

C# Review/OOP Practice – Shape Class

**Deadline: Monday, October 26th 2020, 09:00**

**GitHub Classroom Repository**

# Introduction

This practice is designed to help you familiarize yourself with inheritance(one object copying everything from another), polymorphism(a reference to a parent object being able to assign to a child object), encapsulation (everything is responsible for itself and only changes the STATE of itself) and abstraction(we have an abstract parent that can be inherit from but can not instantiate itself).

# Requirements

* “.Shapes” namespace extension created. (Create folder ex. People and move Person student and teacher to folder)
  + Create a folder called “Shapes”. (namespase CSharpOOP.People)
    - All classes in “Shapes” should have “.Shapes” as part of their namespace.
* Abstract “Shape” class created in the “Shapes” folder. (class)
  + Properties(this properties are derived), public getters only that **must** be overridden and provided functionality by derived classes: Do not have setter
    - “Area” (formula of Circle,rec,Tri will be overridden in order to change formulas specific for each shape)
    - “Perimeter” (formula of Circle,rec,Tri will be overridden in order to change formulas specific for each shape)
  + Traditional Properties: (getters and setters)
    - “Colour”(string)
      * String for the name of the colour is sufficient.
  + Methods:
    - Default and greedy constructors that **are** overridden and provided functionality by derived classes. (ex. constructor and the base and called the base constructor in child classes)11:24am 23Oct
      * “Colour”(property) should not be explicitly set in the derived classes, the base constructor should be called.
* “Circle” class derived from “Shape” created in the “Shapes” folder.
  + “Perimeter” is equivalent to a circumference for “Circle”.
  + Properties: (declared in this classes that are required to form those formulas)
    - “Radius” (use those to fill in functionality for Area and Perimeter formulas)
    - “Diameter” (use those to fill in functionality for Area and Perimeter formulas)
    - *Area –inherited*
    - *Perimeter - inherited*
      * Public getter that derives its value from “Radius”.
* “Rectangle” class derived from “Shape” created in the “Shapes” folder.
  + Properties: ( declared in this classes that are required to form those formulas)
    - “Length” (use those to fill in functionality for Area and Perimeter formulas)
    - “Width” (use those to fill in functionality for Area and Perimeter formulas)
    - *Area –inherited*
    - *Perimeter - inherited*
* “Triangle” class derived from “Shape” created in the “Shapes” folder.
  + Represents an equilateral or isosceles triangle, not a scalene triangle.
  + Properties: (declared in this classes that are required to form those formulas)
    - “Base” (use those to fill in functionality for Area and Perimeter formulas)
    - “Height” (use those to fill in functionality for Area and Perimeter formulas)
    - *Area –inherited*
    - *Perimeter - inherited*
* “Drawing” class created in the root (not in “Shapes”):
  + Properties: (list of shapes)
    - “Shapes” (private getter and setter)
      * Polymorphic list of “Shape”. (circ, rectan, tri)
    - “SpaceCovered” (public getter only) (will get the sum of all areas of all the shapes in shapes)
      * Total area of the contents of “Shapes”.
    - “LinesDrawn” (public getter only). (will get sum of all diameters)
      * The total perimeter/circumference of the contents of “Shapes”.
  + Methods: (will accept shape and add it to the shape’s List)
    - “Draw(Shape)”
      * Accepts a polymorphic “Shape” argument and adds it to the “Shapes” list.
    - A ToString() override, which will output “A drawing consisting of X shapes.” where X is the number of items/shapes in “Shapes”.
    - A default constructor that will initialize(the List) “Shapes”.
* Add comments to specify at least one area of the program that demonstrates each pillar of OOP (see Introduction).

# 

# Challenges

* Research and implement getter arrow notation on the “Perimeter”, “Area” and “Perimeter” properties.
* Research and implement LINQ in “SpaceCovered” and “LinesDrawn”.
* Ensure that the “Length” of “Rectangle” is always the largest of “Length” and “Width”.
* Allow the “Radius” of “Circle” to be set through either “Radius” or “Diameter”. (set the radius and retrieve it)
* Make “Colour” an enumeration with the values “Red”, “Blue” or “Green”.
  + Require it in the constructors of each shape.
  + In the ToString of drawing, output what percentage of the drawing is each colour, to the nearest hundredth of a percent.
    - Example: “A drawing consisting of X shapes that is AA.AA% red, BB.BB% green and CC.CC% blue."

# Reminders

Ensure you are tracking your time and that your timesheet is in the appropriate folder for viewing and marking.

Ensure you are committing frequently. It is advised to commit once per successful feature implementation at a minimum.

Ensure you are pushing your repository to the GitHub Classroom repository, and not a personal, private repository.

Ensure your repository has a readme with at a minimum your name, the name of the project, the project’s purpose and a link to your Trello board.

Ensure you are using Trello appropriately to keep track of outstanding features, and that it is linked in the README.md file.

Ensure you are using the #class channel to request clarifications on assignment specifications.

Ensure you are using the #homework-help channel in slack to request for assistance from instructional staff if needed, and that you include (at a minimum) a specific description of the problem and a list of what you have tried.

Feel free to reach out on #peer-support if the #homework-help queue is lengthy. If someone helps you out, [give them a shoutout using our handy-dandy form](https://docs.google.com/forms/d/1dKWKfNMABm8XmsB5iik5djcathQ3CT5g10-0rD7xGz0)!

# Citation Guide for Borrowed Code

Whenever you borrow code, the following information must be included:

* Comments to indicate both where the borrowed code begins and ends.
* A source linking to where you found the code (URL, book, example, etc.).
* Your reason for adding the code to your assignment or project instead of writing it out yourself.
* Explain to us how the code is supposed to work, include links to documentation and articles you read to help you understand.
* A small demonstration to prove you understand how the code works.

