# User-Define types – Classes

Classes are first-class citizen in C#. You may define classes anywhere, even inside other classes (nested classes), however defining them externally greatly simplifies access in your code.

This exercise comprises of two parts in implemented in two separate projects.

# Part I

This first design is a poor one for reasons we will see later in this lab and in this course.

## Rectangle class (Original)

Create a project and code the following type outside the existing Program class.

For now, a class is a user made-up data **type**.

public class BadRectangle

{

//these are data members of type int

public int length;

public int width;

}

### Creating methods that uses the Rectangle class

Competed the following programming tasks:

Write the following static methods in your Program class:

#### Method: CreateBadRectangle()

**Name**: CreateBadRectangle

**Returns**: A BadRectangle object

**Argument**: int representing the width, int representing the length

**Action**:

The dot operator allows to you access members of a **type**.

* Creates a BadRectangle object using the new operator
* Set the width and length as specified by the arguments
* Return the object created

#### Method: DescribeBadRectangle()

**Name**: DescribeBadRectangle

**Returns**: void

**Argument**: BadRectangle object

**Action**: Displays the width and length of the Rectangle object

#### Method: CalculateAndDisplayArea()

**Name**: CalculateAndDisplayArea

**Returns**: void

**Argument**: BadRectangle object

**Action**:

* Calculate the area of the rectangle object represented by the parameter (length x width)
* Display the result of the above computation

In your main method write the code to do the following:

1. Create a rectangle of size 4X5 (use the first method) and assign it to an object reference called smallRectangle. You must use the CreateBadRectangle() method for this.

The **new** operator creates a new object.

1. Call the DescribeBadRectangle() method with argument smallRectangle to make sure that the object was created correctly.
2. Call the CalculateAndDisplayArea() method with argument smallRectangle to verify that the method works correctly.

### What is wrong with the Original Rectangle class?

1. The public fields length and width does not any way of stopping external code from changing them. Think of an account object where any code can change the balance!!!
2. The methods **CreateBadRectangle()**, **DescribeBadRectangle()** and **CalculateAndDisplayArea()** are very closely linked to the BadRectangle class and should be in that class.

# Part II

This design is a better one because we will fix the above design flaws.

## Rectangle class (Modified)

Create another project and code the following type outside the existing Program class.

public class Rectangle

{

//These type of data members are called fields.

//Notice the access level is restricted to private.

private int length;

private int width;

//Constructors have the same name as the class.

//Define like methods but no return type is specified.

//It is called immediately after the object is created

//It is a good place to put code to make your object useful

public Rectangle(int len, int wid)

{

length = len;

width = wid;

}

//Classes may contain action members also called methods.

public int GetArea()

{

return length \* width;

}

//This method is called when you need to print an object.

//Very useful when developing code.

public override string ToString()

{

return $"Length:{length}, width:{width}";

}

}

## Write the code to exercise the Rectangle class

In your Main method of the existing Program class write the code to do the following:

Create a rectangle object of length 8 and width 5

Print the object

Print the area of the object

Create a rectangle object of length 5 and width 3

Print the object

Print the area of the object

Create a rectangle object of length 20 and width 10

Print the object

Print the area of the object