CSCI 1301 – Lab 09

February 3, 2019

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 single instance variable in `Circle.cs`, of type `double` and named `radius`. Write a `set` and a `get` method for this instance variable.
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.
2. 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).
3. 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.

1.3 Custom Constructors

1. Write a constructor that takes a `double` as argument, and set the value of the radius created to the value given as a parameter. Note that you can't compile your program anymore. Why?
2. Edit the way you were creating the object in the `Main` method to make your program complilable again. Instead of the default constructor (that isn't available anymore), use the constructor you just defined.
3. Add a custom "no-args" constructor to your class, using

   ```
   public Circle() { }
   ```

4. Edit your `Main` method so that two objects are created: one using the "no-args" constructor, one using the constructor that takes an argument.

1.4 Post-implementation Design

Draw the UML diagram for the class you just implemented. Make sure it matches perfectly your implementation.

2 Pushing Further (Optional)

The following are two independent tasks, to widen your understanding of this class, sharpen your coding skills, and prepare you for the next labs.

1. Re-open your `Rectangle` project from the previous lab and write three constructors:

   - One that doesn't take any argument and set both attributes to 1,
   - One that takes one argument, and set both the length and the width to the value passed as argument,
   - One that takes two arguments and set the length to the first value passed as argument, and the width to the second value passed as argument.

   Test them, and make sure they have the expected behavior.

2. Have a look at the `Math` class's specification, at https://docs.microsoft.com/en-us/dotnet/api/system.math?view=netframework-4.7.1 Can you write a statement that display the result of $Min(2^{14}, \lfloor log(4000 \rfloor)$?