Department of Computer Science
The City College of CUNY

CSc 22100 [46252 M]: Software Design Laboratory [Spring 2023]

Assignment 2

A report in PDF or MS Word, uploaded on the Blackboard course page for the section showing:

[1] *Statement of the problem,*
[2] *Solution methods,*
[3] *Full Java code developed, and*
[4] *Outputs produced by the code in* [3] *for the tasks indicated.*

is due by 11:59 *pm on Friday*, 31 March 2023. ***The deadline is strictly observed***.

1- Create a hierarchy of Java classes as follows:

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

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. The class also includes draw and toString methods, as well as methods that perform point related operations, including but not limited to, shift of point position, distance to the origin and to another point, angle [in degrees] with the x-axis in Java Coordinate System of a line extending from this point to another point.

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 the word-packed and hexadecimal representations and JavaFx Color objects of the constant colors in **MyColor**.

Class MyShape:

Class **MyShape** is an *abstract* class; is the hierarchy's superclass; extends Java class Object; and implements Interface **MyShapeInterface**. Methods *area*, *perimeter*, and *draw* in class **MyShape** are *abstract* methods, and hence must be overridden in each subclass in the hierarchy. Class **MyShape** also includes method *toString* that returns the object's description as a String, and hence must alsobe 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**.

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 **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:

a. *getTLCPoint, getWidth, getHeight* — return the top left corner point, width, and height of the **MyRectangle** object
b. *toString*— returns a string representation of the **MyRectangle** object: top left corner point, width, height, perimeter, and area;
c. *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 **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 the center point, minor axis, and major axis of the **MyOval** object;
b. *toString*— returns a string representation of the **MyOval** object: axes lengths, perimeter, and area;
c. *draw*— draws a **MyOval** object.

2- Interface **MyShapeInterface** is implemented by the abstract class **MyShape**. All subclasses of the hierarchy must therefore be amended in accordance with the interface. The interface includes *constants* (if needed) and appropriate *abstract*, *static*, and *default* methods that perform functions that provide specific functionalities described for the object types of the class hierarchy, including:

Abstract methods:
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.

Static methods:
c. *intersectMyRectangles* — *static* method returns the intersection of two **MyRectangles** objects $\mathbf{R}_1$ and $\mathbf{R}_2$ if they do overlap, and **null** otherwise.
d. *intersectMyShapes* — *static* method returns the set of all points on or within the boundary of the area of intersection of two **MyShape** objects $\mathbf{S}_1$ and $\mathbf{S}_2$ if they do intersect, and **null** otherwise.

Default methods:
e. *drawIntersectMyShapes* — *default* method returns a *canvas* with a drawing of the intersection of two objects in the class hierarchy if they do overlap.

3- Amend the JavaFx **BorderPane** Layout in Assignment 1, as follows:

BorderPane Layout:

BorderPane Layout is the root of the scene graph in the JavaFx Application. It is comprised of three regions – Top, Left, and Center:

a. *Top Region*: The region encompasses a **HBox** Layout, encompassing geometric images of the **MyShape** objects considered, each of which, upon selection, provides a dialogue box for entering the object's parameters and drawing the object on the **Canvas** in the Center region.
b. *Left region*: The region encompasses a **VBox** Layout, encompassing a **Label** object displaying the text "MyColor Palette" and a **MyColorPalatte** object showing a set of constant colors of enum **MyColor** type for color selection.
c. *Center region*: The region encompasses a **Canvas** object used for drawing **MyShape** objects and their intersection.

4- Use JavaFX graphics and the class hierarchy to draw the geometric configuration comprised of a sequence of alternating concentric circles 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 circles and rectangles are filled with different colors of your choice, specified through an enum reference type **MyColor**.

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

Best wishes

Hesham A. Auda
03-11-2023