# Precise Rectangle

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

January 17, 2023 (07:48:16 PM)

## Contents



In this lab you will first edit an existing program then create a new program "from scratch". You will also explore C# documentation and learn new useful IDE features.

# 1 Writing Your Own PreciseRectangle Class

In this exercise, you will create your own first class instead of using and expanding one that was written for you. The idea is to take inspiration from the class you already know (`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 re-enforce what you should know already (except for how to create a class in your IDE).

## 1.1 Implementation

To implement your class in your IDE, you are given two methods below: you can edit the pre-existing project, or start "fresh". It is recommended to pick the one you feel the most comfortable with, and then to try the other technique: you should know both how to edit existing projects and to start from scratch.

### 1.1.1 Edit the Pre-Existing Project

1. Re-download the "Rectangle" project[1], extract it in a folder, and open it with your IDE.

[1] labs/PreciseRectangle/../Rectangle/Rectangle.zip

2. Within your IDE, re-name the project to "PreciseRectangle", and rename the "Rectangle.cs" file to "PreciseRectangle.cs"

   It is important that you re-name 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 have to edit `PreciseRectangle.cs` to now store the `width` and the `length` with `double`, and to propagate this change accordingly. What should be the return type of `GetWidth`, for instance?

6. Declare and manipulate precise rectangles (with non-integer values for the width and the length) in the `Main` method, and make sure they behave as expected (can you compute the area, for instance?).

7. Add the missing methods `ComputePerimeter`, `Swap`, `MultiplyRectangle`, described in the section below.

### 1.1.2 Starting From Scratch

1. Create a new project in your IDE, 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 specification:

   - it should have two attributes, `width` and `length`, of type `double`
   - it should have eight methods:
     - two setters, two getters (i.e., one for each attribute),
     - one method to compute the area of a precise rectangle,
     - one method `ComputePerimeter` to compute the perimeter of a precise rectangle,
     - one method `Swap` to swap the length and the width of a precise rectangle,
     - one method `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 (can you compute the area, for instance?).

## 2 Pushing Further (Optional)

The following is an independent task, to widen your understanding of this class, and to prepare you for the next labs. Now that you know more about naming conventions, have a look at Microsoft's naming guideline[2], and particularly at

- the documentation on general naming conventions[3],
- and the documentation on capitalization conventions[4].

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