Department of Computer Science
The City College of CUNY

CSc 22100: Software Design Laboratory [Spring 2020]

Exercise 2

A printed report showing [1] *the problem,* [2] *solution methods,* [3] *codes developed, and* [4] *outputs produced for the assignment indicated is due during and/or before the end of the class on Tuesday, 24 March 2020. **The deadline is strictly observed**.*

1- Amend the hierarchy of Java classes in Exercise 1 as follows:

MyLine *is_a* **MyShape**;
MyPolygon *is_a* **MyShape**;
MyRectangle *is_a* **MyShape**
MyOval *is_a* **MyShape**;
MyCircle *is_a* **MyOval.**

2- Class **MyShape** is an abstract class; is the hierarchy's superclass; and inherits Java class Object. The *draw* method in class **MyShape** is an *abstract* method and hence must be overridden in each subclass in the hierarchy. Otherwise, the classes **MyShape**, **MyLine**, **MyPolygon**, and **MyCircle** are defined as in Exercise 1 but now utilize.

Class MyRectangle:

Class **MyRectangle** inherits class MyShape. The **MyRectangle** object is a rectangle of height h and width w, centered at a point (x, y), and may be filled with a color. The class includes appropriate class constructors and methods, including methods that perform the following operations:

a. *getWidth, getHeight, getPerimeter, getArea*— return the width, height, perimeter, and area of the **MyRectangle** object;
b. *setWidths, etHeight, setPerimeter, setArea*— set the width, height, perimeter, and area of the MyRectangle object;
c. *toString*— returns a string representation of the **MyRectangle** object: width, height, perimeter, and area;
d. *draw*— draws a **MyRectangle** object of height h and width w, centered at a point (x, y). The center point of the rectangle is defined in class **MyShape**.

Class MyOval:

Class **MyOval** inherits class **MyShape** and may use class **MyRectangle**. The **MyOval** object is defined by an ellipse inscribed in a rectangle of height h and width w, centered at a point (x, y). The **MyOval** object may be filled with a color.

The class includes appropriate class constructors and methods, including methods that perform the following operations:

a. *getPerimeter*— returns the perimeter of the **MyOval** object;
b. *getArea*— returns the area of the **MyOval** object;
c. *toString*— returns a string representation of the **MyOval** object: axes lengths, perimeter, and area;
d. *draw*— draws a **MyOval** object inscribed in a rectangle of height h and width w, centered at a point (x, y). The center point of the oval is defined in class **MyShape**.

3- Interface **MyPoint** and interface **MyShapePosition** are specified in connection with the class hierarchy. The abstract class **MyShape** implements interface **MyShapePosition** which extends interface **MyPoint**. All classes of the hierarchy must be amended in accordance with the two interfaces.

4- Interface **MyPoint** includes appropriate abstract, static, and/or default methods that describe the positional functions and behaviors of the specific object types of the class hierarchy, including:

a. *getPoint, setPoint*– return and set the point (x, y);
b. *moveTo* – moves point (x, y) to point $(x + \Delta x, y + \Delta y)$;
c. *distanceTo*– returns distance from point (x, y) to a point;

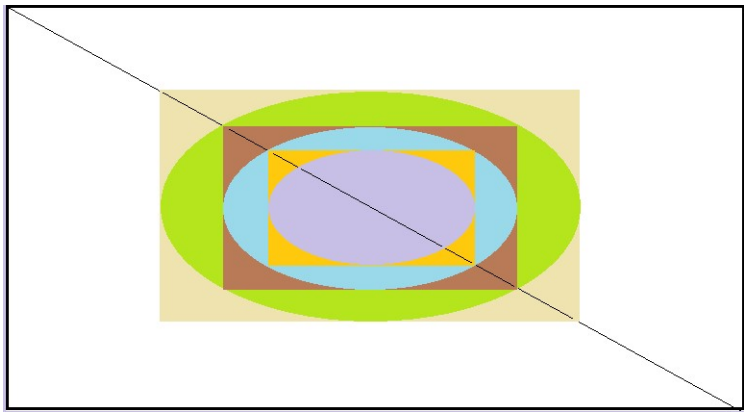
5- Interface **MyShapePosition** includes appropriate abstract, static, and/or default methods that describe the functions and behaviors of the specific object types of the class hierarchy, including:

a. *getMyBoundingBox*— returns the bounding rectangle of an object in the class hierarchy;
b. *doOverlap*— returns true if two objects in the class hierarchy do overlap; false otherwise.

6- Use JavaFX graphics and the class hierarchy to draw a geometric configuration comprised of a sequence of alternating concentric ovals and their 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 recatngles and ovals are filled with different colors of your choice, specified through a **MyColor** enum data type.

Explicitly specify all the classes imported and used in your Java code.



Best wishes

Hesham A. Auda
6 March 2020