CS 420 – Advanced Programming Languages

Assignment 3 – Object Oriented Programming in MATLAB – 100 points

In this comprehensive MATLAB assignment, you will work with a scenario involving geometric shapes. You will design an OOP structure to represent different types of shapes and perform various operations on these shapes. Additionally, you will visualize the shapes and their properties using MATLAB plots.

Submissions:

You will turn in two files on to canvas.

1. zip file containing all .m files
2. A Comprehensive report

Task 1: Class Creation and Constructors

1. Create a base class **Shape** with the following properties and methods:
    - Properties: **Name** (name of the shape) and **Area**
    - Constructor: Implement a constructor that takes the name as an input argument.
    - Methods: **Display** (to display the shape's properties).

Testing Instructions:

- Create two objects of the **Shape** class with a name of your choice using the constructor.
- Use the **Display** method to display the objects' properties.
- Ensure that the object's **Area** property is [] at this stage.

Expected Result:

- Add screenshot of your code (class) and the MATLAB command window to the report displaying the object's properties, including the name. The **Area** property should be [].

Task 2: Inheritance and Constructor Overloading

2. Create three derived (child) classes from **Shape**:
    - **Circle**: This class should have a radius property and implement the **CalculateArea** method to calculate the area of a circle.
        1. Overload the constructor to accept the radius.
            - The constructor should call the parent constructor explicitly using '@' to set the name to be Circle.
    - **Rectangle**: This class should have length and width properties and implement the **CalculateArea** method to calculate the area of a rectangle.
        1. Overload the constructor to accept length, width.
            1. The constructor should call the parent constructor explicitly using '@' to set the name to be Rectangle.
    - **Triangle**: This class should have base and height properties and implement the **CalculateArea** method to calculate the area of a triangle.
        1. Overload the constructor to accept base, height.
            1. The constructor should call the parent constructor explicitly using '@' to set the name to be Triangle.

Testing and Expected Result:

- Create <u>two</u> objects of the **Circle, Square and Rectangle** class with a properties of your choice.
- Add screenshots of all your classes to the report. Also add a screenshot of objects created and their properties.

Task 3: Method Overriding

Override the **Display** method in each of the derived (child) classes to include information specific to the shape type.

- **NOTE**: Display method should output (print) a with all the properties & area. Sample output string for circle class: The area of a circle with a radius of 5 units is approximately 78.54 square units.

Testing Instructions:

- Create more objects of the **Circle**, **Rectangle**, and **Triangle** classes using their respective constructors.
- Calculate the areas for all the objects created.
- Call the **Display** method for each object to display its properties.
- Ensure that the overridden **Display** method shows the specific information for each shape.

Expected Result:

- Separate screenshots showing the MATLAB command window for each shape's **Display** method output. Each should display the name and shape-specific information.

Task 4: Multi-level Inheritance

Create a new **class** named **EquilateralTriangle** that inherits from the Triangle class.

- Add a property for the side length to the **EquilateralTriangle** class and implement its constructor to accept the side length.
    1. Recall that the objects created will make a call to the parent constructor. This constructor in the EquilateralTriangle class calculates and update the base and height property in the parent class. They should be set to appropriate values call using '@'.
        - For example if side length = 10. Base should be set to 10 and height should be set to $5\sqrt{3}$.
        - Look up for the formula to computer the height of equilateral triangle.

- Override the **Display** method in the class to display the side length and the area of an equilateral triangle.
    - Example : area of the equilateral triangle with a side length of 5 units is approximately 10.825 square units
- DO NOT OVERRIDE CalculateArea method.

Testing Instructions (Step 4):

- Create an object of the **EquilateralTriangle** class.
- Call the **CalculateArea** method.
    - NOTE: This is a method in the parent class. You SHOULD NOT override the calculateArea method.
- Display the properties of the equilateral triangle using the **Display** method.
- Ensure that the area is correctly calculated and displayed.

Expected Result (Step 4):

- A screenshot showing the cods and MATLAB command window with the properties of the equilateral triangle, including the calculated area.

Task 5: Multiple Inheritance

Create a new class named **ColorMixin** that contains a property for **color** and a method for setting and getting the color.

- Add a property for the **color** to the **ColorMixin** class and implement its constructor to accept the color variable.
- Write two methods called "GetColor" and "SetColor" to act as assessor and mutator methods.

Modify all the four classes **"Circle", "Square", "Triangle" and "Equilateral Triangle"** to inherit from ColorMixin to include color properties and color-related methods.

- Note: None of the methods or the property in **ColorMixin** should be overwritten.
- Adjust the constructor in the classes to accommodate for the new parent class.
- Modify the display method to include the color information.

Testing Instructions:

- Create more objects of the **Circle**, **Rectangle**, **Triangle** and **EquilateralTriangle** classes using their respective constructors.
- Calculate the areas for all the objects created.
- Call the **Display** method for each object to display its properties (including color).

Expected Result:

- Add screenshots of the modified classes and Colormixin class. Also add, separate screenshots showing the MATLAB command window for each shape's **Display** method output. Each should display the name and shape-specific information.
- Now use the setColor method to change the color of all the four objects created.
- Use Display method to show that the color has been updated. Add screenshots.

Task 6: Static Method

0. Modify the shape class to inherit from `matlab.mixin.Heterogeneous`
1. Create a new **static method** in the **Shape** class called **CalculateStatistics**. This method should compute and display statistics on the areas of all the shapes in an array.
2. Implement the following statistics:
    - Mean area
    - Median area
    - Standard deviation of areas
3. The method should also printout the computed values.

Testing Instructions:

- Create an array of various shapes, including circles, rectangles, triangles and Equilateraltriangles.
- Compute the area of all the objects.
- Place the objects in an array.
- Call the static method **CalculateStatistics** method on the array.
- Display the computed statistics in the MATLAB command window.

Expected Result :

- A screenshot with array of Objects.
- A screenshot of the MATLAB command window showing the computed statistics (mean, median, and standard deviation) for the areas of the shapes in the array.

Task 7: Visualization

Drawing Circles, Rectangles and Triangles.

1. Add a method **Draw** to the **Circle** class that draws an actual circle based on the radius and color of the circle. Use MATLAB's **rectangle** or **viscircles** function for this purpose.
2. Add a method **Draw** to the **Rectangle** class that draws an actual rectangle based on its dimensions (length, width) and color. Use MATLAB's **rectangle** function for this purpose.
3. Add a method **Draw** to the **Triangle** class that draws an actual triangle based on its dimensions (base, height) and color. You can use the MATLAB **plot** function to draw the triangle.
    - The draw method should adjust the axis to make sure the shape is properly visible
    - The draw method should fill in the shape with specified color.
    - Give a title to the plot.
    - The plot should show the properties of the shape.
    - The plot should also show the area of the shape.
    - Explore the plotting functionality in MATLAB to make your plots standout.

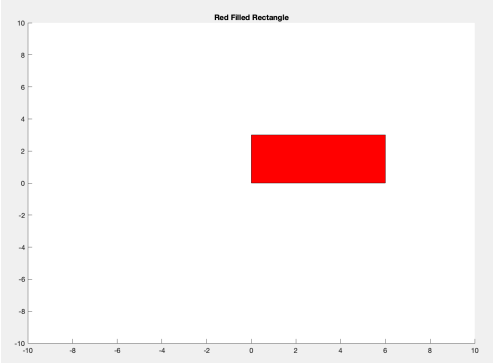
Testing Instructions:

- Create more objects of the **Circle**, **Rectangle**, **Triangle** and **EquilateralTriangle** classes using their respective constructors.
- Compute their Areas
- Call the draw method to visualize the shape.

Expected Result:

- A screenshot showing the drawn circle in the MATLAB figure.

Here is a sample screenshot. (Note this screen shot does not have the length, width or Area information shown- your plot will show these information)



Task 8: User Interaction

This is a fun task – you can get creative.

Write a MATLAB Script called **"MyShapes.m"** to Create a user-friendly command-line interface that allows users to create shapes interactively.

Your script should:

1. Prompt user to choose one of the 4 available shapes.
2. Prompt user for the properties values corresponding to the shape selected by the user.
3. Internally, you script should create an object of the chosen shape, computer area and call the draw method.
4. Output of the script will be a plot.

Testing Instructions:

- Run the script and follow the interactive interface to create various shapes.
- Ensure that the shapes are created with the specified properties.

Expected Result:

- Screenshot of the code in the script file.
- Screenshots of the interactive interface and the created shapes.

Task 9: Optional - Graphical User interface **- (Extra Credit)**

If you work on this part – you should show me a live demo to get the extra credit points.

Your task is to implement task 8 using a Graphical User interface.

- You should create simple buttons (radio button) to choose among the available shapes.
- Use checkboxes, color picker or similar approach for getting the color.
- Use dropdown menu, slider or similar approach to read the property values such as radius, length, etc.
    - Only restriction is text boxes should not be used, you can get creative and choose any other option to input the values from the user.

This can easily be done using Guide Tool (Easier) in MATLAB.
You can also use other advanced GUI tools available in MATLAB. Get creative and have fun!