

**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.

# 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 javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.layout.Background;

import javafx.scene.layout.BackgroundFill;

import javafx.scene.layout.CornerRadii;

import javafx.scene.layout.Pane;

import javafx.scene.paint.Color;

import javafx.scene.text.Font;

import javafx.scene.text.FontPosture;

import javafx.scene.text.FontWeight;

import javafx.stage.Stage;

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.awt.\*;

import java.io.File;

import java.io.IOException;

import java.io.Serializable;

import java.net.URI;

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 {

// premierLeagueManager.displayPremierLeagueTableGUI();//calling the GUI table of the premier league

Desktop.getDesktop().browse(new URI("http://localhost:9000"));

Desktop.getDesktop().browse(new URI("http://localhost:4200"));

}

}

**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;

}

}

**RandomMatchAngularController**

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.getClubName()))) {

//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(),randomElementOpponentTeam.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;

}

}

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 also it helps to learn how to call an API from Angular. And this coursework gave a knowledge how a premier league championship works.