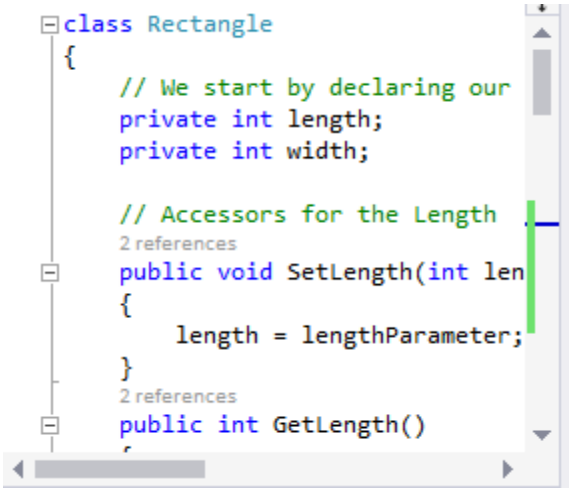
CSCI 1301 - Lab 08

Clément Aubert

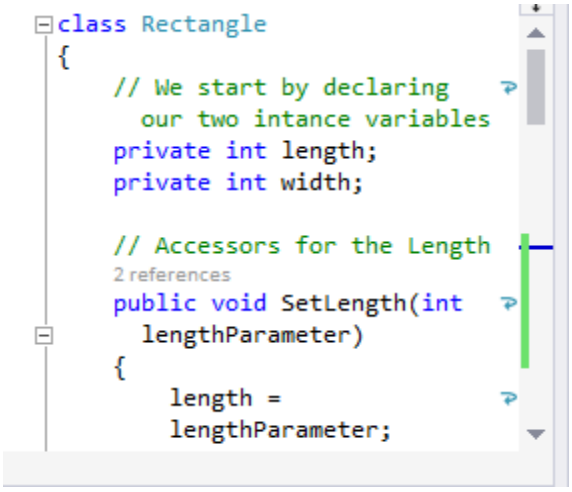
January 30, 2018

Dependencies: This lab has only one part.

Feature of the day: I recommend you activate word-wrap in VS. Refer to https://msdn.microsoft.com/en-us/library/ms165339.aspx (VS 2015) or https://docs.microsoft.com/en-us/visualstudio/ide/reference/how-to-manage-word-wrap-in-the-editor (VS 2017), to go from



to



See the difference? The horizontal scrolling disappeared, every line that is too long is "wrapped", and this is indicated with the ↵ sign.

Part 0 - Project #1 Solution

Make sure you understand the grading scale and my annotations. If you want to look at one possible solution, download this project. Make sure you *extract it* before opening it in VS.

Part I - Using a Pre-Defined Class

Manipulating Two `.cs` Files at a Time

a. Download the Rectangle project, extract it, and open it with VS. Note that in the "Solution Explorer", there are two `cs` files listed: `Program.cs` and `Rectangle.cs`.
b. In the Solution Explorer, double-click on `Rectangle.cs`, and note how close it is from what was presented during the lecture.
c. In the Solution Explorer, double-click on `Program.cs`, and observe it.
d. Compile and execute the code.
e. Now, do the following:

- Introduce a syntactical error in `Program.cs` (e.g., remove a `;`), and try to build the solution: what do you observe? Restore the program to its previous state, using CTRL + z to "undo" your operation.
- Introduce a syntactical error in `Rectangle.cs` (e.g., remove a `;`), and try to build the solution: what do you observe? Undo the modification using CTRL + z.
- Add `length = 12;` in the main method of `Program.cs` and try to build the solution: what do you observe? Undo the modification using CTRL + z.

Enriching `Program.cs`

Edit the `Main` method of `Program.cs` by adding at its end statements that perform the following:

1. Create a second `Rectangle` object and set its length to 3 and its width to 3.
2. Create a third `Rectangle` object, and ask the user to specify its length and width. Print the area of the rectangle whose dimensions were given by the user.
3. Create a fourth `Rectangle` object, do not specify its length or width, and print them. What do you observe?

In the last exercise, you may notice that the length and the width of the newly created object were assigned default values. To know more about this, refer to https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/default-values-table.

Editing `Rectangle.cs`

Edit `Rectangle.cs`:

1. Rename uniformly `lengthParameter` to `lengthP` in the `SetLength` method (that is, replace the two occurrences). Compile and run your program, what do you observe?
2. Some people use the convention of prefixing instance variables with _ (the underscore character), `m` (for "member"), or even `m_`. You will always find someone furiously advocating for one particular convention, the truth is that if you're not forced to use one, you should pick whichever suits you best. Still, just to use it at least once, rename uniformly `width` into `m_width` and see how it feels. Compile and run your program, what do you observe? Either undo this modification, or rename `length` into `m_length` (you have to be consistent!).
3. Change the name of one of the accessor method in `Rectangle.cs` without changing it in `Program.cs`. Compile and run your program, what do you observe? Undo your modification.

Enriching `Rectangle.cs`

By taking inspiration from the `CalculateAre()` method, write three new methods:

a. A method that returns the perimeter of the calling object.
b. A method that double the heigt and the width of the calling object.
c. A method that swap the height and the width of the calling object.

For each method, pick a (valid) name, think about the return type and the parameters, and write the body of the method carefully. After compliation succeed, call that method in `Program.cs` and see if it has the expected behaviour.

Part II (Optional) - Pushing Further

The following are two independent tasks, to widen your understanding of this class, and to prepare you for the next labs.

a. Go back to the problem from the previous lab (that is, Lab 7, Part II). Change your program so that:

- The user can decide how many slices he wants to cut in every pizza,
- The program prints the number of slices per guest *without fraction*, and the number of remaining slices.

For instance, a user entering 4 guests and 2 pizzas to be cut in 8 slices should read that every member of the party will have $\lfloor (2 \times 8)/(4+1) \rfloor = 3$ slices, and that $(2 \times 8) mod (4+1) = 1$ slice will be left.

b. *Properties* are introduced in Section 4.6 of your textbook, and at https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/properties in the documentation. Have a look at both, and open the project `Account2.sln` in the `ch04/Account2/` folder of the source code of the textbook. Properties **will not** be introduced in this class, but if you feel confident enough to use them, feel free to do so.