DESIGN PROBLEM #1

You are developing a drawing editor that lets a user draw and arrange graphical elements (e.g. lines, polygons, and text) into pictures and diagrams. The key abstraction in the software is the graphical object, implemented by an abstract class called Shape. The editor defines a subclass of Shape for each kind of graphical object: a *Line* class for lines, a *Polygon* class for polygons, and so forth.

The classes for the elementary geometric shapes like *Line* and *Polygon* are rather easy to implement; their drawing and editing capabilities are inherently limited. But a *TextBox* subclass that can display and edit text is considerably more difficult to implement, since even basic text editing involves complicated screen update and buffer management.

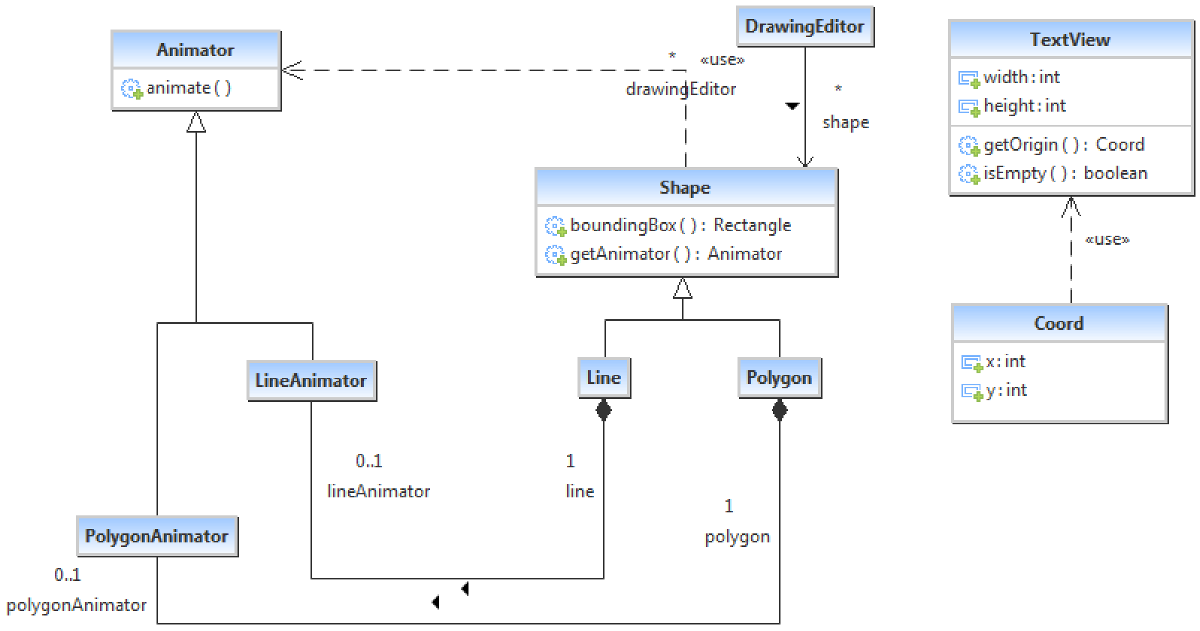
However, you have found an off-the-shelf user interface toolkit that already provides a sophisticated *TextView* class for displaying and editing text. You would like to reuse *TextView* to implement *TextBox*, but the toolkit wasn't designed with *Shape* classes in mind. So we can't use *TextView* and *Shape* objects interchangeably. And as you do not have the source code for the toolkit, you can't change the interface for *TextView*.

Shown below is a UML diagram of the relevant parts of the drawing editor and the user interface toolkit.

- The *Shape* class defines operations for getting the bounding box for the shape and access to an Animator which knows how to handle user interaction with a particular shape.
- The *TextView* class defines an operation for getting the origin (i.e. lower-left corner) and an operation for getting the width and height of the *TextView*. It also provides an operation for determining if the text widget is empty (i.e. has no text in it).

1. **Draw a UML diagram that shows how you could use TextView in your drawing program without changing its interface.**

2. **Use Java code to show how the Shape operations are implemented using the appropriate operations from TextView.**


Animator
animate ( )
DrawingEditor
«use»
*
drawingEditor
*
shape
Shape
boundingBox ( ) : Rectangle
getAnimator ( ) : Animator
TextView
width : int
height : int
getOrigin ( ) : Coord
isEmpty ( ) : boolean
«use»
Coord
x : int
y : int
LineAnimator
Line
Polygon
0..1
lineAnimator
1
line
1
polygon
PolygonAnimator
0..1
polygonAnimator