

import java.io.Serializable;

import java.util.Comparator;

import java.util.Objects;

public abstract class SportsClub implements Serializable {

private static final long serialVersionUID = 2906642554793891381L;

private String clubName;

private String clubLocation;

public SportsClub(String clubName, String clubLocation) {

this.clubName = clubName;

this.clubLocation = clubLocation;

}

public String getClubName() {

if (clubName.matches("^[A-Za-z ]\*$")) {

return clubName;

} else {

System.out.println("");

System.out.println(" ! ! ! ! ! ! Name Cannot contain Number or Characters and Numbers ! ! ! ! ! ! ");

}

return clubName;

}

public void setClubName(String clubName) {

this.clubName = clubName;

}

public String getClubLocation() {

return clubLocation;

}

public void setClubLocation(String clubLocation) {

this.clubLocation = clubLocation;

}

static class CustomComparator implements Comparator<FootballClub> {

public int compare(FootballClub t, FootballClub t1) {

if (t.getNumberOfPointsHas() > t1.getNumberOfPointsHas())

return -1;

else if (t.getNumberOfPointsHas() < t1.getNumberOfPointsHas())

return 1;

else {

int goalDifference = t.getNumberOfGoalsScored() - t.getNumberOfGoalsReceived();

int goalDifference1 = t1.getNumberOfGoalsScored() - t1.getNumberOfGoalsReceived();

if (goalDifference > goalDifference1)

return -1;

else if (goalDifference < goalDifference1)

return 1;

else return 0;

}

}

}

static class CustomComparatorForWins implements Comparator<FootballClub> {

@Override

public int compare(FootballClub o1, FootballClub o2) {

return o1.getNumberOfWins() - o2.getNumberOfWins();

}

}

static class CustomComparatorForGoals implements Comparator<FootballClub> {

@Override

public int compare(FootballClub f1, FootballClub f2) {

return f1.getNumberOfGoalsScored() - f2.getNumberOfGoalsScored();

} }}

import java.io.Serializable;

import java.util.Objects;

public class FootballClub extends SportsClub implements Serializable {

private int numberOfWins;

private int numberOfDefeats;

private int numberOfDraws;

private int numberOfGoalsReceived;

private int numberOfGoalsScored;

private int numberOfPointsHas;

private int numberOfPlayedMatches;

public FootballClub(String clubName, String clubLocation, int numberOfWins, int numberOfDefeats, int numberOfDraws, int numberOfGoalsReceived, int numberOfGoalsScored, int numberOfPointsHas, int numberOfPlayedMatches) {

super(clubName, clubLocation);

this.numberOfWins = numberOfWins;

this.numberOfDefeats = numberOfDefeats;

this.numberOfDraws = numberOfDraws;

this.numberOfGoalsReceived = numberOfGoalsReceived;

this.numberOfGoalsScored = numberOfGoalsScored;

this.numberOfPointsHas = numberOfPointsHas;

this.numberOfPlayedMatches = numberOfPlayedMatches;

}

public int getNumberOfWins() {

return numberOfWins;

}

public void setNumberOfWins(int numberOfWins) {

this.numberOfWins = numberOfWins;

}

public int getNumberOfDefeats() {

return numberOfDefeats;

}

public void setNumberOfDefeats(int numberOfDefeats) {

this.numberOfDefeats = numberOfDefeats;

}

public int getNumberOfDraws() {

return numberOfDraws;

}

public void setNumberOfDraws(int numberOfDraws) {

this.numberOfDraws = numberOfDraws;

}

public int getNumberOfGoalsReceived() {

return numberOfGoalsReceived;

}

public void setNumberOfGoalsReceived(int numberOfGoalsReceived) {

this.numberOfGoalsReceived = numberOfGoalsReceived;

}

public int getNumberOfGoalsScored() {

return numberOfGoalsScored;

}

public void setNumberOfGoalsScored(int numberOfGoalsScored) {

this.numberOfGoalsScored = numberOfGoalsScored;

}

public double getNumberOfPointsHas() {

return numberOfPointsHas;

}

public void setNumberOfPointsHas(int numberOfPointsHas) {

this.numberOfPointsHas = numberOfPointsHas;

}

public int getNumberOfPlayedMatches() {

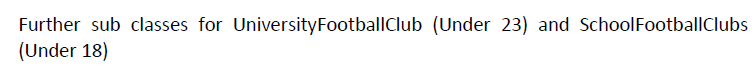
return numberOfPlayedMatches;

}

public void setNumberOfPlayedMatches(int numberOfPlayedMatches) {

this.numberOfPlayedMatches = numberOfPlayedMatches;

}



import java.io.Serializable;

import java.util.Objects;

public class UniversityFootballClub implements Serializable {

private String universityName;

public UniversityFootballClub(String universityName) {

this.universityName = universityName;

}

public String getUniversityName() {

return universityName;

}

public void setUniversityName(String universityName) {

this.universityName = universityName;

}

}

import java.io.Serializable;

import java.util.Objects;

public class SchoolFootballClub implements Serializable {

private String schoolName;

public SchoolFootballClub(String schoolName) {

this.schoolName = schoolName;

}

public String getSchoolName() {

return schoolName;

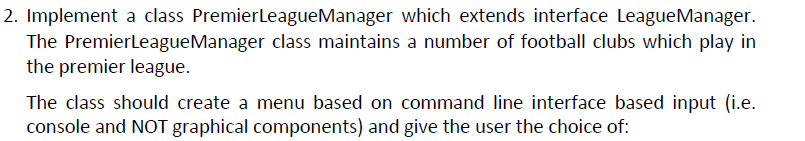
}

public void setSchoolName(String schoolName) {

this.schoolName = schoolName;

}

}



import java.io.\*;

import java.time.LocalDate;

import java.time.format.DateTimeFormatter;

import java.time.format.DateTimeParseException;

import java.time.format.ResolverStyle;

import java.util.\*;

public class PremierLeagueManager implements LeagueManager {

private final int numberOfClubs = 20;

private static ArrayList<FootballClub> league = new ArrayList<>();

private final Scanner premierLeagueScanner = new Scanner(System.in);

private static ArrayList<Match> matches = new ArrayList<>();

private static PremierLeagueManager instance = null;

PremierLeagueManager() {

}

public static PremierLeagueManager getInstance(){

if (instance == null){

synchronized (PremierLeagueManager.class){

if (instance == null){

instance = new PremierLeagueManager();

}

}

}

return instance;

}

import java.io.IOException;

import java.text.ParseException;

public interface LeagueManager {

void addNewTeam();

void deleteAnExistingTeam();

void displayListOfStatistics();

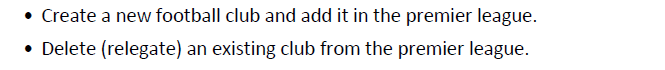
void displayLeagueTable();

void addPlayedMatch() throws IOException, ParseException;

void saveData() throws IOException;

void loadData() throws IOException, ClassNotFoundException;

}



@Override

public void addNewTeam() {

System.out.println("\n\t\t++++++ ADD A NEW FOOTBALL CLUB INTO THE PREMIER LEAGUE +++++++++\n");

if (league.size() == numberOfClubs) {

System.out.println("\n\t\t ! ! ! ! ! ! No enough space to add a new Football Club ! ! ! ! ! ! \n");

return;

}

System.out.println(" # Club Name :");

String clubNameInput = premierLeagueScanner.nextLine();

while (true) {

if (NameLocationValidator(clubNameInput) == true) {

break;

} else {

System.out.println(" # Club Name :");

clubNameInput = premierLeagueScanner.nextLine();

}

}

boolean clubChecker = false;

for (FootballClub club : league) {

if (club.getClubName().equals(clubNameInput)) {

clubChecker = true;

System.out.println("\n\t\t ! ! ! ! ! ! This Club is already in the league ! ! ! ! ! ! \n");

}

}

if (clubChecker == true) {

return;

}

System.out.println("");

System.out.println(" # Club Location : ");

String clubLocationInput = premierLeagueScanner.nextLine();

FootballClub club = new FootballClub(clubNameInput, clubLocationInput, 0, 0, 0, 0, 0, 0, 0);

league.add(club);

System.out.println();

System.out.println("\n\t\t \*\*\*\*\*\*\*\*\*\* " + club.getClubName() + " club added successfully \*\*\*\*\*\*\*\*\*\* \n");

}

@Override

public void deleteAnExistingTeam() {

System.out.println("\n\t\t++++++ DELETE EXISTING MEMBERS +++++++++\n");

if (league.size() == 0) {

System.out.println("\n\t\t ! ! ! ! ! ! No clubs to delete ! ! ! ! ! ! \n");

return;

}

System.out.print(" # Club Name :");

String deleteClubName = premierLeagueScanner.nextLine();

for (int premiereLeagueLoop = 0; premiereLeagueLoop < league.size(); premiereLeagueLoop++) {

FootballClub club = league.get(premiereLeagueLoop);

if (club.getClubName().equals(deleteClubName)) {

league.remove(club);

System.out.println("\n\t\t \*\*\*\*\*\*\*\*\*\* " + club.getClubName() + " club removed successfully \*\*\*\*\*\*\*\*\*\* \n");

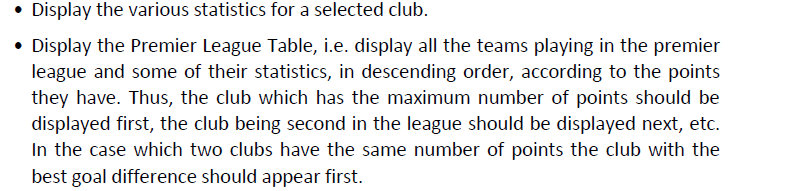
return;

}

}

System.out.println("\n\t\t ! ! ! ! ! ! This Club is not in the league ! ! ! ! ! ! \n");

}



@Override

public void displayListOfStatistics() throws NullPointerException {

System.out.println("\n\t\t++++++ DISPLAY STATISTICS +++++++++\n");

System.out.println(" # Club Name :");

String statsClubName = premierLeagueScanner.nextLine();

while (true) {

if (NameLocationValidator(statsClubName) == true) {

if (league.size() == 0) {

System.out.println("\n\t\t ! ! ! ! ! ! No clubs to display ! ! ! ! ! ! \n");

break;

} else {

for (int premiereLeagueLoop = 0; premiereLeagueLoop < league.size(); premiereLeagueLoop++) {

FootballClub club = league.get(premiereLeagueLoop);

if (club.getClubName().equals(statsClubName)) {

System.out.println(" -- Number of matches won by " + club.getClubName() + " club : " + club.getNumberOfWins());

System.out.println(" -- Number of matches lost " + club.getClubName() + " club : " + club.getNumberOfDefeats());

System.out.println(" -- Number of matches draw " + club.getClubName() + " club : " + club.getNumberOfDraws());

System.out.println(" -- Number of scored goals " + club.getClubName() + " club : " + club.getNumberOfGoalsScored());

System.out.println(" -- Number of received goals " + club.getClubName() + " club : " + club.getNumberOfGoalsReceived());

System.out.println(" -- Number of points has " + club.getClubName() + " club : " + club.getNumberOfPointsHas());

System.out.println(" -- Number of matches played " + club.getClubName() + " club : " + club.getNumberOfPlayedMatches());

return;

} else {

System.out.println("\n\t\t ! ! ! ! ! ! This Club is not in the league ! ! ! ! ! ! \n");

System.out.println();

}

}

}

}

break;

}

}

@Override

public void displayLeagueTable() {

System.out.println("\n\t\t++++++ PREMIER LEAGUE TABLE +++++++++\n");

if (league.size() == 0) {

System.out.println("\n\t\t ! ! ! ! ! ! No clubs to show ! ! ! ! ! ! \n");

return;

}

System.out.println("Club Points Wins Defeats Draws Goals-Scored Goals-Received");

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

Collections.sort(league, new SportsClub.CustomComparator());

for (int premiereLeagueLoop = 0; premiereLeagueLoop < league.size(); premiereLeagueLoop++) {

FootballClub club = league.get(premiereLeagueLoop);

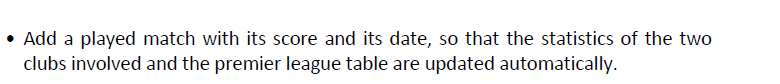
{

System.out.println(club.getClubName()+" "+club.getNumberOfPointsHas()+" "+club.getNumberOfWins()+" "+club.getNumberOfDefeats()+" "+club.getNumberOfDraws()+" "+club.getNumberOfGoalsScored()+" "+club.getNumberOfGoalsScored());

}

}

}



@Override

public void addPlayedMatch() throws NullPointerException {

System.out.println(" # Enter date (format YYYY-M-D): ");

String userInput = premierLeagueScanner.nextLine();

Date date = null;

while (true) {

if (DateValidatorDateTimeFormatter(userInput) == true) {

break;

} else {

System.out.println(" # Enter date (format YYYY-M-D): ");

userInput = premierLeagueScanner.nextLine();

}

}

System.out.println(" # Enter Home Team: ");

userInput = premierLeagueScanner.nextLine();

FootballClub homeTeam = null;

for (FootballClub club : league) {

if (club.getClubName().equals(userInput))

homeTeam = club;

}

if (homeTeam == null) {

System.out.println("\n\t\t ! ! ! ! ! ! No such club in league ! ! ! ! ! ! \n");

return;

}

System.out.println(" # Enter Away Team: ");

userInput = premierLeagueScanner.nextLine();

FootballClub awayTeam = null;

for (FootballClub club : league) {

if (club.getClubName().equals(userInput))

awayTeam = club;

}

if (awayTeam == null) {

System.out.println("\n\t\t ! ! ! ! ! ! This Club is not in the league ! ! ! ! ! ! \n");

return;

}

System.out.println(" # Enter home team goals: ");

userInput = premierLeagueScanner.nextLine();

int numberOfHomeGoals = -1;

try {

numberOfHomeGoals = Integer.parseInt(userInput);

} catch (Exception e) {

}

if (numberOfHomeGoals == -1) {

System.out.println("\n\t\t ! ! ! ! ! ! You have to enter number of goals ! ! ! ! ! ! \n");

return;

}

System.out.println(" # Enter away team goals: ");

userInput = premierLeagueScanner.nextLine();

int numberOfAwayGoals = -1;

try {

numberOfAwayGoals = Integer.parseInt(userInput);

} catch (Exception e) {

}

if (numberOfAwayGoals == -1) {

System.out.println("\n\t\t ! ! ! ! ! ! You have to enter number of goals ! ! ! ! ! ! \n");

return;

}

System.out.println("\n\t\t \*\*\*\*\*\*\*\*\*\* Played match added successfully \*\*\*\*\*\*\*\*\*\* \n");

Match match = new Match(awayTeam, homeTeam, 0, 0, date);

match.setDate(date);

match.setTeamA(homeTeam);

match.setTeamB(awayTeam);

match.setTeamAScore(numberOfAwayGoals);

match.setTeamBScore(numberOfHomeGoals);

matches.add(match);

homeTeam.setNumberOfGoalsScored(homeTeam.getNumberOfGoalsScored() + numberOfHomeGoals);

awayTeam.setNumberOfGoalsScored(awayTeam.getNumberOfGoalsScored() + numberOfAwayGoals);

homeTeam.setNumberOfGoalsReceived(homeTeam.getNumberOfGoalsReceived() + numberOfAwayGoals);

awayTeam.setNumberOfGoalsReceived(awayTeam.getNumberOfGoalsReceived() + numberOfHomeGoals);

homeTeam.setNumberOfPlayedMatches(homeTeam.getNumberOfPlayedMatches() + 1);

awayTeam.setNumberOfPlayedMatches(awayTeam.getNumberOfPlayedMatches() + 1);

if (numberOfHomeGoals > numberOfAwayGoals) {

homeTeam.setNumberOfPointsHas((int) (homeTeam.getNumberOfPointsHas() + 3));

homeTeam.setNumberOfWins(homeTeam.getNumberOfWins() + 1);

awayTeam.setNumberOfDefeats(awayTeam.getNumberOfDefeats() + 1);

} else if (numberOfHomeGoals < numberOfAwayGoals) {

awayTeam.setNumberOfPointsHas((int) awayTeam.getNumberOfPointsHas() + 3);

awayTeam.setNumberOfWins(awayTeam.getNumberOfWins() + 1);

homeTeam.setNumberOfDefeats(homeTeam.getNumberOfDefeats() + 1);

} else {

homeTeam.setNumberOfPointsHas((int) homeTeam.getNumberOfPointsHas() + 1);

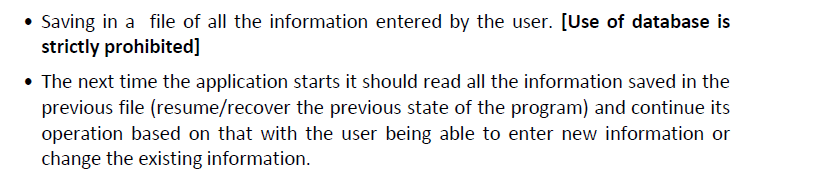
awayTeam.setNumberOfPointsHas((int) awayTeam.getNumberOfPointsHas() + 1);

homeTeam.setNumberOfDraws(homeTeam.getNumberOfDraws() + 1);

awayTeam.setNumberOfDraws(awayTeam.getNumberOfDraws() + 1);

}

}



@Override

public void saveData() throws IOException {

try{

File FootballClubCollection= new File(" FootballClubCollection.TXT");

FileOutputStream footballOutputStream = new FileOutputStream(FootballClubCollection);

ObjectOutputStream footballOOS = new ObjectOutputStream(footballOutputStream);

for (FootballClub club : league){

footballOOS.writeObject(club);

}

footballOOS.flush();

footballOutputStream.close();

footballOOS.close();

}catch (FileNotFoundException e){

System.out.println("\n\t\t ! ! ! ! ! ! File not Found ! ! ! ! ! ! \n");

}

try {

File MatchCollection= new File(" MatchCollection.txt");

FileOutputStream matchOutputStream = new FileOutputStream(MatchCollection);

ObjectOutputStream matchOOS = new ObjectOutputStream(matchOutputStream);

for (Match match : matches){

matchOOS.writeObject(match);

}

matchOOS.flush();

matchOutputStream.close();

matchOOS.close();

System.out.println("\n\t\t ! ! ! ! ! ! Data has been saved successfully ! ! ! ! ! ! \n");

}catch (FileNotFoundException e){

System.out.println("\n\t\t ! ! ! ! ! ! File not Found ! ! ! ! ! ! \n");

}

}

/\*tis method is commanded to trigger at beginning of each compilation. it will automatically store data

\* that stored in the files to the system. so user can continue where stopped. this method throws IOException

\* and ClassNotFoundException.\*/

@Override

public void loadData() throws IOException, ClassNotFoundException {

try {

File FootballClubCollection = new File(" FootballClubCollection.txt");

FileInputStream footballInputStream = new FileInputStream(FootballClubCollection);

ObjectInputStream footballOIS = new ObjectInputStream(footballInputStream);

for (; ; ) {

try {

FootballClub club = (FootballClub) footballOIS.readObject();

league.add(club);

} catch (EOFException e) {

break;

}

}

footballInputStream.close();

footballOIS.close();

File MatchCollection = new File(" MatchCollection.txt");

FileInputStream matchInputStream = new FileInputStream(MatchCollection);

ObjectInputStream matchOIS = new ObjectInputStream(matchInputStream);

for (; ; ) {

try {

Match match = (Match) matchOIS.readObject();

matches.add(match);

} catch (EOFException e) {

break;

}

}

matchInputStream.close();

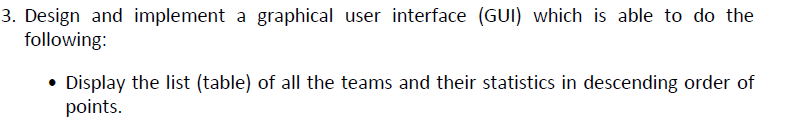
matchOIS.close();

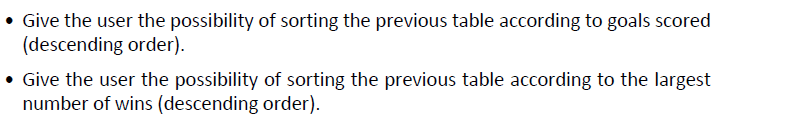
} catch (FileNotFoundException e) {

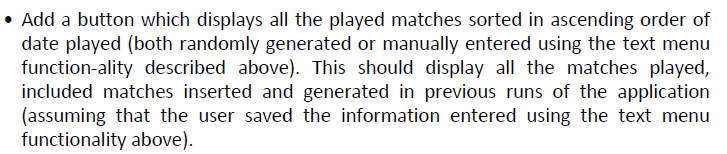
System.out.println("\n\t\t ! ! ! ! ! ! File not Found ! ! ! ! ! ! \n");

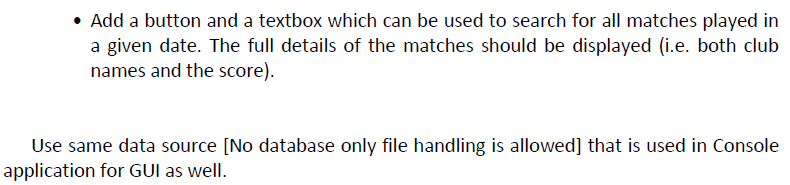
}

}









"I confirm that I understand what plagiarism / collusion / contract cheating is and have read and understood the section on Assessment Offences in the Essential Information for Students. The work that I have submitted is entirely my own. Any work from other authors is duly referenced and acknowledged."