0
Tests passed: 5
9: Git
6: TODO
4: Run
Spring
sbt shell
Terminal
Event Log
Tests passed: 5 (moments ago)
5:45 CRLF UTF-8 4 spaces master

PremierLeagueManagerTest

```
package controllers;


import org.junit.Test;
import java.util.ArrayList;
import java.util.List;



public class PremierLeagueManagerTest {
    String clubName = "Aston Villa";
    String country = "England";
    String location = "Aston";
    int noOfMatchesPlayed = 10;
    int matchesWon = 7;
    int matchesLost = 1;
    int matchesDrawn = 2;
    int goalsScored = 12;
    int goalsReceived = 6;
    int pointsScored = 34;


    LeagueManager premierLeagueManager = new PremierLeagueManager();


    @Test
    public void addNewClubTest() {


        FootballClub footballClubs = new FootballClub(clubName, country, location,
        noOfMatchesPlayed, matchesWon,  matchesLost, matchesDrawn, goalsScored,
        goalsReceived, pointsScored);
```

```
    List<FootballClub> footballClubsList = new ArrayList<>();


    assertArrayEquals(footballClubsList.add(footballClubs));
}

@Test
public void deleteFootballClubTest() {


    FootballClub footballClubs = new FootballClub(clubName, country, location,
    noOfMatchesPlayed, matchesWon, matchesLost, matchesDrawn, goalsScored,
    goalsReceived, pointsScored);
    List<FootballClub> footballClubsList = new ArrayList<>();


    assertArrayEquals(footballClubsList.remove(footballClubs));
}


@Test
public void addPlayedMatchTest() {


    MatchSimulation matchSimulation = new MatchSimulation("Manchester United",
    "Chelsea", new DateMatchesPlayed(2, 4, 2020), 14, 11);
    List<MatchSimulation> playedMatchSimulation = new ArrayList<>();


    assertArrayEquals(playedMatchSimulation.add(matchSimulation));
}
private void assertArrayEquals ( boolean add){
}
```

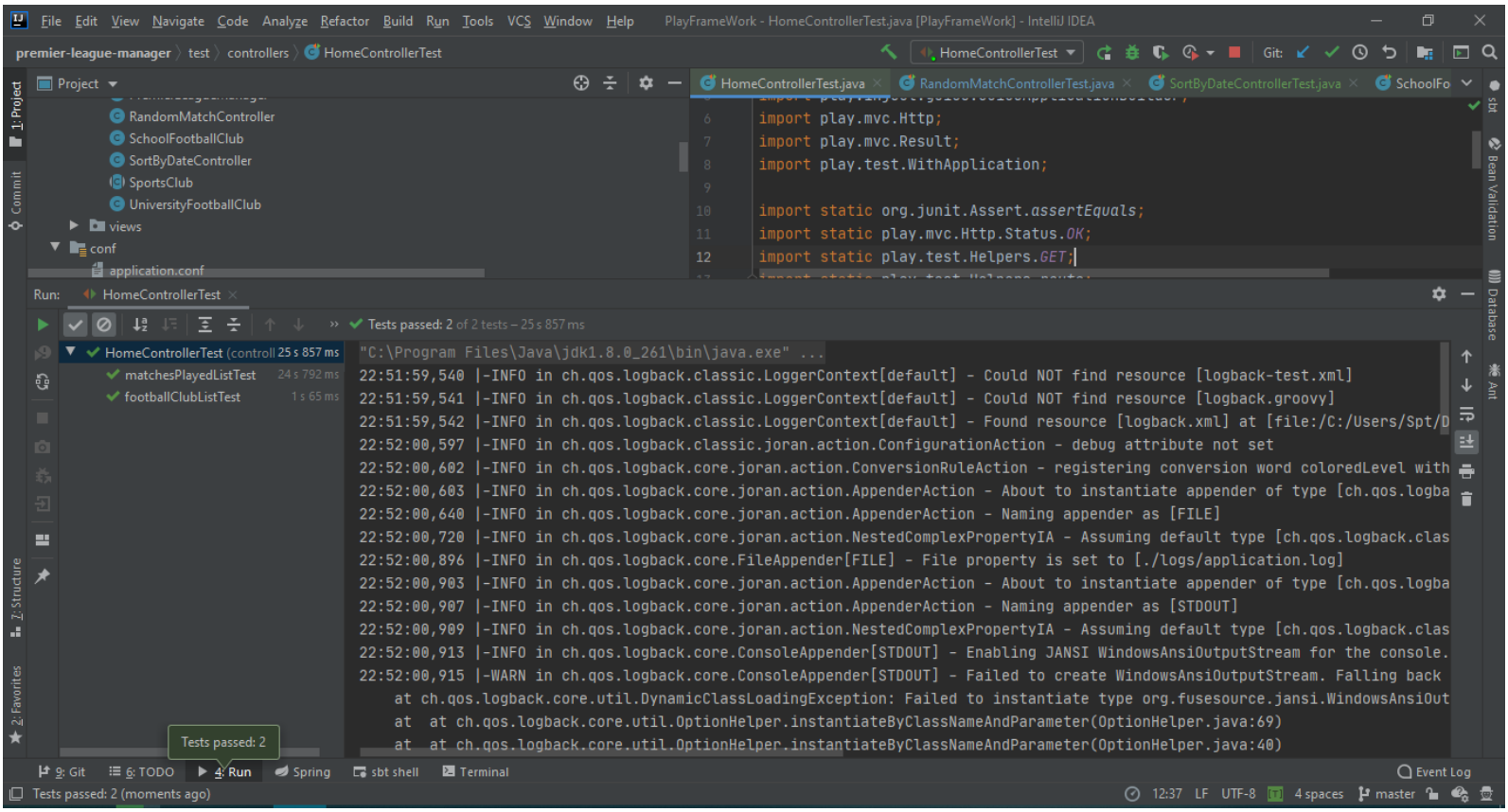
}


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
PlayFrameWork - PremierLeagueManagerTest.java [PlayFrameWork] - IntelliJ IDEA
premier-league-manager test controllers PremierLeagueManagerTest
Project
RandomMatchController
SchoolFootballClub
SortByDateController
SportsClub
UniversityFootballClub
views
conf
application.conf
DateMatchesPlayedTest.java
MatchSimulationTest.java
PremierLeagueManagerTest.java
String location = "Aston";
int noOfMatchesPlayed = 10;
int matchesWon = 7;
int matchesLost = 1;
int matchesDrawn = 2;
int goalsScored = 12;
int goalsReceived = 6;
Run: PremierLeagueManagerTest
Tests passed: 3 of 3 tests – 164 ms
PremierLeagueManagerTest 164 ms
addNewClubTest 89 ms
deleteFootballClubTest 0 ms
addPlayedMatchTest 75 ms
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ...
Process finished with exit code 0
Tests passed: 3
9: Git
6: TODO
4: Run
Spring
sbt shell
Terminal
Event Log
Tests passed: 3 (moments ago)
13:24 CRLF UTF-8 4 spaces master

HomeControllerTest

```
package controllers;

import org.junit.Test;
import play.Application;
import play.inject.guice.GuiceApplicationBuilder;
import play.mvc.Http;
import play.mvc.Result;
import play.test.WithApplication;

import static org.junit.Assert.assertEquals;
import static play.mvc.Http.Status.OK;
import static play.test.Helpers.GET;
import static play.test.Helpers.route;

public class HomeControllerTest extends WithApplication {



    @Override
    protected Application provideApplication() {
        return new GuiceApplicationBuilder().build();
    }

    @Test
    public void footballClubListTest() {
        Http.RequestBuilder request = new Http.RequestBuilder()
```

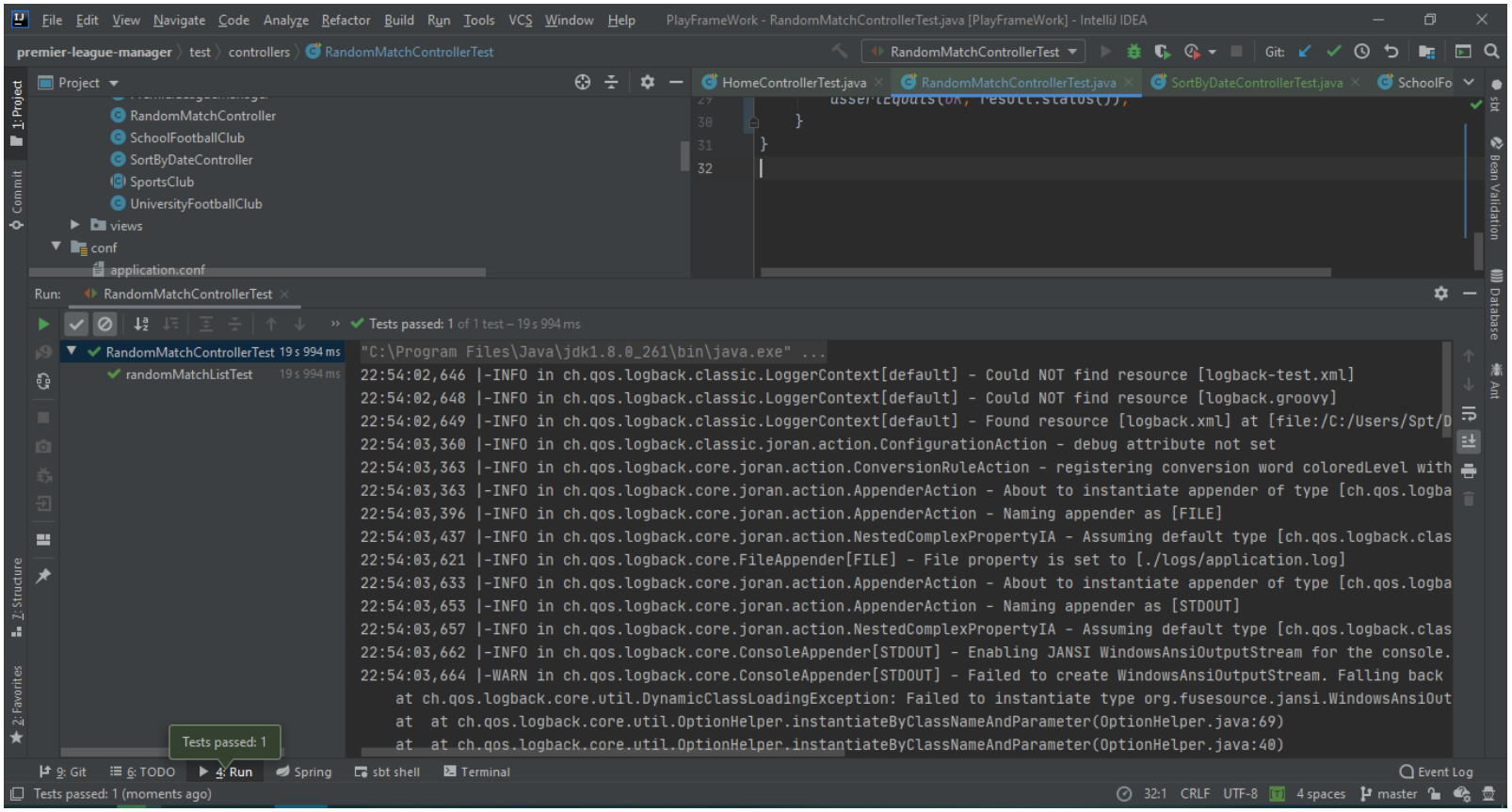
```
            .method(GET)
            .uri("/");

    Result result = route(app, request);
    assertEquals(OK, result.status());
  }

  @Test
  public void matchesPlayedListTest() {
    Http.RequestBuilder request = new Http.RequestBuilder()
            .method(GET)
            .uri("/matchesPlayed");

    Result result = route(app, request);
    assertEquals(OK, result.status());
  }

}
```


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
PlayFrameWork - HomeControllerTest.java [PlayFrameWork] - IntelliJ IDEA
premier-league-manager test controllers HomeControllerTest
HomeControllerTest
Git:
Project
RandomMatchController
SchoolFootballClub
SortByDateController
SportsClub
UniversityFootballClub
views
conf
application.conf
HomeControllerTest.java
RandomMatchControllerTest.java
SortByDateControllerTest.java
import play.mvc.Http;
import play.mvc.Result;
import play.test.WithApplication;
import static org.junit.Assert.assertEquals;
import static play.mvc.Http.Status.OK;
import static play.test.Helpers.GET;
Run: HomeControllerTest
Tests passed: 2 of 2 tests – 25 s 857 ms
HomeControllerTest (controll 25 s 857 ms
matchesPlayedListTest 24 s 792 ms
footballClubListTest 1 s 65 ms
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ...
22:51:59,540 |-INFO in ch.qos.logback.classic.LoggerContext[default] - Could NOT find resource [logback-test.xml]
22:51:59,541 |-INFO in ch.qos.logback.classic.LoggerContext[default] - Could NOT find resource [logback.groovy]
22:51:59,542 |-INFO in ch.qos.logback.classic.LoggerContext[default] - Found resource [logback.xml] at [file:/C:/Users/Spt/D
22:52:00,597 |-INFO in ch.qos.logback.classic.joran.action.ConfigurationAction - debug attribute not set
22:52:00,602 |-INFO in ch.qos.logback.core.joran.action.ConversionRuleAction - registering conversion word coloredLevel with
22:52:00,603 |-INFO in ch.qos.logback.core.joran.action.AppenderAction - About to instantiate appender of type [ch.qos.logba
22:52:00,640 |-INFO in ch.qos.logback.core.joran.action.AppenderAction - Naming appender as [FILE]
22:52:00,720 |-INFO in ch.qos.logback.core.joran.action.NestedComplexPropertyIA - Assuming default type [ch.qos.logback.clas
22:52:00,896 |-INFO in ch.qos.logback.core.FileAppender[FILE] - File property is set to [./logs/application.log]
22:52:00,903 |-INFO in ch.qos.logback.core.joran.action.AppenderAction - About to instantiate appender of type [ch.qos.logba
22:52:00,907 |-INFO in ch.qos.logback.core.joran.action.AppenderAction - Naming appender as [STDOUT]
22:52:00,909 |-INFO in ch.qos.logback.core.joran.action.NestedComplexPropertyIA - Assuming default type [ch.qos.logback.clas
22:52:00,913 |-INFO in ch.qos.logback.core.ConsoleAppender[STDOUT] - Enabling JANSI WindowsAnsiOutputStream for the console.
22:52:00,915 |-WARN in ch.qos.logback.core.ConsoleAppender[STDOUT] - Failed to create WindowsAnsiOutputStream. Falling back
at ch.qos.logback.core.util.DynamicClassLoadingException: Failed to instantiate type org.fusesource.jansi.WindowsAnsiOut
at at ch.qos.logback.core.util.OptionHelper.instantiateByClassNameAndParameter(OptionHelper.java:69)
at at ch.qos.logback.core.util.OptionHelper.instantiateByClassNameAndParameter(OptionHelper.java:40)
Tests passed: 2
9: Git
6: TODO
4: Run
Spring
sbt shell
Terminal
Event Log
Tests passed: 2 (moments ago)
12:37 LF UTF-8 4 spaces master

RandomMatchControllerTest

```
package controllers;

import org.junit.Test;
import play.Application;
import play.inject.guice.GuiceApplicationBuilder;
import play.mvc.Http;
import play.mvc.Result;
import play.test.WithApplication;

import static org.junit.Assert.assertEquals;
import static play.mvc.Http.Status.OK;
import static play.test.Helpers.GET;
import static play.test.Helpers.route;

public class RandomMatchControllerTest extends WithApplication {

    @Override
    protected Application provideApplication() {
        return new GuiceApplicationBuilder().build();
    }

    @Test
    public void randomMatchListTest() {
        Http.RequestBuilder request = new Http.RequestBuilder()
                .method(GET)
                .uri("/randomMatch");
```

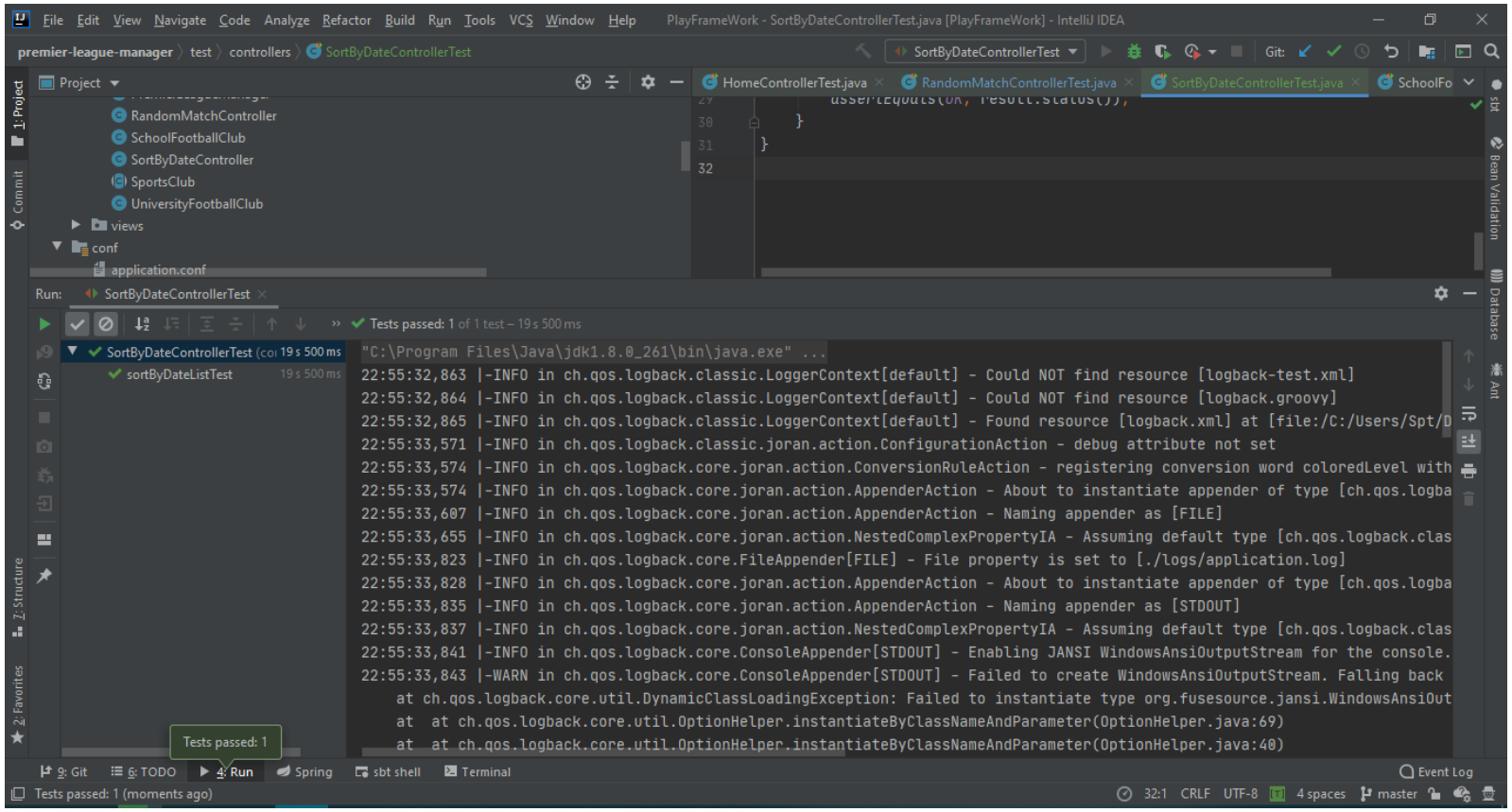
```
        Result result = route(app, request);
        assertEquals(OK, result.status());
    }
}
```



SortByDateControllerTest

```
package controllers;

import org.junit.Test;
import play.Application;
import play.inject.guice.GuiceApplicationBuilder;
import play.mvc.Http;
import play.mvc.Result;
import play.test.WithApplication;

import static org.junit.Assert.assertEquals;
import static play.mvc.Http.Status.OK;
import static play.test.Helpers.GET;
import static play.test.Helpers.route;

public class SortByDateControllerTest extends WithApplication {

    @Override
    protected Application provideApplication() {
        return new GuiceApplicationBuilder().build();
    }

    @Test
    public void sortByDateListTest() {
        Http.RequestBuilder request = new Http.RequestBuilder()
                .method(GET)
                .uri("/sortByDate");
```

```
        Result result = route(app, request);
        assertEquals(OK, result.status());
    }
}
```



Conslusion

By doing this coursework, it helped to gain a vast knowledge about Object oriented programming concepts, Angular for the front-end and playframework for the backend while using REST API. And it helps to learn how to call an API from Angular. And this coursework gave a knowledge how a premier league championship works.

References

- Stackblitz.com. 2020. *Angular-Clock-1-Q2tuyq - Stackblitz*. [online] Available at: <https://stackblitz.com/edit/angular-clock-1-q2tuyq?file=src%2Fapp%2Fapp.component.html> [Accessed 31 December 2020].

- Youtu.be. 2020. [online] Available at: <https://youtu.be/rdLJNGZvlAA> [Accessed 31 December 2020].