Department of Computer Science
The City College of CUNY

CSc 22100: Software Design Laboratory [Fall 2019]

Exercise 2

A *printed report* *showing the problem, solution methods, codes developed, and outputs produced for the assignment indicated is due during and before the end of the class on* *Tuesday*, 29 October 2019. *The deadline is strictly observed.*

Consider the class hierarchy in Exercise 1.

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, MyRectangle, and MyOval are defined as in Exercise 1.

class MyCircle:

Class MyCircle inherits class MyShape. The MyCircle object is defined by its radius, *radius*, and center (x, y), and may be filled with a color. The class includes appropriate class constructors and methods that perform the following operations:

a. *getRadius* — returns the radius of the MyCircle object;
b. *setRadius* — sets the radius of the MyCircle object;
c. *toString* — returns a string representation of the MyCircle object: radius, perimeter, and area;
d. *draw* — draws a MyCircle object of radius *radius*. The center point of the circle is defined in class MyShape.

class MyPolygon:

Class MyPolygon inherits *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. The MyPolygon object may be filled with a color. The class includes appropriate class constructors and methods that perform the following operations:

   a. *toString* — returns a string representation of the xxxPolygon object: side length, interior angle, perimeter, and area;
   b. *draw* — draws a MyPolygon object and inscribed in a circle of radius *radius*. The center point of the circle is defined in class MyShape.

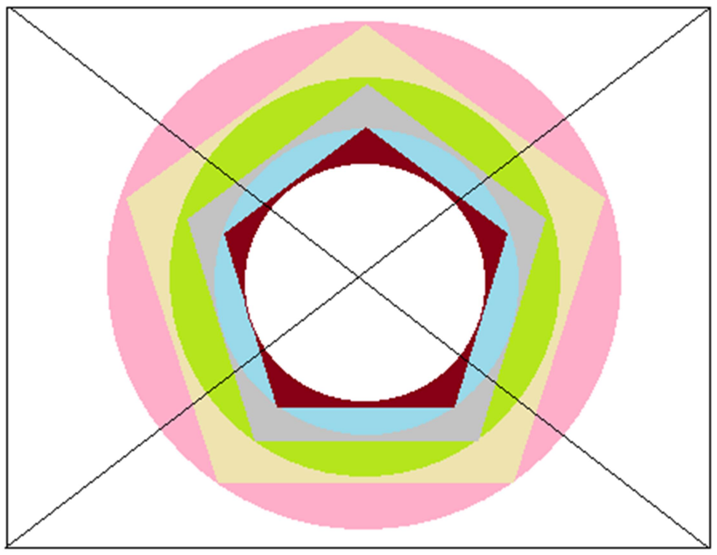
3- *Interface* MyPositionInterface and *interface* MyShapePositionInterface are specified in connection with the class hierarchy. The *abstract class* MyShape implements *interface* MyShapePositionInterface which extends *interface* MyPositionInterface.

4- *Interface* MyPositionInterface 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* – returns 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* MyShapePositionInterface 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.

6- Use JavaFX graphics and the class hierarchy to draw a geometric configuration comprised of a sequence of alternating concentric circles and polygons 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. Only the drawing methods of the GraphicsContext object may be used for drawing the shapes in the class hierarchy;
   d. The circles and polygons are filled with different colors of your choice.
   e. The colors used to fill the shapes in the class hierarchy are specified by an enum data type **MyColor**. The enum type includes appropriate constructors and methods, including methods that define, mix, and return colors.



Hesham A Auda
15 October 2019