DALARNA UNIVERSITY

Programming assignment (3 credits)

Examination of this part consist of a mini-project. It includes an **individual programming hand-in assignment** containing all the parts referred to in this course apart from "R", which is excluded here.

You may not copy someone else's solution or parts of it and you may not allow someone else to copy your solution or parts of it. The fact that you send in a solution also confirms that you have read and understand the rules that apply to cheating and plagiarism, see
http://www.du.se/globalassets/local/utbildning/study-at-du/the-academic-honour-principle.pdf
Your solution might be checked against other solutions.

You shall develop an object-oriented program in a mini project.

To pass this module with grade G you have to,

- develop a program in which you solve the *programming task* marked **[G]**
  - the program must work when teacher runs it
- write a project report in which you explain your solution

For grade G you have possibility to do ***one*** correction of your mini project.

To get the grade VG (pass with distinction) you have to

- fulfill the points above for grade G
- implement all of the programming tasks in the part below for VG and they need to work when teacher runs the program
- **not** need to do a correction after that you have handed in the project

Due date for hand in is **Monday 21st of March**

Hand in your files, **as a group hand-in**, containing the project report and **.py** files! in a zip file named:
YourGroupNumber_project.zip example: Group54_project.zip

Hand in the zip file in Learn, see here: "Assignments --> Python project"
The **zip file** shall contain the **.py files and** your project **report**.
Note: *Before submitting the zip file, verify the zip file to make sure it contains all the necessary files and that document(s) can be opened and read. Also check that the .py files can be executed where applicable. Do this by extracting the files from the zip file and verifying as above.*

The project report

The report shall have a good structure and *must* at least consist of the following:

- **Title** page with,
    - the title
    - name of authors
    - date when the report was finished
- **Table of contents**
- An **Introduction**, in which you with your own words describe what you have been working with.
- A **Description** of the solution with classes defined and used.
    - For each class created,
        - explain the purpose and responsibilities of it and the collaborations with other classes
- A **User guide** on how to run the program

The report for the mini-project has to be in English.

If any of the parts are missing the report is considered to be incomplete

The program

You shall create an object-oriented program where it is possible to register and administrate hockey teams in a youth hockey cup. This program will keep track of all the teams that a user of the program creates and administers. Please note that the intention is *not* to keep track of all matches played or results. So only the teams that are suppose to participate in the event with their related data. Every team in the program shall be represented as an *object*, an instance, of the class "**Team**".

Functionality, it must be possible to:

- **create** new teams
- **read** a specific team using *unique id* to show its information
- **update** *fields* for a selected team
- **delete** a team
- **list teams**, list all girls teams or boys teams registered
- **list all teams** all teams registered regardless of girls or boys

create
read
update
delete

also known as:

CRUD operations

Your solution must at least consist of these classes, including the "main" program,

- one class that has the name **Team**, which shall represent a single team
- one class that represent the ***user interface*** with its menu system
    - The user interface shall *be row-oriented* and executed in the command shell. The user interface shall consist of different menu options. The menu options shall be possibly to use in an arbitrary order. One option has to be for quitting the program. The user interface must be easy to use for a person that is not familiar with the program.
- the main program, executing program, uses the classes above
    - main program could also be the user interface mentioned above

If you do not know what a menu system is, you can look in chapter 5 in your textbook.

Note: The *classes has to be in separate .py - files*.

Example: "Team" will be in it's own .py-file as well as the "user interface" in it's own .py-file.

The solution is up to you to design and implement during the work with the programming tasks ☺
If you feel that more classes are needed, it's ok to add these classes. Your solution has to follow object oriented principles. You have to comment your code, i.e. for each class and all methods.
Tasks marked [G] a) and b) must be solved to pass.

DALARNA UNIVERSITY

Python- and R-programming 7.5 HEC

Programming tasks

For G

Programming task

[G]

Both item a) and b) must be implemented and work correctly when teacher runs the program

a) Design and implement the class "Team". The class shall have separate private instance fields for:

	Datatypes
- **id**, a unique value for each object created	[integer]
- **date**, when team was created	[date]
- **name** of team	[String]
- **type**, whether the team is for girls or boys	[String]
- **fee paid**, if fee has been paid or not to participate in the event	[Boolean]
- a fee variable that holds the fee that each team must pay to participate in the event	[*you decide*]

Note: The "id" and "date" attributes should be set *automatically* when an instance from Team class is created. Therefore, the user running the program *should not* set these. In addition, after creation of an object, no manipulation (changing values) should be possible of these two attributes.

b) Add a menu item that makes it possible for a user of the program to:
- create
  - a menu option to add new teams, *instances* of class Team
  - id and date has to be created automatically when the instance is created
  - store teams in an appropriate data structure
- read individual teams
  - a menu option for selecting a team, based on the *id*, and show all information about the selected team in a clear way
  - make sure it's easy for the user to choose a team to be presented
- read teams, boys or girls
  - a menu option to list all teams for boys *or* girls and show all information about the teams in a clear way
- read all teams
  - a menu selection to list all teams, regardless of team type, and display all information about the teams in a clear way
- update
  - update the fields of a team from a particular menu option, *except* the *id and date*
  - make sure it's easy for the user to choose an existing team in the program
- delete
  - delete a selected team by a particular menu option
  - make sure it's easy for the user to choose a team
- exit,
  - menu option to exit the program

DALARNA UNIVERSITY

Python- and R-programming 7.5 HEC

For VG

All of the programming tasks below must be implemented and work when teacher runs the program.

Programming tasks

Modify your program so it includes the following

By using one menu option

- show how many teams that exists, at the moment
- how many percent of all teams that has paid their fee

If a team has cancelled their participation, show this by

- adding a date, a cancellation date in class Team, when a team announces that they cannot attend the event, format: YYYY-MM-DD
    a. the date has to be stored in the object for the actual team
    b. you decide yourself how this should be implemented with regard to how this date should be stated

File handling

By using one menu option

- store the information about all the teams to a text file. It has to be in a text file with file extension **.txt**

By using one menu option

- restore the content of all the teams to the program, with help of the *same* text file

Good Luck!