

**Informatics Institute of Technology**

**Department of Computing**

BSc in Computer Science

**Module: 5COSC007C Object Oriented Programming**

**Module Leader: Mr. Guhanathan Poravi**

**Tutorial Group** : SE/CS Group F

**Student IIT ID** : 20191191

**Student UoW ID**  : W17903352

**Student First Name** : Mahfoos

**Student Surname** : Ahamed

**Individual Course Work**

Table of Contents

[Design 3](#_Toc60623862)

[Class Diagram 3](#_Toc60623863)

[Use Case Diagram 4](#_Toc60623864)

[Java Codes 4](#_Toc60623865)

[Question – 1 4](#_Toc60623866)

[SportsClub.java 5](#_Toc60623867)

[FootballClub.java 6](#_Toc60623868)

[UniversityFootballClub.java 12](#_Toc60623869)

[SchoolFootballClub.java 12](#_Toc60623870)

[Question – 2 13](#_Toc60623871)

[LeagueManager.java 13](#_Toc60623872)

[PremierLeagueManager.java 13](#_Toc60623873)

[ConsoleApplication.java 23](#_Toc60623874)

[Question – 3 32](#_Toc60623875)

[Angular Code 32](#_Toc60623876)

[Points-table.component.html 32](#_Toc60623877)

[Points-table.component.ts 33](#_Toc60623878)

[points-table.service.ts 34](#_Toc60623879)

[Random-match.component.html 35](#_Toc60623880)

[Random-match.component.ts 36](#_Toc60623881)

[Random-match.service.ts 37](#_Toc60623882)

[Match-table.component.html 38](#_Toc60623883)

[Match-table.component.ts 39](#_Toc60623884)

[Match-table.service.ts 40](#_Toc60623885)

[Play Framework Codes 41](#_Toc60623886)

[Controllers 41](#_Toc60623887)

[MatchTableController.java 41](#_Toc60623888)

[PointsTableController.java 42](#_Toc60623889)

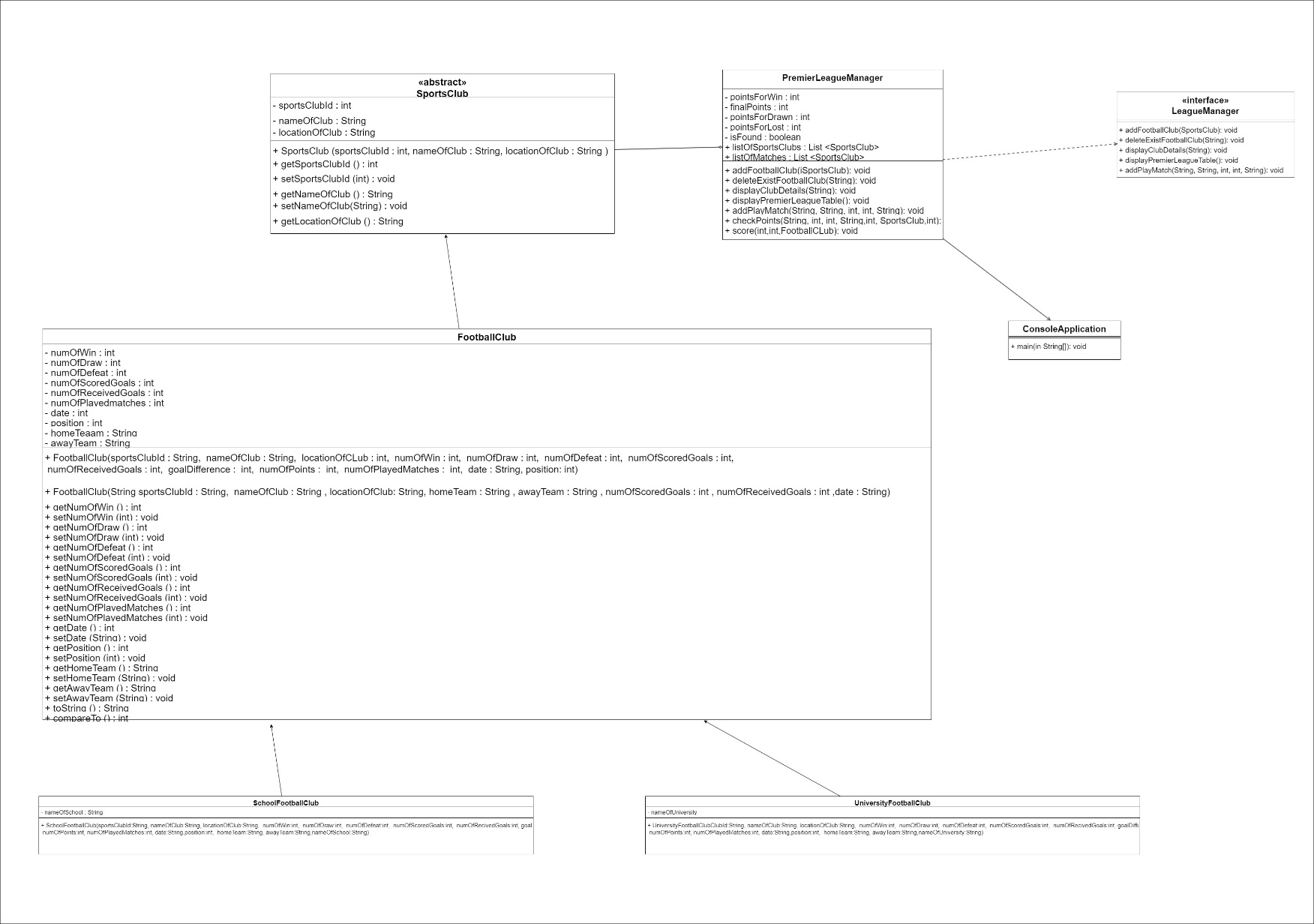
[Services 43](#_Toc60623890)

[DateService.java 43](#_Toc60623891)

[PointsTableService.java 44](#_Toc60623892)

# Design

## Class Diagram



## Use Case Diagram

* Use Case for Command line Application



# Java Codes

## Question – 1

### 

### SportsClub.java

public abstract class SportsClub implements Serializable {

private String sportsClubId;

private String nameOfClub;

private String locationOfClub;

public SportsClub(String sportsClubId,String nameOfClub, String locationOfClub) {

this.sportsClubId = sportsClubId;

this.nameOfClub = nameOfClub;

this.locationOfClub = locationOfClub;

}

public String getSportsClubId() {

return sportsClubId;

}

public void setSportsClubId(String sportsClubId) {

this.sportsClubId = sportsClubId;

}

public String getNameOfClub() {

return nameOfClub;

}

public void setNameOfClub(String nameOfClub) {

this.nameOfClub = nameOfClub;

}

public String getLocationOfClub() {

return locationOfClub;

}

public void setLocationOfClub(String locationOfClub) {

this.locationOfClub = this.locationOfClub;

}

@Override

public String toString() {

return "SportsClub{" +

"nameOfClub='" + nameOfClub + '\'' +", locationOfClub='" + locationOfClub + '\'' +’}’;}}

### FootballClub.java

public class FootballClub extends SportsClub implements Comparable<FootballClub> {

private int numOfWin;

private int numOfDraw;

private int numOfDefeat;

private int numOfScoredGoals;

private int numOfReceivedGoals;

private int goalDifference;

private int numOfPoints;

private int numOfPlayedMatches;

private String date;

private int position;

private String homeTeam;

private String awayTeam;

public FootballClub(String sportsClubId,String nameOfClub, String locationOfCLub, int numOfWin, int numOfDraw, int numOfDefeat, int numOfScoredGoals, int numOfReceivedGoals, int goalDifference, int numOfPoints, int numOfPlayedMatches, String date,int position) {

super(sportsClubId,nameOfClub, locationOfCLub);

this.numOfWin = numOfWin;

this.numOfDraw = numOfDraw;

this.numOfDefeat = numOfDefeat;

this.numOfScoredGoals = numOfScoredGoals;

this.numOfReceivedGoals = numOfReceivedGoals;

this.goalDifference = goalDifference;

this.numOfPoints = numOfPoints;

this.numOfPlayedMatches = numOfPlayedMatches;

this.date = date;

this.position = position;

}

public FootballClub(String sportsClubId,String nameOfClub, String locationOfClub, String homeTeam, String awayTeam, int numOfScoredGoals, int numOfReceivedGoals, String date) {

super(sportsClubId,nameOfClub, locationOfClub);

this.homeTeam = homeTeam;

this.awayTeam = awayTeam;

this.numOfScoredGoals = numOfScoredGoals;

this.numOfReceivedGoals = numOfReceivedGoals;

this.date = date;

}

public int getNumOfWin() {

return numOfWin;

}

public void setNumOfWin(int numOfWin) {

this.numOfWin = numOfWin;

}

public int getNumOfDraw() {

return numOfDraw;

}

public void setNumOfDraw(int numOfDraw) {

this.numOfDraw = numOfDraw;

}

public int getNumOfDefeat() {

return numOfDefeat;

}

public void setNumOfDefeat(int numOfDefeat) {

this.numOfDefeat = numOfDefeat;

}

public int getNumOfScoredGoals() {

return numOfScoredGoals;

}

public void setNumOfScoredGoals(int numOfScoredGoals) {

this.numOfScoredGoals = numOfScoredGoals;

}

public int getNumOfReceivedGoals() {

return numOfReceivedGoals;

}

public void setNumOfReceivedGoals(int numOfReceivedGoals) {

this.numOfReceivedGoals = numOfReceivedGoals;

}

public int getGoalDifference() {

return goalDifference;

}

public void setGoalDifference(int goalDifference) {

this.goalDifference = goalDifference;

}

public int getNumOfPoints() {

return numOfPoints;

}

public void setNumOfPoints(int numOfPoints) {

this.numOfPoints = numOfPoints;

}

public int getNumOfPlayedMatches() {

return numOfPlayedMatches;

}

public void setNumOfPlayedMatches(int numOfPlayedMatches) {

this.numOfPlayedMatches = numOfPlayedMatches;

}

public String getDate() {

return date;

}

public void setDate(String date) {

this.date = date;

}

public int getPosition() {

return position;

}

public void setPosition(int position) {

this.position = position;

}

public String getHomeTeam() {

return homeTeam;

}

public void setHomeTeam(String homeTeam) {

this.homeTeam = homeTeam;

}

public String getAwayTeam() {

return awayTeam;

}

public void setAwayTeam(String awayTeam) {

this.awayTeam = awayTeam;

}

@Override

public int compareTo(FootballClub football) {

return Comparator.comparing(FootballClub::getNumOfPoints)

.thenComparing(FootballClub::getGoalDifference)

.compare(this, football);

}

@Override

public String toString() {

return "FootballClub{" +

"numOfWin=" + numOfWin +

", numOfDraw=" + numOfDraw +

", numOfDefeat=" + numOfDefeat +

", numOfScoredGoals=" + numOfScoredGoals +

", numOfReceivedGoals=" + numOfReceivedGoals +

", goalDifference=" + goalDifference +

", numOfPoints=" + numOfPoints +

", numOfPlayedMatches=" + numOfPlayedMatches +

", date='" + date + '\'' +

", position=" + position +

", homeTeam='" + homeTeam + '\'' +

", awayTeam='" + awayTeam + '\'' +

'}';

}

}

/\*

Reference

# Comparing Multiple field

https://stackoverflow.com/questions/369512/how-to-compare-objects-by-multiple-fields

\*/

### UniversityFootballClub.java

public class UniversityFootballClub extends FootballClub {

private String nameOfUniversity;

public UniversityFootballClub (String sportsClubId,String nameOfClub, String locationOfCLub, int numOfWin, int numOfDraw, int numOfDefeat, int numOfScoredGoals, int numOfReceivedGoals, int goalDifference, int numOfPoints,int numOfPlayedMatches, String date,int position, String nameOfUniversity) {

super (sportsClubId, nameOfClub, locationOfCLub, numOfWin, numOfDraw, numOfDefeat, numOfScoredGoals, numOfPoints,numOfReceivedGoals,goalDifference, numOfPlayedMatches,date,position);

this.nameOfUniversity = nameOfUniversity;

}

}

### SchoolFootballClub.java

public class SchoolFootballClub extends FootballClub{

private String nameOfSchool;

public SchoolFootballClub (String sportsClubId,String nameOfClub, String locationOfCLub, int numOfWin, int numOfDraw, int numOfDefeat, int numOfScoredGoals, int numOfReceivedGoals, int goalDifference, int numOfPoints, int numOfPlayedMatches, String date, int position,String nameOfSchool) {

super (sportsClubId, nameOfClub, locationOfCLub, numOfWin, numOfDraw, numOfDefeat, numOfScoredGoals, numOfReceivedGoals,goalDifference, numOfPoints, numOfPlayedMatches,date,position);

this. nameOfSchool = nameOfSchool;

}

}

## Question – 2

### LeagueManager.java

public interface LeagueManager {

void addFootballClub(SportsClub sportsClub);

void deleteExistFootballClub(String clubId);

void displayClubDetails(String clubId);

void displayPremierLeagueTable();

void addPlayMatch(String homeTeam,String awayTeam, int homeTeamScore,int awayTeamScore, String date);

}

### PremierLeagueManager.java

public class PremierLeagueManager implements LeagueManager {

private static int pointsForWin;

private static int finalPoints;

private static int pointsForDrawn;

private static int pointsForLost;

private static boolean isFound;

public static List<SportsClub> listOfSportsClubs = new ArrayList<>(); // Array list For Sports CLub

public static List<SportsClub> listOfMatches = new ArrayList<>(); // Arraylist for Matches

@Override

public void addFootballClub(SportsClub sportsClub) {

isFound = false;

for (SportsClub sportsClub1 : listOfSportsClubs) {

if (sportsClub1.getNameOfClub().equalsIgnoreCase(sportsClub.getNameOfClub())) { // Check Whether Club is exist

isFound = true;

System.out.println("\nThis Club Already added in Premier League");

break;

}

}

if (!isFound) {

listOfSportsClubs.add(sportsClub); // add the Football Club into ListOF Sports Club

System.out.println("\nSuccessfully added the Football Club");

}

}

@Override

public void deleteExistFootballClub(String clubId) {

isFound = false;

for (SportsClub sportsClub : listOfSportsClubs) {

if (sportsClub.getSportsClubId().equalsIgnoreCase(clubId)) {

isFound = true;

listOfSportsClubs.remove(sportsClub);

System.out.println("\nSportsClub with the " + clubId + " Successfully removed ");

break;

}

}

if (!isFound) {

System.out.println("\nNot found the Entered Id");

}

}

@Override

public void displayClubDetails(String clubId) {

isFound = false;

if (listOfSportsClubs.isEmpty()) { // Check Whether ListOfSportsClub is Empty

System.out.println("No Clubs are Added in Premier League ");

} else {

for (SportsClub sportsClub : listOfSportsClubs) {

if (sportsClub.getSportsClubId().equalsIgnoreCase(clubId)) { // check the football club in the added list

System.out.println("\n:::::::::::::::::::::: Created Football Clubs ::::::::::::::::::::::::::::::");

System.out.println("\n1: Id Of the Club : " + sportsClub.getSportsClubId());

System.out.println("\n1: Name Of the Club : " + sportsClub.getNameOfClub());

System.out.println("\n2: Location of the Club : " + sportsClub.getLocationOfClub());

System.out.println("\n3: Number of Played Matches : " + ((FootballClub) sportsClub).getNumOfPlayedMatches());

System.out.println("\n4: Number of Won Matches : " + ((FootballClub) sportsClub).getNumOfWin());

System.out.println("\n5: Number of Defeat Matches : " + ((FootballClub) sportsClub).getNumOfDefeat());

System.out.println("\n6: Number of Drawn Matches : " + ((FootballClub) sportsClub).getNumOfDraw());

System.out.println("\n7: Total Score of the Club : " + ((FootballClub) sportsClub).getNumOfScoredGoals());

System.out.println("\n8: Total Received Goal : " + ((FootballClub) sportsClub).getNumOfReceivedGoals());

System.out.println("\n9: Goal Difference : " + ((FootballClub) sportsClub).getGoalDifference());

System.out.println("\n8: Total Points of the Club : " + ((FootballClub) sportsClub).getNumOfPoints());

}

}

if (!isFound) {

System.out.println("\nNot found the Entered Id");

}

}

}

@Override

public void displayPremierLeagueTable() {

listOfSportsClubs.sort(Collections.reverseOrder()); // Sort

System.out.println("\n\t\t\t\t\t\t\t\t\t\t\t:::::::::::::::::::::::::::::::::::::Ranking Table::::::::::::::::::::::::::::::::::::::\n");

String tableAlignment = "| %-15s | %-7s | %-20s | %-15s | %-8s | %-8s | %-8s | %-8s | %-12s | %-15s | %-15s | %-7s | %n";

System.out.format("+-----------------+---------+----------------------+-----------------+----------+----------+----------+----------+--------------+-----------------+-----------------+---------+%n");

System.out.format("| Position | Club Id | Club Name | Club Location | Played | Won | Drawn | Lost | Goals For | Goal Against | Goal Difference | Points |%n");

System.out.format("+-----------------+---------+----------------------+-----------------+----------+----------+----------+----------+--------------+-----------------+-----------------+---------+%n");

for (SportsClub sportsClub : listOfSportsClubs) {

System.out.format(tableAlignment, ((FootballClub) sportsClub).getPosition(), sportsClub.getSportsClubId(), sportsClub.getNameOfClub(), sportsClub.getLocationOfClub(), ((FootballClub) sportsClub).getNumOfPlayedMatches(), ((FootballClub) sportsClub).getNumOfWin(), ((FootballClub) sportsClub).getNumOfDraw(), ((FootballClub) sportsClub).getNumOfDefeat(), ((FootballClub) sportsClub).getNumOfScoredGoals(), ((FootballClub) sportsClub).getNumOfReceivedGoals(),((FootballClub) sportsClub).getGoalDifference(), ((FootballClub) sportsClub).getNumOfPoints());

}System.out.format("+-----------------+---------+----------------------+-----------------+----------+----------+----------+----------+--------------+-----------------+-----------------+---------+%n");

}

@Override

public void addPlayMatch(String homeTeam, String awayTeam, int homeTeamScore, int awayTeamScore, String date) {

int count = 0;

for (SportsClub sportsClub : listOfSportsClubs) {

int difference1 = homeTeamScore + ((FootballClub) sportsClub).getNumOfScoredGoals() - awayTeamScore + ((FootballClub) sportsClub).getNumOfReceivedGoals();

int difference2 = awayTeamScore + ((FootballClub) sportsClub).getNumOfScoredGoals() - homeTeamScore + ((FootballClub) sportsClub).getNumOfReceivedGoals();

if (sportsClub.getNameOfClub().equals(homeTeam)) {

checkPoints(homeTeam, homeTeamScore, awayTeamScore, date, count, sportsClub, difference1);

} else if (sportsClub.getNameOfClub().equals(awayTeam)) {

checkPoints(awayTeam, awayTeamScore, homeTeamScore, date, count, sportsClub, difference2);

}

count++;

}

int position = 0;

listOfSportsClubs.sort(Collections.reverseOrder());

for (SportsClub sportsClub2 : listOfSportsClubs) {

SportsClub sportsClub3 = new FootballClub(sportsClub2.getSportsClubId(), sportsClub2.getNameOfClub(), sportsClub2.getLocationOfClub(), ((FootballClub) sportsClub2).getNumOfWin(), ((FootballClub) sportsClub2).getNumOfDraw(), ((FootballClub) sportsClub2).getNumOfDefeat(), ((FootballClub) sportsClub2).getNumOfScoredGoals(), ((FootballClub) sportsClub2).getNumOfReceivedGoals(),

((FootballClub) sportsClub2).getGoalDifference(), ((FootballClub) sportsClub2).getNumOfPoints(), ((FootballClub) sportsClub2).getNumOfPlayedMatches(), ((FootballClub) sportsClub2).getDate(), position + 1);

listOfSportsClubs.set(position, sportsClub3);

position++;

}

}

private void checkPoints(String homeTeam, int homeTeamScore, int awayTeamScore, String date, int count, SportsClub sportsClub, int difference1) {

score(homeTeamScore, awayTeamScore, ((FootballClub) sportsClub));

SportsClub sportsClub1 = new FootballClub(sportsClub.getSportsClubId(), homeTeam, sportsClub.getLocationOfClub(), pointsForWin, pointsForDrawn, pointsForLost, homeTeamScore + ((FootballClub) sportsClub).getNumOfScoredGoals(), awayTeamScore + ((FootballClub) sportsClub).getNumOfReceivedGoals(),

difference1, finalPoints, ((FootballClub) sportsClub).getNumOfPlayedMatches() + 1, date, ((FootballClub) sportsClub).getPosition());

listOfSportsClubs.set(count, sportsClub1);

}

private void score(int homeTeamScore, int awayTeamScore, FootballClub footballClub) {

pointsForWin = 0;

finalPoints = 0;

pointsForDrawn = 0;

pointsForLost = 0;

if (homeTeamScore > awayTeamScore) {

pointsForWin = footballClub.getNumOfWin() + 1;

finalPoints = footballClub.getNumOfPoints() + 3;

} else if (homeTeamScore == awayTeamScore) {

pointsForDrawn = footballClub.getNumOfDraw() + 1;

finalPoints = footballClub.getNumOfPoints() + 1;

} else {

pointsForLost = footballClub.getNumOfDefeat() + 1;

finalPoints = footballClub.getNumOfPoints();

}

}

}

/\*

Reference

# Printing the Table in Console

\* https://stackoverflow.com/questions/15215326/how-can-i-create-table-using-ascii-in-a-console

\*/

// Write the data to file

public static void saveData() {

try {

FileOutputStream fileOut = new FileOutputStream("premierLeagueData.txt");

ObjectOutputStream objOut = new ObjectOutputStream(fileOut);

objOut.writeObject(PremierLeagueManager.listOfSportsClubs);

objOut.writeObject(PremierLeagueManager.listOfMatches);

System.out.println("Data Saved Successfully");

objOut.flush();

objOut.close();

fileOut.close();

} catch (IOException ioe) {

ioe.printStackTrace();

} }

// Load the Data

public static void loadData() {

try {

FileInputStream fileInput = new FileInputStream("premierLeagueData.txt");

ObjectInputStream objOutput = new ObjectInputStream(fileInput);

PremierLeagueManager.listOfSportsClubs = (ArrayList) objOutput.readObject();

PremierLeagueManager.listOfMatches = (ArrayList) objOutput.readObject();

objOutput.close();

fileInput.close();

} catch (IOException ioException) {

ioException.printStackTrace();

} catch (ClassNotFoundException classNotFound) {

System.out.println("Class not found");

classNotFound.printStackTrace();

}

}

/\*

Reference

# Printing the Table in Console

\* <https://stackoverflow.com/questions/15215326/how-can-i-create-table-using-ascii-in-a-console>

# Serialization & Deserialization

\* https://www.geeksforgeeks.org/serialization-in-java/

\*/

### ConsoleApplication.java

public class ConsoleApplication {

private static boolean decide;

private static String clubId;

private static String clubName;

private static String location;

private static boolean idCheck;

private static boolean clubNameCheck;

private static boolean locationCheck;

private static String homeTeam;

private static String awayTeam;

private static int homeTeamScore;

private static int awayTeamScore;

private static String date;

private static boolean homeTeamCheck;

private static boolean awayTeamCheck;

private static boolean homeTeamScoreCheck;

private static boolean awayTeamScoreCheck;

private static boolean dateCheck;

static LeagueManager manager = new PremierLeagueManager();

static Scanner userInput = new Scanner(System.in);

public static void main(String[] args) {

FileHandleUtil loadData = new FileHandleUtil();

loadData.loadData(); // Call the function for load the file

menu:

do {

try {

userInput = new Scanner(System.in);

displayMenu();

System.out.print("\nEnter the option here : ");

int option = userInput.nextInt();

switch (option) {

case 1:

addFootballClub();

userDecision();

break;

case 2:

deleteClub();

userDecision();

break;

case 3:

DisplayClubDetails();

userDecision();

break;

case 4:

manager.displayPremierLeagueTable();

userDecision();

break;

case 5:

addPlayMatch();

userDecision();

break;

case 6:

Desktop.getDesktop().browse(new URI("http://localhost:4200/points\_table ")); // open Gui From Cli

userDecision();

break;

case 7:

FileHandleUtil saveData = new FileHandleUtil(); // Write to the File

saveData.saveData();

userDecision();

break;

case 8:

System.out.println("\n:::::::::::Thank You For Use the Premier League Console Application::::::::");

decide = false;

break menu;

default:

System.out.println("\nYou Selected an Invalid Option. Please Try Again! ");

userDecision();

}

} catch (Exception e) {

System.out.println("\nPlease Enter Valid input");

decide = true;

}

} while (decide);

}

private static void displayMenu() {

System.out.println("\n:::::::::::::::::::::::::::Welcome to Premier League Football::::::::::::::::::::::::::");

System.out.println("\n1: Create a New Football Club ");

System.out.println("\n2: Delete The existing club from the Premier league");

System.out.println("\n3: Display the Information about Selected Club ");

System.out.println("\n4: Display the Premier League Table");

System.out.println("\n5: Add a played match with its score and its date ");

System.out.println("\n6: Open the GUI Application ");

System.out.println("\n7: Save the Entered Data ");

System.out.println("\n8: Quit the Application\n");

System.out.println("\n::::::::::::::::::::::::::::Choose the option You want:::::::::::::::::::::::::::::::::");

}

private static void addFootballClub() {

SportsClub sportsClub;

System.out.println("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Add the FootballClub \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

do {

System.out.print("\nEnter the Club Id: ");

clubId = userInput.next();

idCheck = clubId.matches("^[a-zA-Z0-9]+$");

if (idCheck) {

break;

} else {

System.out.println("Please Enter the valid id");

}

} while (!idCheck);

do {

System.out.print("\nEnter the Club Name: ");

clubName = userInput.next();

clubNameCheck = clubName.matches("^[a-zA-Z0-9\_ ]\*$");

if (clubNameCheck) {

break;

} else {

System.out.println("Please enter the valid Foot club name");

}

} while (!clubNameCheck);

do {

userInput = new Scanner(System.in);

System.out.print("\nEnter the Location of Club: ");

location = userInput.next();

locationCheck = location.matches("^[a-zA-Z0-9\_ ]\*$");

if (locationCheck) {

break;

} else {

System.out.println("Please valid the valid location name");

}

} while (!locationCheck);

sportsClub = new FootballClub(clubId, clubName, location, 0, 0, 0, 0, 0, 0, 0, 0, "", 0);

manager.addFootballClub(sportsClub);

}

private static void addPlayMatch() {

SportsClub sportsClub;

System.out.println("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Add the Match Between two Clubs \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

do {

System.out.print("\nEnter the Home Club name : ");

homeTeam = userInput.next();

homeTeamCheck = homeTeam.matches("^[a-zA-Z]\*$"); // Check the String using regex

if (homeTeamCheck) {

break;

} else {

System.out.println("Please enter valid Club Name");

}

} while (!homeTeamCheck);

do {

System.out.print("\nEnter the Away Club Name : ");

awayTeam = userInput.next();

awayTeamCheck = awayTeam.matches("^[a-zA-Z]\*$");

if (awayTeamCheck) {

break;

} else {

System.out.println("Please enter Valid Club Name");

}

} while (!awayTeamCheck);

do {

try {

userInput = new Scanner(System.in);

System.out.print("\nEnter the Home Club Score : ");

homeTeamScore = userInput.nextInt();

homeTeamScoreCheck = false;

} catch (Exception e) {

System.out.println("Please enter the valid input");

homeTeamScoreCheck = true;

}

} while (homeTeamScoreCheck);

do {

try {

userInput = new Scanner(System.in);

System.out.print("\nEnter the Away Club Score : ");

awayTeamScore = userInput.nextInt();

awayTeamScoreCheck = false;

} catch (Exception e) {

System.out.println("Please enter the valid input");

awayTeamScoreCheck = true;

}

} while (awayTeamScoreCheck);

do {

System.out.print("\nEnter the Date: ");

date = userInput.next();

dateCheck = date.matches("^(0?[1-9]|[12][0-9]|3[01])-(0?[1-9]|1[012])-([12][0-9]{3})$"); // check the Date Format

if (dateCheck) {

break;

} else {

System.out.println("Please Enter Valid Date Format Like this (dd-mm-yyyy) ");

}

} while (!dateCheck);

sportsClub = new FootballClub("", "", "", homeTeam, awayTeam, homeTeamScore, awayTeamScore, date);

manager.addPlayMatch(homeTeam, awayTeam, homeTeamScore, awayTeamScore, date);

PremierLeagueManager.listOfMatches.add(sportsClub); // Add the match into listOfMatches ArrayList

}

private static void deleteClub() {

System.out.print("\nEnter the Existing Club Id : ");

clubId = userInput.next();

manager.deleteExistFootballClub(clubId);

}

private static void DisplayClubDetails() {

System.out.print("\nEnter the Club Id: ");

clubId = userInput.next();

manager.displayClubDetails(clubId);

}

private static void userDecision() {

while (true) {

System.out.println("\nExit the Program Enter :: E \nContinue the Program Enter :: C");

System.out.println("\n:::::::::::::::::::::::::::::::Choose the Option You Want :::::::::::::::::::::::::::");

System.out.print("\nEnter the Option here : ");

userInput = new Scanner(System.in);

String decision = userInput.next();

if (decision.equalsIgnoreCase("e")) {

System.out.println("::::::::::::::::::::::::::::::: Exit the Program ::::::::::::::::::::::::::::::::");

decide = false;

break;

} else if (decision.equalsIgnoreCase("c")) {

decide = true;

break;

} else {

System.out.println("\nPlease type valid command");

}

}

}

}

## Question – 3

## Angular Code

### Points-table.component.html

<div>

<h2>Points Table</h2>

</div>

<div class="tbl">

<table class="table table-light table-striped ">

<thead class="thead-light">

<tr>

<th scope="col">Position</th>

<th scope="col">Club</th>

<th scope="col">Location</th>

<th scope="col">Wins</th>

<th scope="col">Draw</th>

<th scope="col">Defeat</th>

<th scope="col" >Goal Scored</th>

<th scope="col">Goal Against</th>

<th scope="col">Goal Difference</th>

<th scope="col">Points</th>

<th scope="col">Played Matches</th>

</tr>

</thead>

<tbody>

<tr class="text-center" \*ngFor="let col of matchTable " >

<td> {{col.position}}</td>

<td>{{col.nameOfClub}}</td>

<td>{{col.locationOfClub}}</td>

<td>{{col.numOfWin}}</td>

<td>{{col.numOfDraw}}</td>

<td>{{col.numOfDefeat}}</td>

<td>{{col.numOfScoredGoals}}</td>

<td>{{col.numOfReceivedGoals}}</td>

<td>{{col.goalDifference}}</td>

<td>{{col.numOfPoints}}</td>

<td>{{col.numOfPlayedMatches}}</td>

</tr>

</tbody>

</table>

</div>

/\*

# Reference

# Bootstrap Table Documentation

https://getbootstrap.com/docs/4.0/content/tables/

\*/

### Points-table.component.ts

@Component({

selector: 'app-points-table',

templateUrl: './points-table.component.html',

styleUrls: ['./points-table.component.css']

})

export class PointsTableComponent implements OnInit {

matchTable : any;

constructor(private pointsService : PointsTableService) { }

ngOnInit(){

this.pointsService.getAllClubDetails().subscribe((data) =>{

this.matchTable = data.response;

})

}

}

### points-table.service.ts

@Injectable({

providedIn: 'root'

})

export class PointsTableService {

constructor(private httpRequest : HttpClient) { }

getAllClubDetails(){

return this.httpRequest.get("http://localhost:9000/pointsTable");

}

}

### Random-match.component.html

<div class="head">

<h2>Random Match Table</h2>

</div>

<div>

<button class="btnRandom" mat-raised-button (click)="randomMatchGenerate()"><span>Random match</span> </button>

</div>

<div class="tbl">

<table class="table table-light table-striped">

<thead class="thead-light">

<tr>

<th scope="col">Date</th>

<th scope="col">Home Team</th>

<th scope="col">Away Team</th>

<th scope="col">Goal Scored </th>

<th scope="col">Goal Against</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let col of randomTable " >

<td>{{col.date}}</td>

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

<td>{{col.awayTeam}}</td>

<td>{{col.numOfScoredGoals}}</td>

<td>{{col.numOfReceivedGoals}}</td>

</tr>

</tbody>

</table>

</div>

### Random-match.component.ts

@Component({

selector: 'app-random-match',

templateUrl: './random-match.component.html',

styleUrls: ['./random-match.component.css']

})

export class RandomMatchComponent implements OnInit {

randomTable : any;

constructor(private randomService : RandomMatchService) { }

ngOnInit(){

this.randomService.getRandomMatch().subscribe((data) =>{

this.randomTable = data.response;

})

}

randomMatchGenerate(): void {

this.randomService.getRandomMatch()

.subscribe((data) => {

this.randomTable = data.response;

});

}}

### Random-match.service.ts

@Injectable({

providedIn: 'root'

})

export class RandomMatchService {

constructor(private httpRequest : HttpClient) { }

getRandomMatch(){

return this.httpRequest.get("http://localhost:9000/getRandom");

}

}

### Match-table.component.html

<div>

<h2>Match Table</h2>

</div>

<mat-form-field class="example-full-width" appearance="fill">

<mat-label>Search By Date</mat-label>

<input matInput type="text" [(ngModel)]="dateArray" />

</mat-form-field>

<div>

<button class="btnSort" mat-raised-button (click)="sortByDate()"> <span>Sort By Date </span></button>

</div>

<div class="tbl">

<table class="table table-light table-striped ">

<thead class="thead-light">

<tr>

<th scope="col">Date</th>

<th scope="col">Home Team</th>

<th scope="col">Away Team</th>

<th scope="col">Goal Scored </th>

<th scope="col">Goal Against</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let col of footballClubMatch | dateSearch:dateArray" >

<td >{{col.date}}</td>

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

<td>{{col.awayTeam}}</td>

<td>{{col.numOfScoredGoals}}</td>

<td>{{col.numOfReceivedGoals}}</td>

</tr>

</tbody>

</table>

</div>

### Match-table.component.ts

@Component({

selector: 'app-match-table',

templateUrl: './match-table.component.html',

styleUrls: ['./match-table.component.css']

})

export class MatchTableComponent implements OnInit {

footballClubMatch : any;

dateArray="";

constructor(private matchService : MatchTableService ) { }

ngOnInit() {

this.matchService.getMatchTable()

.subscribe((data) =>{

this.footballClubMatch = data.response;

})

}

sortByDate(): void {

this.matchService.sortDate()

.subscribe((data) => {

this.footballClubMatch = data.response;

})

}

}

### Match-table.service.ts

@Injectable({

providedIn: 'root'

})

export class MatchTableService {

constructor(private httpRequest : HttpClient) { }

getMatchTable(){

return this.httpRequest.get("http://localhost:9000/getMatch");

}

sortDate(){

return this.httpRequest.get("http://localhost:9000/getDate");

}

}

## Play Framework Codes

## Controllers

### MatchTableController.java

public class MatchTableController {

public Result MatchTableData() {

PointsTableService.loadTheData();

ObjectMapper objectMap = new ObjectMapper();

JsonNode dataOfJson = objectMap.convertValue(PointsTableService.listOfServiceMatches, JsonNode.class);

return created(ResponseUtil.createResponse(dataOfJson,true));

}

public Result RandomTableData() {

PointsTableService.get();

ObjectMapper objectMap = new ObjectMapper();

JsonNode dataOfJson = objectMap.convertValue(PointsTableService.listOfServiceMatches, JsonNode.class);

return created(ResponseUtil.createResponse(dataOfJson, true));

}

public Result SortedDate() {

DateService.getDate();

ObjectMapper objectMap = new ObjectMapper();

JsonNode dataOfJson = objectMap.convertValue(PointsTableService.listOfServiceMatches, JsonNode.class);

return created(ResponseUtil.createResponse(dataOfJson, true));

}

}

### PointsTableController.java

public class PointsTableController extends Controller {

public Result pointsTableData() {

PointsTableService.loadTheData();

ObjectMapper objectMap = new ObjectMapper();

JsonNode dataOfJson = objectMap.convertValue(PointsTableService.listOfServiceSportsClubs, JsonNode.class);

return created(ResponseUtil.createResponse(dataOfJson, true));

}

}

## Services

### DateService.java

public class DateService implements Comparator<SportsClub> {

public static void getDate() {

PointsTableService.loadTheData(); // load the data from Points Table Service

PointsTableService.listOfServiceMatches.sort(new DateService()); // Sort the data

}

DateTimeFormatter formatOfDate = DateTimeFormatter.ofPattern("dd-MM-yyyy"); // Formatting the Date

@Override

public int compare(SportsClub firstSportsClub, SportsClub secondSportsClub) {

LocalDate firstDate = LocalDate.parse(((FootballClub) firstSportsClub).getDate(), formatOfDate); // get the date and assign to firstDate

LocalDate secondDate = LocalDate.parse(((FootballClub) secondSportsClub).getDate(), formatOfDate); // get the date and assign to secondDate

try {

return firstDate.compareTo(secondDate); // Comparing the first and second date

} catch (Exception exception) {

throw new IllegalArgumentException(exception);

}

}

}

## PointsTableService.java

public class PointsTableService {

public static ArrayList <SportsClub> listOfServiceSportsClubs = new ArrayList();

public static ArrayList <SportsClub> listOfServiceMatches = new ArrayList();

private static int point;

private static int win;

private static int drawn;

private static int lost;

private static int firstPositionOfIndex;

private static int secondPositionOfIndex;

public static void loadTheData() {

try {

FileInputStream fileOfInput = new FileInputStream("premierLeagueData.txt");

ObjectInputStream objectOfOutput = new ObjectInputStream(fileOfInput);

listOfServiceSportsClubs = (ArrayList) objectOfOutput.readObject(); // get the the data from file and assign the listOfServiceSportsClubs List

listOfServiceMatches = (ArrayList) objectOfOutput.readObject(); // get the the data from file and assign the listOfServiceMatches List

objectOfOutput.close();

fileOfInput.close();

} catch (IOException ioException) {

ioException.printStackTrace();

} catch (ClassNotFoundException classNotFound) {

System.out.println("Class not found");

classNotFound.printStackTrace();

}

}

public static void saveTheData() {

try {

FileOutputStream fileOfInput = new FileOutputStream("premierLeagueData.txt");

ObjectOutputStream objectOfOutput = new ObjectOutputStream(fileOfInput);

objectOfOutput.writeObject(listOfServiceSportsClubs);

objectOfOutput.writeObject(listOfServiceMatches);

System.out.println("Data Saved Successfully");

objectOfOutput.flush();

objectOfOutput.close();

fileOfInput.close();

} catch (IOException ioe) {

ioe.printStackTrace();

}

}

public static void get(){

loadTheData();

randomMatch();

saveTheData();

}

public static void randomMatch() {

java.text.SimpleDateFormat formatOfDate = new java.text.SimpleDateFormat("dd-MM-yyyy");

while (true) {

firstPositionOfIndex = new Random().nextInt(listOfServiceSportsClubs.size());

secondPositionOfIndex = new Random().nextInt(listOfServiceSportsClubs.size());

if (firstPositionOfIndex != secondPositionOfIndex) {

break;

}

}

String homeTeam = listOfServiceSportsClubs.get(firstPositionOfIndex).getNameOfClub(); // Pick the one club randomly and assign to home team

String awayTeam = listOfServiceSportsClubs.get(secondPositionOfIndex).getNameOfClub(); // Pick the one club randomly and assign to away team

int homeTeamGoal = new Random().nextInt(15); // Randomly pick one number and assign to Home team goal

int awayTeamGoal = new Random().nextInt(15); // Randomly pick one number and assign to away team goal

// Randomly generate the date

Random randomDate = new Random();

java.util.Calendar calenderClass = java.util.Calendar.getInstance();

calenderClass.set(java.util.Calendar.MONTH, Math.abs(randomDate.nextInt()) % 12);

calenderClass.set(java.util.Calendar.DAY\_OF\_MONTH, Math.abs(randomDate.nextInt()) % 30);

calenderClass.setLenient(true);

String date = formatOfDate.format(calenderClass.getTime());

addPlayMatch(homeTeam, awayTeam, homeTeamGoal, awayTeamGoal, date);

FileHandleUtil.saveData();

SportsClub sportsClub = new FootballClub("","", "", homeTeam, awayTeam, homeTeamGoal, awayTeamGoal, date);

listOfServiceMatches.add(sportsClub);

}

public static void addPlayMatch(String homeTeam, String awayTeam, int homeTeamScore, int awayTeamScore, String date) {

int count = 0;

for (SportsClub sportsClub : listOfServiceSportsClubs) {

int difference1 = homeTeamScore + ((FootballClub) sportsClub).getNumOfScoredGoals() - awayTeamScore + ((FootballClub) sportsClub).getNumOfReceivedGoals();

int difference2 = awayTeamScore + ((FootballClub) sportsClub).getNumOfScoredGoals() - homeTeamScore + ((FootballClub) sportsClub).getNumOfReceivedGoals();

if (sportsClub.getNameOfClub().equals(homeTeam)) {

checkPoints(homeTeam, homeTeamScore, awayTeamScore, date, count, sportsClub, difference1);

} else if (sportsClub.getNameOfClub().equals(awayTeam)) {

checkPoints(awayTeam, awayTeamScore, homeTeamScore, date, count, sportsClub, difference2);

}

count++;

}

int position = 0;

listOfServiceSportsClubs.sort(Collections.reverseOrder());

for(SportsClub sportsClub2 : listOfServiceSportsClubs) {

SportsClub sportsClub3 = new FootballClub(sportsClub2.getSportsClubId(),sportsClub2.getNameOfClub(), sportsClub2.getLocationOfClub(), ((FootballClub) sportsClub2).getNumOfWin(), ((FootballClub) sportsClub2).getNumOfDraw(), ((FootballClub) sportsClub2).getNumOfDefeat(), ((FootballClub) sportsClub2).getNumOfScoredGoals(), ((FootballClub) sportsClub2).getNumOfReceivedGoals(),

((FootballClub) sportsClub2).getGoalDifference(), ((FootballClub) sportsClub2).getNumOfPoints(), ((FootballClub) sportsClub2).getNumOfPlayedMatches(), ((FootballClub) sportsClub2).getDate(), position + 1);

listOfServiceSportsClubs.set(position, sportsClub3);

position++;

}

}

private static void checkPoints(String homeTeam, int homeTeamScore, int awayTeamScore, String date, int count, SportsClub sportsClub, int difference1) {

score(homeTeamScore, awayTeamScore, ((FootballClub) sportsClub));

SportsClub sportsClub1 = new FootballClub(sportsClub.getSportsClubId(),homeTeam, sportsClub.getLocationOfClub(), win, drawn, lost, homeTeamScore + ((FootballClub) sportsClub).getNumOfScoredGoals(), awayTeamScore + ((FootballClub) sportsClub).getNumOfReceivedGoals(),

difference1, point, ((FootballClub) sportsClub).getNumOfPlayedMatches() + 1, date, ((FootballClub) sportsClub).getPosition());

listOfServiceSportsClubs.set(count, sportsClub1);

}

private static void score(int homeTeamScore, int awayTeamScore, FootballClub footballClub) {

win = 0;

point = 0;

drawn = 0;

lost = 0;

if (homeTeamScore > awayTeamScore) {

win = footballClub.getNumOfWin() + 1;

point = footballClub.getNumOfPoints() + 3;

} else if (homeTeamScore == awayTeamScore) {

drawn = footballClub.getNumOfDraw() + 1;

point = footballClub.getNumOfPoints() + 1;

} else {

lost = footballClub.getNumOfDefeat() + 1;

point = footballClub.getNumOfPoints();

}

}

}

# Junit Test

## Code

### premierLeagueManagerTest

class PremierLeagueManagerTest {

@Test

void addFootballClub() {

List <SportsClub> sportsClubList = new ArrayList<>();

SportsClub newFootballClub = new FootballClub("C001","Barcelona","Sapin","","",0,0,"");

sportsClubList.add(newFootballClub); // adding new Football Club into sportClubList

assertTrue("Successfully added into SportsClub List",sportsClubList.add(newFootballClub));

assertEquals(true,sportsClubList.contains(newFootballClub));

}