CSCI 1301 - Lab 16

Clément Aubert

March 6, 2018

Part 0 - Wrapping Up

The code we just studied in class, slightly expanded and with an application program, is available to download.

Part I - Milestone #2

You accomplished a lot since Milestone #1 (that was presented in Lab 07), so let's take a moment to look back at what you learned.

What You Learned

To Get Better at Using Softwares

- How to have 2 `.cs` files open in one project in VS (Lab 8 - Part I)
- How to "undo" things, using CTRL + z (Lab 8 - Part I)
- How to use a software to compare text files (Lab 12 - Part I)

To Design, Edit and Enhance Classes

- How to draw UML diagrams (Lab 9 - Part II, Lab 10 - Part I)
- That renaming uniformily attribute names was possible (Lab 8 - Part I)
- How to modify a pre-existing class (Lab 8 - Part I)
- How to add methods to your class (Lab 8 - Part I)
- To write your custom constructor (Lab 10 - Part I)
- To write or edit a `ToString` method (Lab 12 - Part III)
- To write static methods (Lab 12 - Part III)
- To write "no-args" constructor (Lab 10 - Part I)

To Use Classes

- To create objects, using the default constructor (Lab 8 - Part I)
- To use setters and getters (Lab 8 - Part I, Lab 12 - Part II)
- To create objects, using custom constructor (Lab 10 - Part I)
- To use `ToString` methods (Lab 12 - Part II)
- To use static methods (Lab 12 - Part II)

Various

- To use constant values (Lab 10 - Part I)
- To convert `int` into `char`, and reciprocally (Lab 14 - Part II)
- To read a single character from the user (Lab 14 - Part II)

To Write Branching Programs

- To understand logical operators (Lab 13 - Part I)
- To understand precedence among operators (Lab 13 - Part II)
- To write simple `if` statements (Lab 14 - Part I)
- To practise writing programs that use nested `if` (Lab 14 - Part III)

To Solve Problems!

- To decompose a problem into smaller pieces (Lab 11 - Part I)
- To enhance pre-existing code (Lab 12 - Part III)
- To write a game! (Lab 13 - Part III)
- To design and implement basic decision-tree (Lab 14 - Part III)
- To implement and test in parallel a complex class (Lab 15 - Part II)

Did You Pushed Further?

- Have a look at properties (Lab 8 - Part II)
- Have a look at the variety of UML diagrams (Lab 9 - Part III)
- Know more about the `Math` class (Lab 10 - Part II)
- Rework previous problems (Lab 8 - Part II, Lab 10 - Part II, Lab 11 - Part II)
- Design a static class (Lab 13 - Part IV)
- Have a first look at `for` statement (Lab 14 - Part IV)
- Discover how strings are ordered (Lab 14 - Part IV)

Part II - Working on Problem-Solving

Download, extract, compile and execute Course.zip. Then, read the code, and make sure you understand everything in it: the role of the static attribute, the difference between the two constructors, the branching in the constructor that takes two arguments. Then, do the following:

a. Write a `ToString` method that returns
    1. The name of the student, followed by
    2. "is registered for section A." of the student is in section A, "is on the waiting list." if the student is on the waiting list.
b. Test your method in the program, printing the information about various students.
c. Your next step will be to open a second section:
    1. We want to have a section B, with 4 seats available. Implement the existence of this section with a static attribute.
    2. The constructor that takes one argument should now assign the student to section A if there is room available, to section B if section A is full but section B isn't, or to the waiting list if both sections are full.
    3. Think about how you should change the constructor that takes two arguments, and implement what seems to be logical. For instance, if a student wants to enroll in section B, but section B is full and section A isn't, should the student go to section A or be on the waiting list?
    4. Finally, correct your `ToString` method, so that the string returned describes accurately the section the student is enrolled in.

Bonus: There is an inconsistency in the constructor that takes two arguments, can you spot it? What will happen if the user gave a code for the section that isn't `'A'` or `'a'`, and section A is full? Can you fix this?