Examination Module 3

Examination of this module is made through a mini-project. It consists of an **individual programming hand-in assignment** containing all the parts referred to in this module and earlier modules.

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 of the mini project, confirms also 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 a object oriented program in a mini project. There are 10 programming tasks. To pass this module you have to,

- develop an program in which you at least solve *Programming tasks 1 - 5*, and
- fulfill the criteria for grade G, see Study Plan for Module #3, Examination, and
- write a project report in which you explain your solution, and
- get at least 30 points on the examination, and
- hand in your project before 8th of November at 23:59, Central European Time (CET/CEST).

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

To get the grade VG for this module you have to

- fulfill the points above for grade G and,
- fulfill the criteria for grade VG, see Study Plan for Module #3, Examination, and
- get at least 40 points on the examination, and
- hand in your project before 4th of November at 23:59, Central European Time (CET/CEST), and
- **not** need to do a correction after that you have handed in the project.

If you fail this module you have a possibility to re-register to next course occurrence and do the mini project for that course occurrence.

Hand in the complete folder for the Netbeans project and the project report in a zipfile named: *YourUsername_ExamModule3.zip*. Hand in the zipfile in Learn to the map: Assignments > CM3: Hand-In Assignment (Mini project). The zipfile shall contain the Netbeans project and your project report. Before you send in your zip-file you should unpack it and check that it contains all what you want to send in.

After that you have handed in your mini project, please fill in the Student Reflections: Course Module 3, and hand it in to: Assignments > CM3: Reflection Form.

Project report, 15p

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

- Title page, with the title of the report, name of the author, and the date when the report was finished. 1p
- A table of the report contents. 1p
- An introduction, in which you with your own words describe the problem that you have been working with. 2p
- A description of your solution. In this part you shall explain briefly your solution with help of an UML class-diagram that show the relationships between the classes you have defined. The diagram shall be comment and explained in your text. 4p
- For each class that you have defined you have to explain the purpose of it and describe its responsibilities and collaborations with other classes. 2p
- A user guide of how to run the program. 3p
- Comments of your work. This part shall contain your comments about the work that you have done. 2p

The report for the mini-project has to be written in English. A report in any other language will not be accepted.

The program

You shall create an object oriented program that work as **your own personal computer program to keep order of the teams participating in a sports event**. The purpose of the program is to give you help to keep track of all the teams. Every team in the program shall be represented by its own object, as an instance of the class **Team**.

It shall be possible to add new teams to the event and it shall be possible to select a team by its name and get the information about it and also it must be possible to update the information for a selected team. Finally, it shall be possible to remove a team.

Your solution shall consist of at least two classes, one class that has the name **Team**, which shall represent a single team and one class that represent the user interface with its menu system.

The user interface shall *be row-oriented* and execute in the command shell, the Command Prompt. 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.

If you don’t know what a row-oriented menu system is, you can look at an exercise in your text book. You find that exercise at the end of the chapter "A Second Look at Classes and Objects". Go to the section "Programming Challenges", and study exercise "9. Geometry Calculator". In the exercise a menu system has to be constructed that shall run in the Command Prompt.

The solution is up to you to design and implement during the work with the programming tasks. If you are missing classes you are free to add these classes. Your solution has to follow object oriented principles. Every class and all methods must be comment. You are allowed to use all tools that Netbeans has. You have to solve task 1 – 5. Which of the tasks 6 – 9 that you decide to solve, and in what order you solve them is up to you. I advise you to try to solve some more tasks than the first five; otherwise it will not be sure that you reach the limit for grade G.

Note that you are not allowed to use any third-party software libraries; you are limited to only use Java SE.

Programming task 1, 3p

Design and implement the class, Team. The class shall have separate instance fields for:

- The name of the team
- A short note about the team
- If the team paid the entry fee or not
- Entry fee
- The number of teammembers

All fields shall be impossible to reach outside the class Team. Object of the class shall not accept unrealistic or erroneous data, e.g. a name can't consist of only blank spaces and a value can't be less than zero. **An object of the type Team shall not allow** changing its name after the object has been created. The other fields of the object shall be possible to change.

Programming task 2, 3p

Apart from the menu option to exit the program, you must add a menu item that makes it possible for a user of the program to add new instances of the class Team to the program. The fields of a team, except for the name, shall be possible to update after the team is added to the program, see task 4.

Programming task 3, 3p

It also has to be a menu option for selecting a team. All information from the selected team shall be displayed in a clear way.

Programming task 4, 3p

Add a menu option which makes it possible to update the note, fee and value of teammembers.

Programming task 5, 3p

It has to be possible to remove a selected team from the program by a particular menu option.

Programming task 6, 3p

Modify your program so it shows how many teams that is stored for the moment and the total sum of all the fee values. It is up to you if the information shall be shown by choosing extra menu option or if that information shall be shown all the time.

Programming task 7, 3p

Modify the program so it shows how many percent of all the teams that still have an unpaid fee. The percent can be shown together with the information from programming task 6 if you have solved that task; else you have to decide by yourself how the percent has to be shown.

Programming task 8, 4p

Modify the program so that it is possible to add the date team was registered when it is created in the program. The date must be realistic. Dates that cannot exist or exists in the future shall not be allowed.

Programming task 9, 4p

Modify the program so it becomes possible to see how long ago, in number of days, the team was enrolled. The age shall be calculated with help of the registered day that is stored in the object, see task 8.

Programming task 10, 6p

It shall be possible to write out the information from all the teams to a text file before you quit the program execution. It shall also be possible to restore the content of all the teams to the program with help of the text file, when you start the program.

Don’t forget that it take time to write the report. Begin with the report in a good time. Save the trickiest tasks that you want to solve to after that you have written the report. Update the report if you succeed with those tasks. The report must correspond to the solution that you hand in.

Good Luck!