Department of Computer Science
The City College of CUNY

CSc 22100 1XB 6822: Software Design Laboratory Summer 2022

<u>Assignment 1</u>

A <u>report</u> uploaded on the Bloackboard's course page for the section showing:

[1] *statement of the problem,*
[2] *solution methods,*
[3] *all codes developed, and*
[4] *outputs produced for the tasks indicated,*

is due by ***11:59 pm on Sunday, 26 June 2022***. ***The deadline is strictly observed***.

1- Create a hierarchy of Java classes as follows:

MyLine *extends* **MyShape**;
MyRectangle *extends* **MyShape**;
MyOval *extends* **MyShape**.

Class MyPoint:

Class **MyPoint** is used by class **MyShape** to define the reference point, **p**(x, y), of the Java display coordinate system, as well as by all subclasses in the class hierarchy to define the points stipulated in the subclass definition. The class utilizes a color of **enum** reference type **MyColor**, and includes appropriate class constructors and methods, including methods that perform point related operations.

Enum MyColor:

Enum **MyColor** is used by class **MyShape** and all subclasses in the class hierarchy to define the colors of the shapes. The **enum** reference type defines a set of constant colors by their red, green, blue, and opacity, components, with values in the range [0 – 255]. The **enum** reference type includes a constructor and appropriate methods, including methods that return the corresponding hexadecimal representation and JavaFx Color objects of the constant colors.

Class MyShape:

Class **MyShape** is the superclass of the hierarchy, extending the Java class Object. An implementation of the class defines a reference point, **p**(x, y), of type **MyPoint**, and the color of the shape of **enum** reference type **MyColor**. The class includes appropriate class constructors and methods, including methods, including methods that perform the following operations:

a. *perimeter, area* – return, respectively, the perimeter and meter of the **MyShape** object. These methods must be overridden in each subclass in the hierarchy. For the **MyShape** object, the methods return zero.
b. *toString* – returns the object's description as a String. This method must be overridden in each subclass in the hierarchy;

c. *draw* – draws the object shape. This method must be overridden in each subclass in the hierarchy. For the **MyShape** object, it paints the drawing canvas in the color specified.

Class MyLine:

Class **MyLine** extends class MyShape. The **MyLine** object is a straight line segment defined by the endpoints $\mathbf{p}_1(x_1, y_1)$ and $\mathbf{p}_2(x_2, y_2)$ of type **MyPoint**. The **MyLine** object may be of any color of **enum** reference type **MyColor**. The class includes appropriate class constructors and methods, including methods that perform the following operations:

a. *getLine* — returns the **MyLine** object*;*
b. *length, xAngle* — returns, respectively the length and angle with the *x*-axis of the **MyLine** object;
c. *perimeter, area* — return, respectively, the perimeter and area of the **MyLine** object;
d. *toString* — returns a string representation of the **MyLine** object, including the line's endpoints, length, and angle with the *x*-axis;
e. *draw* — draws a **MyLine** object.

Class MyRectangle:

Class **MyRectangle** extends class **MyShape**. The **MyRectangle** object is a rectangle of height *h* and width *w*, and a top left corner point $\mathbf{p}(x, y)$, and may be filled with a color of **enum** reference type **MyColor**. The class includes appropriate class constructors and methods, including methods that perform the following operations:

f. *getTLCP, getWidth, getHeight* — return, respectively, the top left corner point, width, and height of the **MyRectangle** object
g. *perimeter, area* — return, respectively, the perimeter and area of the **MyRectangle** object;
h. *toString*— returns a string representation of the **MyRectangle** object: top left corner point, width, height, perimeter, and area;
i. *draw*— draws a **MyRectangle** object.

Class MyOval:

Class **MyOval** extends class **MyShape**. The **MyOval** object is defined by an ellipse within a rectangle of height *h* and width *w*, and a center point $\mathbf{p}(x, y)$. The **MyOval** object may be filled with a color of **enum** reference type **MyColor**. The class includes appropriate class constructors and methods, including methods that perform the following operations:

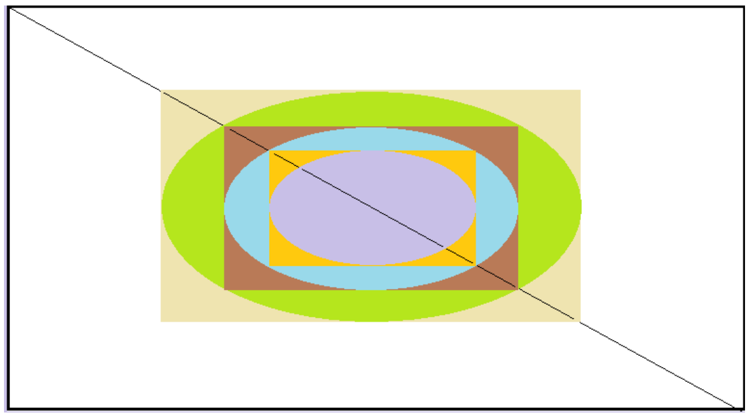
a. *getCenter, getMinorAxis, getMajorAxis* — return, respectively, the center point and semi minor and major axes of the **MyOval** object;
b. *perimeter, area* — return, respectively, the perimeter and area of the **MyOval** object;
c. *toString*— returns a string representation of the **MyOval** object: semi minor and major axes lengths, perimeter, and area;
d. *draw*— draws a **MyOval** object.

2- Use JavaFX graphics and the class hierarchy to draw the geometric configuration comprised of a sequence of alternating concentric ovals and inscribed rectangles, as illustrated below, subject to the following additional requirements:

a. The code is applicable to canvases of variable height and width;
b. The dimensions of the shapes are proportional to the smallest dimension of the canvas;

c. The ovals and rectangles are filled with different colors of your choice, specified through an **enum** reference type **MyColor**.

3- <u>Explicitly specify all the classes imported and used in your code</u>.



Best wishes

Hesham A. Auda
06-16-2022