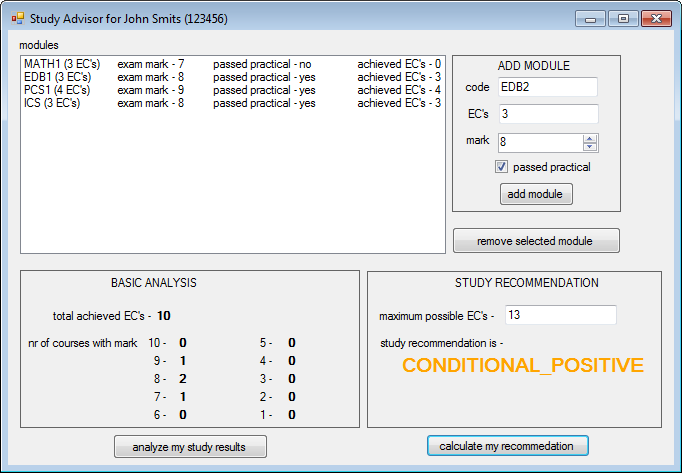
# Exam PCS2 – My Study Advisor January 2014

You are asked to develop an application that you can use to analyze your study results. Here you can enter information of every module: code (e.g., PCS1, EDB1, etc.), module EC’s, exam mark and whether you passed the practical or not. After you entered information about all modules of one block, the program will calculate several study performance indicators for you: the total EC’s that you achieved, the number of modules you passed with a certain mark (marks 1, 2, 3, …, 10), and the study recommendation.



Now open the startup-project. You can see that there is one form with a lot of controls on it (see the picture above). During this exam, you will add two classes: class Module and class Student. Of course, you will also add some code to the Form as well.

**Important note:** whenever this exam paper suggests to create a variable, property or method with a certain name (e.g., property called “ExamMark”), you are required to indeed use that name (e.g., “ExamMark”) in your program.

Exam points:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Assignment | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Bonus | Total |
| Points | 20 | 10 | 10 | 10 | 10 | 15 | 15 | 10 | 100 |

A**ssignment 1: Class Module (5 + 5 + 5 + 5 pts)**Add a class called Module to the project (in a separate file). This class will represent one module (e.g., PCS1 or EDB1 or MATH1…). Implement this class by doing the following:

1. Add all properties listed below. Wherever possible make an auto-implemented property. If it is not possible to make an auto-implemented property, you can define an instance variable with a ‘plain’ property (where you implement set and get accessors).
   * Property called **Code** should store the code of one module (as a string).
   * Property called **EC** should store the standard number of EC’s for this course (as an integer number). This property may only store values higher than zero.
   * Property called **ExamMark** should store the exam mark of the module (as an integer number) . It can only store values in the range 1, 2, 3, …, 9, 10.
   * Property **PassedPractical** should store the information about whether the practical of the module is passed or not (use type bool).
2. Add one constructor to the Module class. This constructor should have parameters to initialize values of all properties.
3. Add method called **GetAchievedEC**. This method should return:
   * The value of property EC if the exam mark is sufficient (i.e., greater than 5) and the practical is passed.
   * Zero (0) if the exam mark is insufficient (i.e., smaller than 6) or the practical is not passed.
4. Add method **AsString()** which returns the information about the module as one string. Use a format similar to the one shown in the ListBox in the screen shot on the first page. You should use method GetAchievedEC to get the achieved EC’s.

**Assignment 2: Class Student (5 + 5 pts)**

Add a class called Student to the project. This class will represent the information about one student.

1. Add the following properties/fields:

* Property called **Name** should store the full name of the student (as a string).
* Property called **Number** should store the student number (as an integer number).
* Add a field called **modules**, which should be a list (not array) for storing all modules this student is following. This field must be inaccessible for other classes!

1. Add one constructor that initializes all properties/fields. This constructor should have parameters to initialize values of properties Name and Number. The list of modules should not contain any modules at the start.

**The following assignments are about the form.**

All controls (buttons, labels) you need are already in the form. You don’t need to add any new controls during this exam.

**Assignment 3: At startup (5 + 5 pts)**

1. Add one field of the type Student to the form. Make sure it is created and initialized on startup with your name and your student number.
2. Set the Title of the form on startup to something like “Study Advisor for John Smiths (123456)”, where you give the name and number of the student you created.

**Assignment 4: Add a module (5 + 5 pts)**

Add functionality to add a new model:

1. Add method **AddModule** to the **Student** class, which will be used to add a new module to the list of modules. This method should use several parameters to indicate all information about the module to be added: code, ec, exam mark and whether the practical is passed or not. In this module you should create a new Module object and add it to the list. The method should return the added (new) module.
2. Add functionality to the **“add module”-button**. Here you should use the values from textBoxCode, textBoxEC, numericUpDownMark and checkBoxPractical to add a new module:

* to the student object (use method AddModule you just added to the Student class), and
* and to the listBoxModule (add the result of method AsString() of the module, so that a nice string is shown in the listBoxModule).

**Assignment 5: Remove a module (4 + 6 pts)**

Add functionality to remove an existing model:

1. Add method **RemoveModule** to the **Student** class, which should remove one module from the list. This method should have one parameter that indicates the index (position in the list) of the module that should be removed. This method does not return anything.
2. Add functionality to the **“remove selected module”-button**. After clicking this button, the module which is currently selected in the list box should be removed:
   * from the student object (use method RemoveModule you just added to the Student class), and
   * from the listBoxModules on the form. Hint: Use the selectedIndex of the ListBox to find out which module is selected. You can then use this index to remove the module both from the student object and from the listBoxModule.

**Assignment 6: Perform the BASIC analysis of study results (5 + 7 + 3 pts)**

Add functionality to perform the basic study analyzis:

1. Add method **GetAchievedEC** to the **Student** class, which should return the sum of achieved ec of all modules in the list (use method GetAchievedEC of class Module).
2. Add method **NrOfModulesWithExamMark** to the **Student** class, which should return the number of modules with one specific exam mark. This specific exam mark should be given as an integer parameter in this method.
3. Add functionality to the **“analyze my study results”-button**. Here you should let the student object calculate eleven numbers, which you should show in the appropriate labels on the form (all bold labels on panel BASIC ANALYSIS).

**The following assignment is about calculating the study recommendation.**

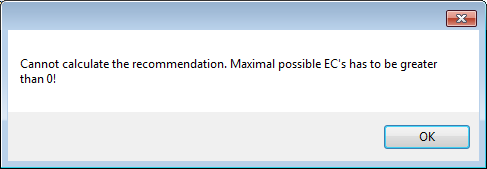
This paragraph explains how study recommendation is calculated in our school. At the end of one school block each student gets a study recommendation. This recommendation is calculated based on the percentage of the achieved EC’s:

1. If the student achieved 100% of the maximal possible EC’s then the recommendation is POSITIVE.
2. If the student achieved more than 75% of the maximal possible EC’s then the recommendation is CONDITIONAL\_POSITIVE.
3. If the student achieved more than 50% of the maximal possible EC’s then the recommendation is CONDITIONAL\_NEGATIVE.
4. If the student achieved less than 51% of the maximal possible EC’s then the recommendation is NEGATIVE.

**Assignment 7: Calculate the STUDY RECOMMENDATION based on study results (3 + 6 + 6 pts)**

In order to add this kind of recommendation to your program you should do the following:

1. Add an **enumeration type** called **Recommendation** to your program. Make sure it contains the four recommendation types explained above.
2. Add method **GetRecommendation** to the Student class. Based on the maximal possible total EC’s and achieved EC’s, this method should return the correct recommendation for this student. The maximal possible total EC’s will be given as a parameter in this method.
3. Extend the implementation of the **“calculate my recommendation”-button** click in such a way that:
   * You get the maximal possible EC’s from the textBoxMaxPossibeEC on the form,
   * Check if the maximal possible EC’s is greater than 0 (zero). If this is not the case, show appropriate error message. For example:



* + If the maximal possible EC’s is greater than 0 (zero), you should get the recommendation from the student object (given the maximal possible EC’s),
  + Show this recommendation in the labelRecommendation, and
  + Set the color of this label (labelRecommendation.ForeColor) to:
    - Color.Green if POSITIVE,
    - Color.Orange if CONDITIONAL\_POSITIVE,
    - Color.Coral if CONDITIONAL\_NEGATIVE, and
    - Color.Red if NEGATIVE.