

CSCI 1301 – Lab 8

1 Writing A Circle Class

This time, you will have to start your project “from scratch” and shouldn’t try to edit a previous program.

1.1 Foundations

1. Create a new project in VS, name it “Circle”.
2. In the Solution Explorer, right-click on “Circle”, then on “Add...” and select “Class”. Then, select “Class”, write “Circle.cs” as the name of the file, and click on “Add”.
3. You are now suppose to have two .cs files opened and displayed in the Solution Explorer: Program.cs and Circle.cs.
4. Declare one attribute in Circle.cs, of type double and named radius. Write a set and a get method for this attribute.
5. In Program.cs, write statements that create a new Circle object and set its radius to 2.3. Display its radius at the screen using the method you defined previously.

1.2 Extending the Class

1. In C#, Math.PI is a double holding an approximation of π . In the Main method of Program.cs, write a statement that displays its value at the screen. It should be 3.14159265358979.
2. Now, edit this statement and use the format specifier N, to display the value of π rounded to 3.14.
3. In the Circle.cs file, add two methods:
 - a) A method that returns the circumference of the circle that calls it (i.e., 2π times the radius),
 - b) A method that returns the area of the circle that calls it (i.e., π times the radius squared).
4. Test those two methods in your Main program, by displaying at the screen the area and the circumference of the object you created at the previous exercise.
5. Use the format specifier N to round the circumference.

1.3 Solution

You can download an archive¹ with a possible solution to the problem (that also includes a ToString method for the Circle class).

2 A Room Class

Now, we will create a Room class “from scratch”.

¹[Circle.zip](#)

2.1 Initial Set-Up

Create a Room class, with three attributes: one to hold the name of the room, one for the length of the room, and one for the width of the room. Name the attributes the way you want, and pick appropriate datatypes, knowing that we want to be able to store the length and the width of the rooms (expressed in meter) using floating point numbers.

Create 6 methods:

- A method to set the value of each attribute (“setters”),
- A method to get the value of each attribute (“getters”).

To test your Room class, edit your main method to create a Room object and ask the user for its name, length and width, and then display at the screen the name of the Room object that was created.

2.2 Adding Methods

Now, add two methods:

- A constructor that takes three arguments,
- A method that returns the area of the room in square meters.

Test them before moving on.

2.3 Internationalization of the Room Class

Suppose that we want to accomodate US users, that are more familiar with feet. Note that *we don't want to change the purpose of our width and length attributes, that are still supposed to hold dimensions in meters*, but we want to create methods that perform the conversions for us. Remembering that

- 1 meter = 3.28084 feet,
- 1 feet = 0.3048 meter,

add four methods to your class:

- A method that returns the width of the room in feet,
- A method that returns the length of the room in feet,
- A method that returns the area of the room in square feet,
- A constructor that takes two arguments, a length and a width in feet, and create an object with the corresponding measures in meters.

Try to write those methods using constant values for the conversion factors, and test them before moving on.

2.4 A ToString Method

Finally, create a `ToString` method. To understand the need for such a method, start by trying to display an object “directly”: suppose you have a `Room` object called `myKitchen`, add in your main method

```
Console.WriteLine(mykitchen);
```

Compile and execute your program. Is the information displayed at the screen what you expected? Is it useful?

Add the following to your `Room` class:

```
public override string ToString() {  
    return "The name of the room is...";  
}
```

1. Test this method by adding

```
Console.WriteLine(mykitchen.ToString());
```

to your main method.

2. Remove `.ToString()` from the previous statement and compile your program again: did something change?
3. “Expand” this method by having it return a more meaningful string: the string returned should also contain the name of the room and its dimensions in meters and feet. Use format specifiers to make it look nice!