

# CSCI 1301 – Lab 10

September 9, 2019

## 1 Solution to Previous Lab

You can download an archive<sup>1</sup> with a possible solution to the previous lab (that also includes a `ToString` method for the `Circle` class).

## 2 A Room Class

In this lab, we will create a `Room` class “from scratch”, as we did in the previous lab.

### 2.1 Initial Set-Up

Create a `Room` class, with three instance variables: 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 instance variable (“setters”),
- A method to get the value of each instance variable (“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.

### 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.

---

<sup>1</sup>[Circle.zip](#)

## 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 feets,
- 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, 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 meaningfull 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!