Contents

Precise Rectangle

This lab serves multiple goals:

- Learn to edit an existing program containing a class,
- Implement a change of datatype in the attributes of a class,
- Reinforce basic class manipulation,
- Discover how to add a class to an existing project.

Writing Your Own PreciseRectangle Class

In this exercise, you will create your own class instead of using and expanding one that was written for you. The idea is to take inspiration from the class you have already encountered (`Rectangle`) to create a new class, called `PreciseRectangle`, that will manipulate rectangles whose width and length are floating-point values instead of integers (as in `Rectangle`).

This should be a fairly straightforward exercise that mostly reinforces what you have already encountered, but you will be exposed to creating new classes in your IDE for the first time.

Implementation

To implement your class in your IDE, you are given two methods below: you can edit the pre-existing project or you can create a new blank class. It is recommended to pick the one that you feel the most comfortable with initially, and then you should try the other technique. You will need to know how to edit existing projects and how to create new ones.

Edit the Pre-Existing Project

1. Re-download the "Rectangle" project[1], extract it in a folder, and open it with your IDE.
2. Within your IDE, re-name the project to "PreciseRectangle", and re-name the "Rectangle.cs" file to "PreciseRectangle.cs"

[1] https:/princomp.github.io/code/projects/Rectangle.zip

It is important that you rename the files within your IDE. If you try to rename your files, or their folders, outside of the IDE then it will break your solution. The solution will still be looking for the original file/folder names, and will not recognize the changed names. If such an error occurs, restore the previous names and then rename your files through the IDE as instructed.

3. In the "PreciseRectangle.cs" file, replace `class Rectangle` with `class PreciseRectangle`.
4. Comment out the body of the `Main` method in "Program.cs".
5. Your program should compile as it is, but you need to edit `PreciseRectangle.cs` to now store the `width` and the `length` attributes as type `double`, and then you will need to edit the rest of the class accordingly. (e.g., What should the return type of the `GetWidth` method be?)
6. Declare and manipulate precise rectangles (with `double` values for the width and the length) in the `Main` method, and make sure they behave as expected (i.e., Can you compute the area, set attributes, etc.?).
7. Add the missing methods `ComputePerimeter`, `Swap`, and `MultiplyRectangle`, as described in the Rectangle lab[2] but also below.

Starting From Scratch

1. Create a new project in your IDE, and name it "PreciseRectangle".
2. In the Solution Explorer, right-click on "PreciseRectangle", then on "Add..." and select "Class". Then, select "Class" in the dialog box, write "PreciseRectangle.cs" as the name of the file, and click on "Add".
3. You should now have two ".cs" files opened and displayed in the Solution Explorer: "Program.cs" and "PreciseRectangle.cs".
4. Implement the `PreciseRectangle` class according to the following specifications:
   - it should have two attributes, `width` and `length`, of type `double`
   - it should have eight methods:
     - two setters and two getters (i.e., one for each attribute),
     - a method to compute the area of a precise rectangle,

[2] https:/princomp.github.io/labs/Rectangle#enriching-rectangle.cs

- a method named `ComputePerimeter` to compute the perimeter of a precise rectangle,
- a method named `Swap` to swap the length and the width of a precise rectangle, and
- a method named `MultiplyRectangle` to multiply the length and width of a precise rectangle by a factor given in the argument as an integer.

5. Declare and manipulate rectangles with floating-point (i.e., `double`) values for the width and the length in the `Main` method, and make sure they behave as expected (i.e., Can you compute the area, set attributes, etc.?).

Pushing Further (Optional)

The following is an independent task with the goal of widening your understanding of this class and preparing you for the next labs. Now that you know more about naming conventions, have a look at Microsoft's naming guidelines[3], and particularly at:

- the documentation on general naming conventions[4] and
- the documentation on capitalization conventions[5].

[3] https://docs.microsoft.com/en-us/dotnet/standard/design-guidelines/naming-guidelines

[4] https://docs.microsoft.com/en-us/dotnet/standard/design-guidelines/general-naming-conventions

[5] https://docs.microsoft.com/en-us/dotnet/standard/design-guidelines/capitalization-conventions