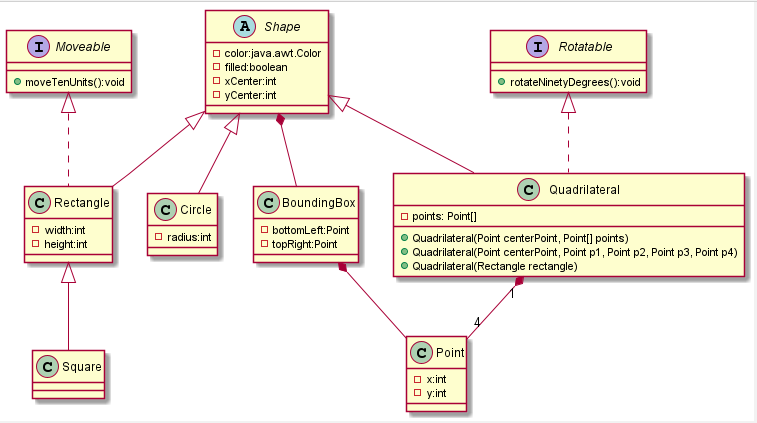
Requirements & Points Percentage Scheme

# Overview.

A Java Swing application that can display, move, and rotate filled and outlined shapes. Implementation will involve the concepts of Inheritance, Polymorphism, Abstract Classes and Interfaces.

Notes:

Java Swing Requirements can be satisfied using a JFrame containing a JPanel, with a MouseAdapter listener that handles mousePressed events (and overriding the paintComponent() method for Shape and String drawing). There are no requirements for additional UI elements such as JButtons.



*Figure 1: UML Class Diagram*

# Percentage Scheme

Correct display of standard shapes – Circle, Rectangle, Square [25 %] Correct display of Quadrilaterals [12 %]

Ability to “select” and toggle filled/outline of shapes based

on mousePressed events. [5 %

Ability to move relevant shapes based on

right-mouse-button pressed on selected shapes. [6 %]

Ability to rotate relevant shapes based on

right-mouse-button pressed on selected shapes. [13 %]

Show the Shape’s ClassName if ShapesManager

displayName Boolean is set to true. [5 %]

Show each Shapes’s BoundingBox based on a

Boolean variable (initialised at compile-time) [10 %]

Tester App and Non-functional Requirements:

JavaDoc commenting is supplied and adequate: [4 %]

Code – elegance of solutions; avoiding redundancy; well-structured; [12 %]

Code – Demo Application should display at least one of each of the following shapes: circle, rectangle, square.

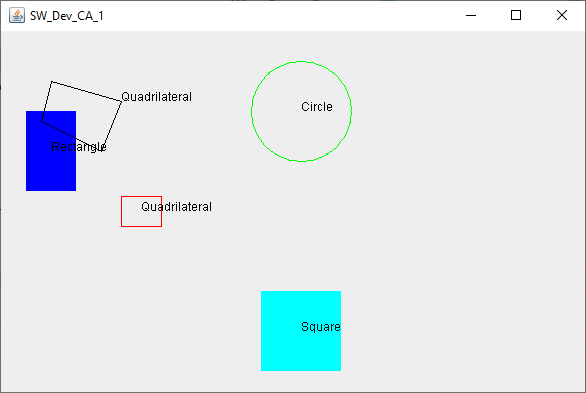
Additionally, at least two quadrilaterals should be displayed: one of these should be initialised/constructed via a Rectangle object, and one should be a non-regular quadrilateral.

The displayed shapes should ideally display the attributes associated with the shapes, i.e. some should be filled and some unfilled; different coloured shapes should also be displayed.

See the following screenshot for an example (I’ve turned on

the display of Shape names for clarity). In addition, I’ve supplied *some*

of the code used to create some of the shapes shown. [8 %]



*Figure 2: Screenshot of Sample Application*

public class TesterApp {

public static void main(String[] args) { ShapesManager shapesManager = new ShapesManager();

shapesManager.setDisplayName(true); shapesManager.setDisplayBoundingBox(false);

shapesManager.addShape(new Circle(Color.*green*, 300,80, 50));

Rectangle rect = new Rectangle(Color.*blue*, 50,120, true,50, 80); shapesManager.addShape(rect);

*//Note: I don't display this rectangle - I merely use it to initialise a quadrilateral*

Rectangle rect2 = new Rectangle(Color.*red*, 140,180, 40, 30); Quadrilateral quad = new Quadrilateral(rect2);

shapesManager.addShape(quad);

//ADDITIONAL CODE OMMITED.

*Code Listing 1: Example (partial) of Application initialisation used to produce Figure 2 Screenshot*