Department of Computer Science
The City College of CUNY

CSc 22100 [P 34721]: Software Design Laboratory [Fall 2021]

Assignment 2

A report uploaded on the Bloackboard's course page for the section showing:

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

is due by 11:00 *pm on Thursday*, 21 October 2021. ***The deadline is strictly observed***.

1- Create a hierarchy of Java classes as follows:

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

Class MyShape:

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. The implementation of the class defines a reference point **p**(*x*, *y*), an object of type **MyPoint**, and the color of the shape of enum reference type **MyColor**. Otherwise, the classes **MyPoint**, **MyShape**, **MyLine**, **MyRectangle**, **and MyOval** are as defined in Assignment 1.

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*, of the circle in which it is inscribed. The **MyPolygon** object may be filled with a color. The class includes, among others, 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. *apothem* — returns the apothem of a **MyPolygon** object;
c. *toString* — returns a string representation of a **MyPolygon** object: center point, side length, interior angle, perimeter, and area;

2- Interface **MyShapeInterface** is implemented by class **MyShape**. All subclasses of the hierarchy must be amended in accordance with the interface. The interface includes

constants (as needed) and appropriate *abstract*, *static*, and *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.
d. *drawIntersectMyShapes* — default method returns a canvas with a drawing of the area of intersection of two objects in the class hierarchy if they do overlap.

3- 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:

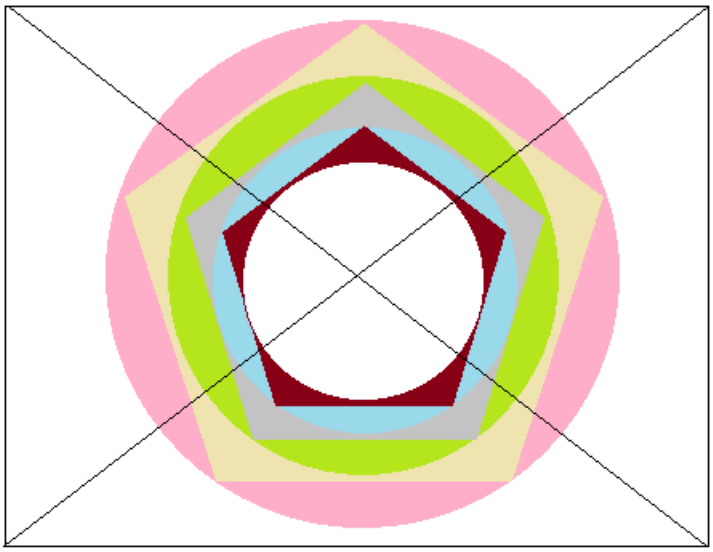
a. The code is applicable to canvases of variable height and width;
b. The dimensions of the shapes are proportional to the smallest dimension of the canvas;
c. The polygons and circles are filled with different colors of your choice, specified through the reference type **MyColor**; and
d. All objects are processed polymorphically.

Further:

- Draw the bounding rectangle of **MyLine**, **MyPolygon**, and **MyOval** objects of your choice;
- Draw the area of intersection of:

a. Two **MyRectangle** objects;
b. A **MyRectangle** object and a **MyCircle** Object; and
c. A **MyRectangle** object and a **MyPolygon** Object.

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



Best wishes

Hesham A. Auda
10-08-2021