Introduction to Classes – Rectangle Class

https://csci-1301.github.io/about#authors

August 16, 2021 (08:12:03 PM)

Contents

1 Using a Pre-Defined Class

This lab will guide you in your first manipulation of a programmer-defined class. The last part is challenging; therefore, we provide a possible solution at the end of the page, but make sure you try to solve it by yourself beforehand.

1.1 Manipulating Two `.cs` Files at a Time

1. Download the Rectangle project[1], extract it, and open it with your IDE.
2. Note that in the "Solution Explorer", there are two `cs` files listed: `Program.cs` and `Rectangle.cs`.
3. In the Solution Explorer, double-click on `Rectangle.cs` and note how close it is to what was presented during the lecture.
4. In the Solution Explorer, double-click on `Program.cs` and observe it.
5. Compile and execute the code.
6. 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.
- Introduce a syntactical error in `Rectangle.cs` (e.g., remove a `;`), and try to build the solution: what do you observe? Undo the modification.
- Add `length = 12;` in the main method of `Program.cs` and try to build the solution: what do you observe? Undo the modification.

[1] Rectangle.zip

1.2 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 and width to 3.
2. Create a third `Rectangle` object and ask the user to specify its length and width. Display the area of this rectangle on the screen.
3. Create a fourth `Rectangle` object, do not specify its length or width, and display them on the screen. What do you observe?

In the last part, 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 the documentation on default values of C# types[2].

1.3 Editing `Rectangle.cs`

Edit `Rectangle.cs`:

1. Rename every instance of `lengthParameter` to `lengthP` in the `SetLength` method (that is, replace both occurrences).

   You can use IDE's rename feature to perform this operation. If you are having trouble finding or using it, see the rename guide for your IDE: Visual Studio[3], MonoDevelop[4], Rider[5]

2. Compile and run your program. What do you observe? What happens if you change one instance to `lengthP` while leaving the other as `lengthParameter`? Try it out by manually editing one of these instances and compiling the program. Be sure to change it back after.

3. Some people use the convention of prefixing instance variables with `_` (the underscore character), `m` (for "member"), or even `m_`. You can always find someone furiously advocating for one particular convention, but the truth is that if you're not forced to use one (for example, by the rules of a software company you work for), you should pick whichever suits you best. Still, just to use it at least once, rename every instance of `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!).

4. Change the name of one of the accessor methods in `Rectangle.cs` without changing it in `Program.cs`. Compile and run your program. What do you observe? Undo your modification.

5. What has this section taught you about variable and method names within `.cs` files and across `.cs` files within the same project? What about naming is important to the compiler, and what is only important to the programmer?

1.4 Enriching `Rectangle.cs`

Taking inspiration from the `ComputeArea()` method, write three new methods:

1. A method that returns the perimeter of the calling object.
2. A method that doubles the length and the width of the calling object.
3. A method that swaps the length and the width of the calling object.

[2] https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/default-values-table
[3] https://docs.microsoft.com/en-us/visualstudio/ide/reference/rename?view=vs-2019
[4] https://www.monodevelop.com/documentation/feature-list/refactoring/#rename
[5] https://www.jetbrains.com/help/rider/Refactorings___Rename.html

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

This is more challenging than the rest of the lab, so if you are unable to finish this part during the lab session, do not worry, but take the time to study a possible solution[6] to this problem.

[6] Enriched_Rectangle.zip