Department of Computer Science
The City College of CUNY

CSc 22100: Software Design Laboratory [Spring 2021]

Assignment 2

A report uploaded on the Blackboard's course page for the section showing [1] *the problem,* [2] *solution methods,* [3] *codes developed, and* [4] *outputs produced for the assignment indicated is due by* 2:00 *pm on Thursday,* 25 March 2021. ***The deadline is strictly observed***.

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

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

2- Class **MyShape** is an **abstract** class; is the hierarchy's superclass; and inherits Java class Object. The *area*, *perimeter*, and *draw* methods in class **MyShape** are *abstract* methods and hence must be overridden in each subclass in the hierarchy. Further, the reference point of **MyShape**, **p**(x, y), is an object of class **MyPoint**. Otherwise, the classes **MyShape**, **MyLine**, **MyRectangle**, **and MyOval** are as defined in Assignment 1, but now utilize class **MyPoint**:

Class MyPoint:

Class **MyPoint** is used by class **MyShape** to define the reference point **p**(x, y) of the Java display coordinate system, and by all other subclasses in the class hierarchy any point stipulated in the class definition. The class includes appropriate class constructors and methods, including methods that perform point related operations.

Class MyPolygon:

Class MyPolygon extends class MyShape. The MyPolygon object is a *regular* polygon defined by the integer parameter, N — the number of the polygon's equal side lengths and equal interior angles, and the radius, r, in which it is inscribed. The MyPolygon object may be filled with a color. The class includes appropriate class constructors and methods that perform the following operations:

a. *getCenter, getAngle, getSide* — return the center point, interior angle (in degrees), and side length of a **MyPolygon** object;
b. *toString* — returns a string representation of a **MyPolygon** object: center point, side length, interior angle, perimeter, and area;

c. *draw* — draws a **MyPolygon** object.

3- Interface **MyShapeInterface** is implemented by class **MyShape**. All subclasses of the hierarchy must be amended in accordance with the interface. The interface includes appropriate constants and abstract, static, and/or default methods that describe the functions and behaviors of the specific object types of the class hierarchy, including:

a. *getMyBoundingRectangle* — abstract method returns the bounding rectangle of an object in the class hierarchy;
b. *pointInMyShape*— abstract method returns true if a point p is located within or on the boundary of an object in the class hierarchy;
c. *intersectMyShapes*— static method returns the intersecting area — i.e., the set of all points on or within the boundary of the area — of two objects in the class hierarchy if they do overlap; and **null** otherwise.

4- Use JavaFX graphics and the class hierarchy to run tests of the methods in interface **MyShapeInterface**. The tests should include multiple shapes and show graphic illustrations and textual information of outcomes generated by your code, 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 shapes are filled with colors of your choice, specified through a **MyColor** enum reference type.

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

Best wishes
Hesham A. Auda
11 March 2021