import java.io.\*;  
import java.util.\*;  
import java.text.\*;  
public class Menu {  
 private Engine e;  
 private Scanner console;  
   
 //constructor  
 public Menu (Engine engine)throws Exception {  
 this.e = engine;  
 this.e.loadData();  
 this.console = new Scanner(System.in);   
 }  
   
 //mainMenu also returns a boolean which decides if the program should keep running  
 //by default it's true, but if the console input is 0, mainMenu returns false and stops the program.  
 public boolean mainMenu() {  
 System.out.println("Main menu");  
 System.out.println("1. Members");  
 System.out.println("2. Competitive");  
 System.out.println("3. Economy");  
 System.out.println("\n0. Exit");  
 System.out.print("\nSelect: ");  
 int input = console.nextInt();  
 switch (input) {  
 case 1:  
 membersMenu();  
 break;  
 case 2:  
 recordMenu();  
 break;  
 case 3:  
 economyMenu();  
 break;  
 case 0:  
 System.exit(0);  
 default:  
 System.out.println("Wrong input!\n");   
 }  
 return true;  
 }  
   
 //SÃ¸ren, Martin  
 public void membersMenu() {  
 System.out.println("\nMembers menu");  
 System.out.println("1. Create");  
 System.out.println("2. Show");  
 System.out.println("3. Edit");  
 System.out.println("4. Delete");  
 System.out.println();  
 System.out.println("0. Back");  
 System.out.println();  
   
 System.out.print("Select: ");  
 int input = console.nextInt();  
   
 switch (input) {  
 case 1:  
 createMember();  
 break;  
 case 2:  
 showMemberMenu();  
 break;   
 case 3:  
 editMember();  
 membersMenu();  
 break;  
 case 4:  
 deleteMember();  
 break;  
 case 0:  
 System.out.println("Returning to main menu.\n\n");  
 mainMenu();  
 break;  
 default:  
 System.out.println("There is no menu with that number. Returning to main menu.\n\n");  
 mainMenu();  
 break;  
 }  
 }  
   
 //Bertram, SÃ¸ren, Martin  
 public void deleteMember() {  
 System.out.println("Write CPR of member to delete: ");  
 console.nextLine();  
 String cpr = console.nextLine();  
   
 Member m = e.getMember(cpr);  
 if (m == null) {  
 System.out.println("Failed to find a member from CPR (" + cpr + ")!");  
 membersMenu();  
 }  
 String deletedMember = m.getFirstName() + " " + m.getLastName();  
 System.out.print("Are you sure? The member will no longer exist in the system (y/n): ");  
 String answer = console.nextLine();  
 if (answer.equalsIgnoreCase("y")) {  
 try {  
 e.deleteRecordsByMember(m);  
 e.deleteMember(m);  
 e.saveMembers();  
 e.saveRecords();  
 System.out.println("Member: " + deletedMember + " Has been deleted.");  
 System.out.println("\n");  
 }   
 catch (Exception error) {  
 System.out.println("Something did not go as planned.");  
 System.out.println("\n");  
 }  
 membersMenu();  
 }   
 else {  
 System.out.println("Returning.");  
 membersMenu();  
 }  
   
 }  
   
 //Bertram, SÃ¸ren, Martin  
 public void editMember() {  
 System.out.println("Write CPR of member to edit: ");  
 console.nextLine();  
 String cpr = console.nextLine();   
   
 Member m = e.getMember(cpr);  
 if (m == null) {  
 System.out.println("Failed to find a member from CPR (" + cpr + ")!");  
 membersMenu();  
 }  
   
 System.out.println("Select information to update: ");  
 System.out.println("1. First name");  
 System.out.println("2. Last name");  
 System.out.println("3. Phone number");  
 System.out.println("4. Address");  
 System.out.println("5. Activity");  
 if (m.getIsActive()) {  
 System.out.println("6. Competitive");  
 }  
 System.out.println();  
 System.out.println("0. Back");  
   
 int input = console.nextInt();  
 console.nextLine();  
 switch (input) {  
 case 1:  
 System.out.print("New first name: ");  
 String fName = console.nextLine();  
 m.setFirstName(fName);  
 break;  
 case 2:  
 System.out.print("New last name: ");  
 String lName = console.nextLine();  
 m.setLastName(lName);  
 break;  
 case 3:  
 System.out.print("New phone number: ");  
 String phone = console.nextLine();  
 m.setPhone(phone);  
 break;  
 case 4:  
 System.out.print("New address: ");  
 String address = console.nextLine();  
 m.setAddress(address);  
 break;  
 case 5:  
 System.out.print("Set activity (true/false): ");  
 boolean activity = console.nextBoolean();  
 m.setIsActive(activity);  
 break;  
 case 6:  
 System.out.print("Set competivity (true/false): ");  
 boolean competitive = console.nextBoolean();  
 console.nextLine();  
 if (competitive) {  
 try {  
 CompetitiveMember cm = new CompetitiveMember(m.getCpr(), m.getFirstName(), m.getLastName(), m.getPhone(), m.getAddress(), m.getIsActive(), true, m.getBalance(), e.getSubscriptions(), e.getTeams(), selectDisciplines());  
 e.addMember(cm);  
 e.deleteMember(m);  
 e.saveMembers();  
 }   
 catch (Exception error) {  
 System.out.println("Something happened.");  
 }  
 }   
 else {  
 m.setIsCompetitive(false);  
 try {  
 Member convertedM = new Member(m.getCpr(), m.getFirstName(), m.getLastName(), m.getPhone(), m.getAddress(), m.getIsActive(), false, m.getBalance(), e.getSubscriptions());  
 System.out.println(convertedM);  
 e.addMember(convertedM);  
 e.deleteMember(m);  
 e.saveMembers();  
 }   
 catch (Exception Error) {  
 System.out.println("Something went south.");  
 }  
 }  
 break;  
 case 0:  
 System.out.println("Returning to edit menu.\n\n");  
 membersMenu();  
 break;  
 default:  
 System.out.println("There is no menu with that number. Returning to edit menu.\n\n");  
 editMember();  
 break;  
 }  
   
 }  
   
 //SÃ¸ren, Martin  
 public void showMemberMenu() {  
 System.out.println("1. Show all members");  
 System.out.println("2. Specify by CPR number");  
 System.out.println();  
 System.out.println("0. Back");  
 System.out.print("\nSelect: ");  
 int subInput = console.nextInt();  
 System.out.println();  
 System.out.println();  
   
 switch (subInput) {  
 case 1:  
 showAllMembers();  
 showMemberMenu();  
 break;  
 case 2:  
 showMember();  
 showMemberMenu();  
 break;  
 default:  
 System.out.println("There is no menu with that number. Returning to main menu.\n\n");  
 mainMenu();  
 break;  
 }  
 }  
   
 //Bertram, SÃ¸ren, Martin  
 public void createMember() {  
 console.nextLine();  
 System.out.print("CPR: ");  
 String cpr = console.nextLine();  
 while (cpr.length() != 11) {  
 System.out.println("Invalid CPR length, please try again");  
 System.out.println("Correct format is XXXXXX-XXXX");  
 System.out.println();  
 createMember();  
 }   
 System.out.print("First name: ");  
 String fName = console.nextLine();  
 System.out.print("Last name: ");  
 String lName = console.nextLine();  
 System.out.print("Phone number: ");  
 String phone = console.nextLine();  
 System.out.print("Address: ");  
 String address = console.nextLine();  
 System.out.print("Active member: (y/n) ");  
 boolean active = false;  
   
 if (console.nextLine().equalsIgnoreCase("y")) {  
 active = true;  
 }  
 boolean competitive = false;  
 System.out.print("Competitive member: (y/n) ");  
 ArrayList<Discipline> activeDisciplines = new ArrayList<>();  
 if (console.nextLine().equalsIgnoreCase("y")) {  
 competitive = true;  
 activeDisciplines = selectDisciplines();  
 }  
   
 System.out.print("Balance: ");  
 double balance = console.nextDouble();  
   
 ArrayList<Subscription> subsciptions = e.getSubscriptions();  
   
 try {  
 if (!competitive) {  
 Member m = new Member(cpr, fName, lName, phone, address, active, competitive, balance, subsciptions);  
 e.addMember(m);  
 }   
 else {  
 CompetitiveMember cm = new CompetitiveMember(cpr, fName, lName, phone, address, active, competitive, balance, subsciptions, e.getTeams(), activeDisciplines);  
 e.addMember(cm);  
 }  
 e.saveMembers();  
 System.out.println("Added new member: " + fName + " " + lName);  
 System.out.println();  
 }   
 catch (Exception error) {  
 System.out.println("Failed to create object");  
 }  
   
 mainMenu();  
   
 }  
   
 //Bertram, SÃ¸ren, Martin  
 public void showAllMembers() {  
 ArrayList<Member> members = e.getMembers();  
 System.out.println("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");  
 for (Member m : members) {  
 printMember(m);  
 System.out.println("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");   
 }  
 System.out.println("\n");  
 membersMenu();  
 }  
   
 //Bertram, SÃ¸ren, Martin  
 public void showMember() {  
 System.out.print("Insert CPR: ");  
 console.nextLine();  
 String cprInput = console.nextLine();  
   
 Member m = e.getMember(cprInput);  
 if (m == null) {  
 System.out.println("failed to find a member from cpr (" + cprInput + ")!");  
 showMemberMenu();  
 }  
 System.out.println("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");  
 printMember(m);  
 System.out.println("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");   
 }  
   
 //Bertram, SÃ¸ren, Martin  
 public void printMember(Member m) {   
 System.out.println(m.getFirstName() + " " + m.getLastName());  
 System.out.println("CPR: " + m.getCpr());  
 System.out.println("Phone number: " + m.getPhone());  
 System.out.println("Address: " + m.getAddress());  
 System.out.println("Balance: " + m.getBalance());  
 if (m.getIsActive()) {  
 System.out.println("Passive membership");  
 }   
 else {  
 System.out.println("Active membership");  
 }  
 if (m.getIsCompetitive()) {  
 CompetitiveMember cm = e.castCompetitiveMember(m);  
 System.out.println("Team: " + cm.getTeam().getTeamName());  
 System.out.print("Disciplines: ");  
 ArrayList<Discipline> disciplines = cm.getDisciplines();  
 System.out.print(formatActiveDiscplines(disciplines));   
 }  
 }  
   
 //Bertram  
 public String formatActiveDiscplines(ArrayList<Discipline> disciplines) {  
 String activeDisciplines = "none";  
 int arrayEndIndex = disciplines.size()-1;  
 if (arrayEndIndex != -1) {  
 //use fenceposting to seperate each discipline from the arraylist with a comma  
 activeDisciplines = "";  
 for (int i = 0; i < arrayEndIndex; i++) {  
 activeDisciplines += disciplines.get(i).getName() + ", ";  
 }  
 activeDisciplines += disciplines.get(arrayEndIndex).getName();  
 }  
 return activeDisciplines;  
 }  
   
 //Bertram  
 public void recordMenu() {  
 System.out.println("\n\nCompetitive menu");  
 System.out.println("1. Training records");  
 System.out.println("2. Competition records");  
 System.out.println("3. Teams");  
 System.out.println("\n0. Back");  
 System.out.print("\nSelect: ");  
 int input = console.nextInt();  
 console.nextLine();  
 switch (input) {  
 case 1:  
 personalRecordsMenu();  
 break;  
 case 2:  
 competitionRecordsMenu();  
 break;  
 case 3:  
 teamMenu();  
 break;  
 case 0:  
 //back to main menu  
 break;  
 }  
 }  
   
 //Bertram  
 public void personalRecordsMenu() {  
 System.out.println("\n\nTraining records menu");  
 System.out.println("1. Create new");  
 System.out.println("2. View records");  
 System.out.println("3. Edit record");  
 System.out.println("4. Delete record");  
 System.out.println("\n0. Back");  
 System.out.print("\nSelect: ");  
 int input = console.nextInt();  
 console.nextLine();  
 switch (input) {  
 case 1:  
 //create  
 createRecord(false);  
 recordMenu();  
 break;  
 case 2:  
 //read  
 System.out.println("\n");  
 viewRecords();  
 recordMenu();  
 break;  
 case 3:  
 updateRecord();  
 recordMenu();  
 break;  
 case 4:  
 //delete  
 deleteRecord();  
   
 break;  
 case 0:  
 //back  
 recordMenu();  
 break;  
 }  
 }  
   
 //Bertram  
 public void competitionRecordsMenu() {  
 System.out.println("\n\nCompetition records menu");  
 System.out.println("1. Create new");  
 System.out.println("2. View records");  
 System.out.println("3. Edit record");  
 System.out.println("4. Delete record");  
 System.out.println("\n0. Back");  
 System.out.print("\nSelect: ");  
 int input = console.nextInt();  
 console.nextLine();  
 switch (input) {  
 case 1:  
 //create  
 createRecord(true);  
 competitionRecordsMenu();  
 break;  
 case 2:  
 //read  
 System.out.println("\n");  
 printRecords(true, e.getCompetitionRecords(), false);  
 competitionRecordsMenu();  
 break;  
 case 3:  
 //update  
 updateRecord();  
 recordMenu();  
 break;  
 case 4:  
 //delete  
 deleteRecord();   
 break;  
 case 0:  
 //back  
 recordMenu();  
 break;  
 }  
 }  
   
 //Bertram  
 public void createRecord(boolean fromCompetition){  
 //generate the next id by getting the last saved id from the full record arraylist and add 1  
 int id = e.generateNextId(e.getRecords().get(e.getRecords().size()-1).toString());  
   
 System.out.print("CPR: ");  
 String cpr = console.nextLine();  
 CompetitiveMember member = (CompetitiveMember)e.getMember(cpr);  
 if (member == null) {  
 System.out.println("Failed to find a member from CPR (" + cpr + ")!");  
 createRecord(fromCompetition);  
 }  
 else if (!member.getIsCompetitive()) {  
 System.out.println("Member is not competitive!");  
 createRecord(fromCompetition);  
 }  
 ArrayList<Discipline> disciplines = e.castCompetitiveMember(member).getDisciplines();  
 int i = 1;  
 for (Discipline d : disciplines) {  
 System.out.println(i + ". " + d.getName());  
 i++;  
 }  
 System.out.print("Select a discipline: ");  
 int disciplineId = console.nextInt()-1;  
 System.out.print("Time: ");  
 double time = console.nextDouble();  
 //for competitive records  
 String competitionName = "";  
 int placement = -1;  
 if (fromCompetition) {  
 System.out.println("Competition: ");  
 System.out.println("1. Enter");  
 System.out.println("2. Same as last entry");  
 console.nextLine();  
 System.out.print("Select: ");  
 if (console.nextInt() == 2) {  
 competitionName = e.getLastCompetitiveRecord().getCompetitionName();  
 }  
 else {  
 System.out.print("Competition name: ");  
 console.nextLine();  
 competitionName = console.nextLine();  
 }  
 System.out.print("Placement: ");  
 placement = console.nextInt();  
 }  
   
 System.out.println("Date");  
 String dateStr = selectDate();  
 //due to the record constructor parsing a string to a date, we have to try-catch or throw an exception  
 try {  
 if (!fromCompetition) {  
 Record r = new Record(id, member, disciplines.get(disciplineId), time, dateStr, false);  
 e.addRecord(r);  
 System.out.println("Added new time from " + r.getMember().getFirstName() + " " + r.getMember().getLastName());  
 }  
 else {  
 CompetitionRecord r = new CompetitionRecord(id, member, disciplines.get(disciplineId), time, dateStr, true, competitionName, placement);  
 e.addRecord(r);  
 System.out.println("Added tournament time from " + r.getMember().getFirstName() + " " + r.getMember().getLastName());  
 }  
 e.saveRecords();  
 }  
 catch (Exception error) {  
 System.out.println("Failed to create object");  
 }  
 }  
 public void viewRecords() {  
 System.out.println("\n\nView records");  
 System.out.println("1. View best from member");  
 System.out.println("2. View all from member");  
 System.out.println("3. View all records");  
 System.out.println("\n0. Back\n");  
 System.out.print("Select: ");  
 int input = console.nextInt();  
 CompetitiveMember cm = null;  
 String cpr = "";  
 console.nextLine();  
 switch (input) {  
 case 1:  
 //View best from member  
 System.out.print("CPR: ");  
 cpr = console.nextLine();  
 cm = (CompetitiveMember)e.getMember(cpr);  
 System.out.println(cm.getFirstName() + " " + cm.getLastName() + "\n");  
 printRecords(false, e.getPersonalRecordsFromMember(cm, true), true);  
 break;  
 case 2:  
 //View all from member  
 System.out.print("CPR: ");  
 cpr = console.nextLine();  
 cm = (CompetitiveMember)e.getMember(cpr);  
 System.out.println(cm.getFirstName() + " " + cm.getLastName() + "\n");  
 printRecords(false, e.getPersonalRecordsFromMember(cm, false), true);  
 break;  
 case 3:  
 //View all records  
 printRecords(false, e.getPersonalRecordsFromAll(), false);  
 break;  
 case 0:  
 //back  
 break;  
 }  
 }  
   
 //Bertram  
 public void printRecords(boolean fromCompetition, ArrayList<Record> records, boolean singleMember) {  
 SimpleDateFormat dateFormat = new SimpleDateFormat("d. MMM yyyy");  
 for (Record r : records) {  
 System.out.println("Record ID: " + r.getId());  
 //if we are only printing records from one member, don't repeat the name over and over  
 if (!singleMember) {  
 System.out.println("Member: " + r.getMember().getFirstName() + " " + r.getMember().getLastName());  
 }  
 if (r.isFromCompetition()) {  
 CompetitionRecord cr = (CompetitionRecord)r;  
 cr.getCompetitionName();  
 System.out.println("Competition name: " + cr.getCompetitionName());  
 System.out.println("Placement: " + cr.getPlacement());  
 }  
 System.out.println("Discipline: " + r.getDiscipline().getName());  
 System.out.printf("Time: %.2f\n ", r.getTime());  
 System.out.println("Date: " + dateFormat.format(r.getDate()) + "\n");  
 }  
 }  
  
 //Bertram  
 public void updateRecord() {  
 listRecords();  
 System.out.print("Select a record ID to edit: ");  
 int input = console.nextInt();  
 console.nextLine();  
 Record r = e.getRecordById(input);  
 if (r == null) {  
 System.out.println("Failed to find a personal record from ID (" + input + ")");  
 updateRecord();  
 }  
 System.out.println("Select an attribute to edit");  
 System.out.println("1. Member");  
 System.out.println("2. Time");  
 System.out.println("3. Discipline");  
 System.out.println("4. Date");  
 if (r.isFromCompetition()) {  
 System.out.println("5. Competition name");  
 System.out.println("6. Placement");  
 }  
 System.out.println("\n0. Back");  
 input = console.nextInt();  
 console.nextLine();  
 switch (input) {  
 case 1:  
 System.out.print("New CPR: ");  
 String cpr = console.nextLine();  
 Member member = e.getMember(cpr);  
 if (member == null) {  
 System.out.println("Failed to find a member from CPR (" + cpr + ")!");  
 updateRecord();  
 }  
 r.setMember((CompetitiveMember)member);  
 break;  
 case 2:  
 System.out.print("New time: ");  
 double time = console.nextDouble();  
 r.setTime(time);  
 break;  
 case 3:  
 System.out.println("Select discipline:");  
 //only list disciplines that the member is active in  
 ArrayList<Discipline> disciplines = e.castCompetitiveMember(r.getMember()).getDisciplines();  
 int i = 1;  
 for (Discipline d : disciplines) {  
 System.out.println(i + ". " + d.getName());  
 i++;  
 }  
 input = console.nextInt();  
 Discipline discipline = disciplines.get(input-1);  
 r.setDiscipline(discipline);  
 break;  
 case 4:  
 System.out.println("New date:");  
 try {  
 SimpleDateFormat dateFormat = new SimpleDateFormat("dd-M-yyyy");  
 r.setDate(dateFormat.parse(selectDate()));  
 }  
 catch (ParseException error) {  
 System.out.println("Failed to parse date");  
 }  
 break;  
 case 5:  
 CompetitionRecord cr1 = (CompetitionRecord)r;  
 System.out.print("Competition name: ");  
 String name = console.nextLine();  
 cr1.setCompetitionName(name);  
 break;  
 case 6:  
 CompetitionRecord cr2 = (CompetitionRecord)r;  
 System.out.print("Placement: ");  
 input = console.nextInt();  
 console.nextLine();  
 cr2.setPlacement(input);  
 case 0:  
 personalRecordsMenu();  
 break;  
 }  
 try {  
 e.saveRecords();  
 }  
 catch (IOException error) {  
 System.out.println("Could not save changes to file");  
 }  
 }  
  
 //Bertram  
 public void deleteRecord() {  
 listRecords();  
 System.out.print("Select a record ID to delete: ");  
 int input = console.nextInt();  
 Record r = e.getRecordById(input);  
 e.deleteRecord(input);  
 System.out.println("\nDeleted ID: " + input + " by " + r.getMember().getFirstName() + " " + r.getMember().getLastName() + "\n");  
 try {  
 e.saveRecords();  
 }   
 catch (IOException error) {  
 System.out.println("Could not save changes to file");  
 }  
 }  
   
 //Bertram  
 public void listRecords() {  
 SimpleDateFormat dateFormat = new SimpleDateFormat("dd-M-yyyy");  
 for (Record r : e.getRecords()) {  
 System.out.print("ID: " + r.getId() + ", Member: " + r.getMember().getFirstName() + " " + r.getMember().getLastName() + ", Date: " + dateFormat.format(r.getDate()) + ", Discipline: " + r.getDiscipline().getName());  
 System.out.printf(", Time: %.2f\n", r.getTime());  
 }  
 }  
   
 //Bertram  
 public void teamMenu() {  
 System.out.println("\n\nTeam menu");  
 ArrayList<Team> teams = e.getTeams();  
 for (int i = 1; i <= teams.size(); i++) {  
 System.out.println(i + ". " + teams.get(i-1).getTeamName());  
 }  
 System.out.println("\n0. Back");  
 System.out.print("\nSelect team: ");  
 int input = console.nextInt()-1;  
 if (input == -1) {  
 recordMenu();  
 }   
 Team team = teams.get(input);  
 teamSubMenu(team);  
 }  
 public void teamSubMenu(Team team) {  
 System.out.println("\n\nTeam: " + team.getTeamName());  
 System.out.println("Trainer: " + team.getTrainerName());  
 System.out.println("1. View members");  
 System.out.println("2. Select top 5");  
 System.out.println("\n0. Back\n");  
 System.out.print("Select: ");  
 int input = console.nextInt();  
 switch (input) {  
 case 1:  
 //view members  
 teamShowMembers(team);  
 teamSubMenu(team);  
 break;  
 case 2:  
 //select top5  
 teamGetTop5(team);  
 teamSubMenu(team);  
 case 0:  
 recordMenu();  
 break;  
 default:  
 System.out.println("Wrong input!");  
 }  
 recordMenu();  
 }  
   
 //Bertram  
 public void teamShowMembers(Team team) {  
 System.out.println();  
 ArrayList<CompetitiveMember> teamMembers = e.getTeamMembers(team);  
 for (CompetitiveMember tm : teamMembers) {  
 if (tm.getIsActive()) {  
 System.out.println(tm.getFirstName() + " " + tm.getLastName());  
 System.out.println("CPR: " + tm.getCpr());  
 try {  
 System.out.println("Age: " + tm.getAge());  
 } catch (Exception error) {  
 System.out.println("Something went wrong when retrieving the age");   
 }  
 System.out.println("Disciplines: " + formatActiveDiscplines(tm.getDisciplines()) + "\n");  
 }  
 }  
 }  
   
 //Bertram  
 public void teamGetTop5(Team team) {  
 System.out.println("\n\nDiscipline");  
 ArrayList<Discipline> disciplines = e.getDisciplines();  
 for (int i = 1; i <= disciplines.size(); i++) {  
 System.out.println(i + ". " + disciplines.get(i-1).getName());  
 }  
 System.out.println("\n0. Back\n");  
 System.out.print("Select: " );  
 int input = console.nextInt()-1;  
 if (input == -1) {  
 teamSubMenu(team);  
 }   
 ArrayList<Record> top5records = e.getTop5RecordsByDiscplineAndTeam(disciplines.get(input), team);  
 System.out.println("Top 5 in " + disciplines.get(input).getName() + ":\n");  
 for (int i = 0; i < top5records.size(); i++){  
 Record r = top5records.get(i);  
 System.out.println(r.getMember().getFirstName() + " " + r.getMember().getLastName());  
 System.out.println("CPR: " + r.getMember().getCpr());  
 System.out.printf("Best time: %.2f\n\n", r.getTime());  
 }  
 }  
   
 //Leivur  
 public void economyMenu() {  
 System.out.println("Economy menu");  
 System.out.println("1. Show members in debt");  
 System.out.println("2. Show prices");  
 System.out.println("3. Show total member account balance");  
 System.out.println("4. Show income from member subscriptions\n");  
 System.out.println("0. Back\n");  
 System.out.print("Select: ");  
 int input = console.nextInt();  
 switch (input) {  
 case 1:  
 printDebt();  
 economyMenu();  
 break;  
 case 2:  
 printPrice();  
 economyMenu();  
 break;  
 case 3:  
 totalBalance();  
 economyMenu();  
 break;  
 case 4:  
 totalIncome();  
 economyMenu();  
 break;  
 case 0:  
 mainMenu();  
 break;  
 default:  
 System.out.println("There is no menu with that number. Try again.\n\n");  
 economyMenu();  
 break;  
 }  
 }  
   
 //Leivur  
 public void printDebt() {  
 ArrayList<Member> members = e.getMembers();  
 System.out.println("Members in debt: ");  
 for (Member m : members) {  
 if (m.getBalance() < 0) {  
 System.out.println(m.getFirstName() + " " + m.getLastName());  
 System.out.println("CPR: " + m.getCpr());  
 System.out.println("Balance: " + m.getBalance() + "Kr");  
 System.out.println();  
 }  
 }   
 }  
   
 //Leivur  
 public void printPrice() {   
 System.out.println("The prices for membership are: ");  
 System.out.println("Youth svimmers 1000kr per year (age<18)");  
 System.out.println("Senior svimmers 1600kr per year (age>=18)");  
 System.out.println("Elderly 1200kr per year (age>60)");  
 System.out.println("Non-active members pay 500kr per year for a passive membership");  
 System.out.println();  
 }  
   
 //Leivur   
 public void totalBalance() {  
 ArrayList<Member> members = e.getMembers();  
 System.out.print("Total balance on member accounts is: ");  
 double income = 0.0;  
 for (Member m : members) {  
 if (m.getBalance() > 0) {  
 income +=m.getBalance();  
 }  
 }  
 System.out.println(income+"Kr");  
 System.out.println();  
 }  
   
 //Leivur   
 public void totalIncome() {  
 ArrayList<Member> members = e.getMembers();  
 double income = 0.0;  
   
 for (Member m : members) {  
 income += m.getSubscription().getPrice();  
 }  
 System.out.println("Annual income from subcriptions is: "+income+" Kr");  
 System.out.println();  
   
 }  
 //Bertram  
 public String selectDate() {  
 System.out.println("1. Today");  
 System.out.println("2. Manually enter");  
 System.out.print("Select: ");  
 SimpleDateFormat dateFormat = new SimpleDateFormat("dd-M-yyyy");  
 Date date = new Date();  
 String dateStr = dateFormat.format(date);  
 if (console.nextInt() == 2) {  
 System.out.println("Example: " + dateFormat.format(date));  
 System.out.print("Date: ");  
 dateStr = console.next();  
 }  
 return dateStr;  
 }  
   
 //Bertram, SÃ¸ren, Martin  
 public ArrayList<Discipline> selectDisciplines() {  
 ArrayList<Discipline> activeDisciplines = new ArrayList<>();  
 ArrayList<Discipline> disciplines = e.getDisciplines();  
 int count = 1;  
 String disciplineStr = "";  
 System.out.println("Select active disciplines: ");  
 for (Discipline d : disciplines) {  
 System.out.print(count + ". " + d.getName() + " (y/n) ");  
 if (console.nextLine().equalsIgnoreCase("y")){  
 disciplineStr += d.getId();  
 }  
 count++;  
 }  
 for (int i = 0; i < disciplineStr.length(); i++) {  
 activeDisciplines.add(disciplines.get(Character.getNumericValue(disciplineStr.charAt(i))));  
 }  
 return activeDisciplines;  
 }  
}