# PCS3 Exam

**Date: Wednesday 19 October 2016**

**Time: 10:30 – 13:00 h. (150 minutes)**

**Rules:**

**Admitted resources:**

* You are allowed to use everything on paper (books, notes, etc.) and on your laptop, but only what you bring in: you are not allowed to borrow something from someone else.
* During the exam it is not allowed to use the network. You should make the exam yourself: so no communication with msdn or google for help and no communication with other students, like communication by Facebook, e-mail, Skype, gsm or whatever.

**Way of working during the exam and handing in your exam:** see document "Unimpeded testing (Rustig toetsen) Step by Step".

Remark: the displayed remaining time, as shown at "the finish flag of Unimpeded testing", might be wrong.

**Grading: see assignments.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **EXERCISE** | **1** | **2** | **3** | **4** | **5** | **6** |
| **POINTS** | **25** | **25** | **5** | **15** | **15** | **15** |

**Preliminary remarks:**

* Never duplicate code if there is a better way to achieve the same result. 
* Whenever this exam paper suggests to use a certain name for a method, variable or anything else, you are required to indeed use that name.
* If you think you need more class-members, you are allowed to add more members to the classes than specified in this exam.
* In this exam we use the word "app", which is an abbreviation for "application" or "program".

**Introduction: An application for a school.**

In this assignment we ask you to write a C# application for a school. The school organizes a number of courses that can be attended by participants.

The school offers two different types of courses that can be attended:

* classroom courses

A classroom course takes place in a classroom. However, for each classroom course, a minimum number of participants and a maximum number of participants is specified. The specified minimum number of participants should be more than zero. The course will only take place if the actual number of participants is larger than or equal to the minimum number of participants. As there is limited space in a classroom, there is also a maximum number of participants for a classroom course (dependent on the size of the classroom). A participant can only be added to a classroom course, if the total number of participants does not exceed the maximum. Provided a classroom course takes place, then the price per participant is calculated according to the following formula:

pricePerParticipant =

fee + (totalPriceForAllParticipants / numberOfParticipants)

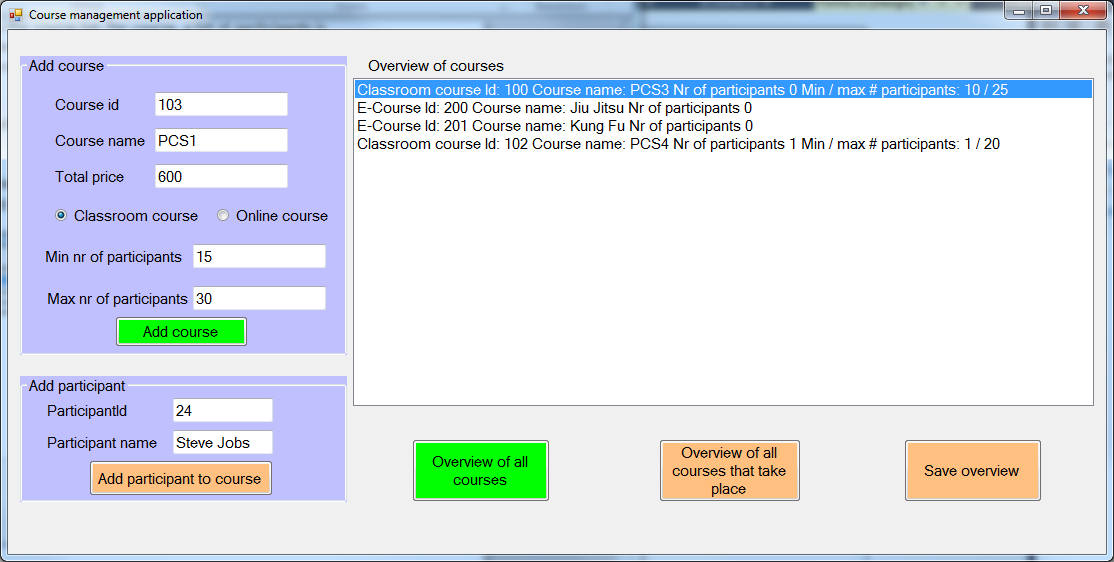
* e-courses

An e-course is an online course. There are no restrictions on the number of participants of an e-course. For an e-course a fixed price per participant is specified. A participant can always be added to an e-course.

With the application the school should be able to manage a list of available courses. It should be possible to add new courses to the course list. Per course, a list of participants is administered. It should be possible to add a participant to a course, and to generate an overview of all courses, of all courses that take place, and to save information about the courses in a file.

In order to help you write this application, some code has already been given in the start-up-project.

The code does NOT compile yet, because some code is missing. You will implement it in the assignments. We put some testing data in it. You are free to use it. Using this testing data could result in the figure below:

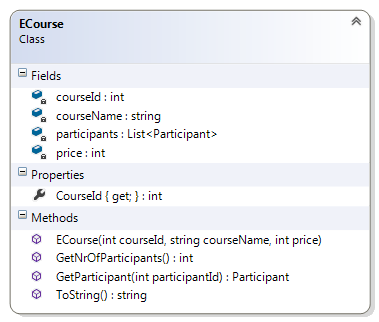
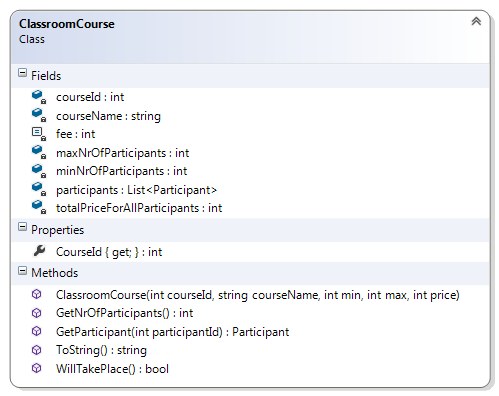


Note that the buttons in the application have two different colours. The button handlers for the green-coloured buttons are already given, and you are asked to provide the implementation of the button handlers for the orange-coloured buttons.

**ASSIGNMENT 1 (20+5 points): The classes Course, ClassroomCourse and ECourse**

In the start-up-project there is already a class ClassroomCourse and a class ECourse, but it is not a good basis for the future. In future there will be more kinds of courses, so it is time to use inheritance.

The class diagram for the current situation is given below (note that some of the methods in the classes below still have to be implemented):



As you can see in the class diagram and the code for these classes, these classes have something in common and something different.

1. Add the class Course to the project. This class Course must be the base class for the classes ClassroomCourse and ECourse (in other words: the classes ClassroomCourse and ECourse must inherit from Course). **Use the benefits of inheritance as much as possible.**
2. Both classes have a method ToString() , which should return a string that
3. First displays the type of course
4. Then shows all relevant information of that object contained in the fields of the object.

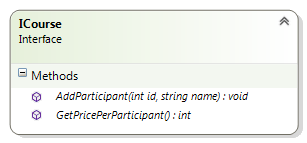
In the listbox in the screenshot on the previous page, some examples are given of the output of the ToString() method.

Implement the ToString() method(s) in the appropriate class(es).

**ASSIGNMENT 2 (5+10+10 points): Interface ICourse**

Below, the class diagram of interface ICourse is given. All subclasses of class Course should have the methods as specified in the interface ICourse.

1. First, we ask you to add the interface class, and have the appropriate class(es) implement the interface, such that the benefits of inheritance are exploited as much as possible.



Now you are asked to implement the methods AddParticipant and GetPricePerParticipant. The implementation is different for a classroom course and an e-course.

1. **The method void AddParticipant(int id, string name).**

Both for an e-course and for a classroom course, a participant can only be added once to a course. When a participant is added to a course with an id (e.g. with id 20) that already occurs in the list of participants, then an exception should be thrown of type CourseException, with message "Participant with id 20 already registered" . The exception class CourseException does not yet exist, you have to implement it yourself.

For a classroom course, there is another restriction: a participant can only be added if the maximum number of participants is not exceeded. In case adding the new participant would exceed the maximum number of participants, then an exception should be thrown of type CourseException, with message "Participation not possible, max nr of participants reached".

Implement the exception class CourseException and make sure that for the different types of courses, the method AddParticipant works as specified above. Again, make sure to use the benefits of inheritance as much as possible.

1. **The method int GetPricePerParticipant().**

For an e-course, the price per participant is just the price specified in the field price.

For a classroom course, the price per participant can only be calculated if the course has at least the minimum number of participants. In that case, the price is calculated by means of the following formula:

pricePerParticipant =

fee + (totalPriceForAllParticipants / numberOfParticipants)

In case the number of participants is less than the minimum (e.g. the number of participants is 4 while the minimum is 10), then the method should return -1.

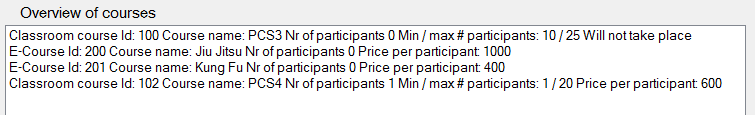
Make sure that for the different types of courses, the method GetPricePerParticipant() works as specified above. Again, make sure to use the benefits of inheritance as much as possible.

**ASSIGNMENT 3 (5 points): Method ToString() again.**

For all courses **that take place**, the return value of the ToString() method(s) should be extended so as to also contain information about the price per participant (showing the result of the GetPricePerParticipant() method).

For courses **that do not take place**, the returned string should be extended with the message “Will not take place”.

Some examples of the extended output of the ToString() method(s) are shown in the screenshot below.



Make it work.

**ASSIGNMENT 4 (15 points): Method btnAddParticipant\_Click.**

In this assignment you are asked to provide the implementation of the event handler of the “Add participant” button. When the button is clicked, for the course that is selected in the list box lbCourseOverview, a participant should be added with the id specified in tbParticipantId and name specified in tbParticipantName. After adding the participant to the course, make sure to show the overview of all courses again.

Implement the event handler method. Make sure to catch all relevant exceptions, and display an appropriate message. Whenever possible, catch exceptions of different types in separate catch blocks.

**ASSIGNMENT 5 (15 points): Method btnOverviewAllThatTakePlace\_Click.**

In this assignment you are asked to provide an implementation of the event handler of the “Overview of all courses that take place” button. When the button is clicked, in the listbox lbCourseOverview, a list should be given with information of all courses that take place. Note that e-courses always take place, and classroom courses take place only when they have at least the minimum number of participants.

**ASSIGNMENT 6 (15 points): Method btnSaveOverview\_Click.**

In this assignment you are asked to provide an implementation of the event handler of the “Save overview” button. When this button is clicked information about all courses that take place should be saved in a text file as follows:

* The first line of the file should contain the name of the school
* The next line should contain the string “\*\*\*\*\* e-courses \*\*\*\*\*”.
* Then, all relevant information about the e-courses should be given.
* Then, add a line with the string “\*\*\*\*\* classroom courses \*\*\*\*\*”.
* Then all relevant information about the classroom courses that take place should be given
* Then, add a line with the string “\*\*\*\*\* revenues \*\*\*\*\*”.
* Then there should be two lines of information:
  + a line stating the total revenue for all e-courses .
  + A line stating the total revenue of all classroom courses that take place.

Note that the revenue for a single course is equal to the

pricePerParticipant\* nr of participants.

----- END OF THE EXAM