Practice Task 4:

This task extends the work from Practice 3. It includes my solutions to practice task 1 and 3.

You might have noticed that in the ShapeOverlay class move() method, we have to test whether each Shape object is a Circle or a Rectangle to figure out it's dimensions in order to see if it has hit a wall and needs to have it's direction reversed.

This is bad polymorphic design. Anytime we create a new subclass of Shape, we will have to update the ShapeOverlay class to handle this in the move() method. It would be better to have a standard interface for all Shape types so we can handle the boundary issue for all shapes without having to change the ShapeOverlay class.

To do this, we've added a small class called BoundingBox. The idea is that for any Shape, we can describe an imaginary box that contains it, and therefore gives the extreme leftmost, rightmost, top, and bottom pixel position of the Shape. To do that, a BoundingBox holds int x and int y representing the coordinate of upper left hand corner of the imaginary box. It also holds int width and int height of that box.

I've added an abstract method to the Shape class called getBounds() that returns the Shapes BoundingBox. This forces any derived class to implement the getBounds() method.

Task Part 1: There are currently method stubs in the Circle and Rectangle classes for the getBounds() method that just return null. Change these so a new BoundingBox is created, give it appropriate values, and return it.

Part 2: Change the move method in the ShapeOverlay class so it doesn't need to test what kind of Shape is being moved. Instead, it will call getBounds() on each Shape and use the BoundingBox that gets returned to figure out if the shape is at the edge of the panel and needs to have it's motion reversed.