Java Programming

Arthur Hoskey, Ph.D.
Farmingdale State College
Computer Systems Department

Today's Lecture

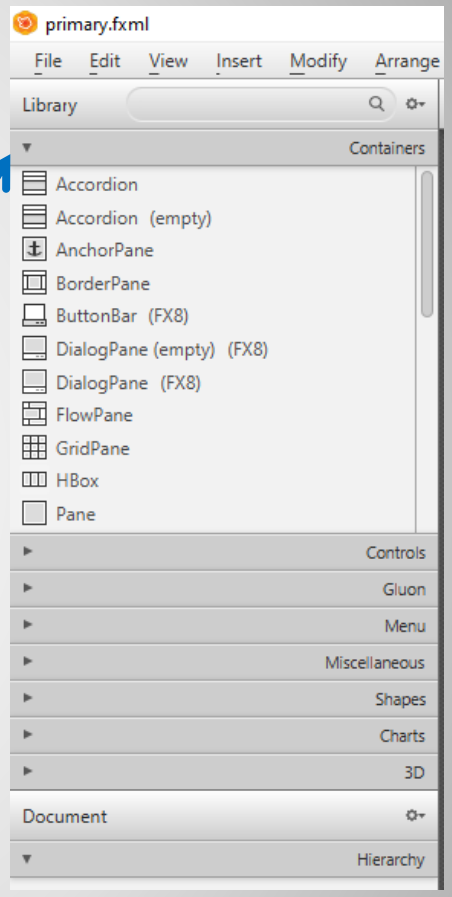
- JavaFX Graphics
  - 2D Shapes
  - Transforms

Pane

- JavaFX container used for absolute positioning of objects.
- This is a base class for other layout containers (VBox, BorderPane, etc…).
- The shapes we add to our application will be put in a Pane.

Open the Containers area

Pane container



Pane

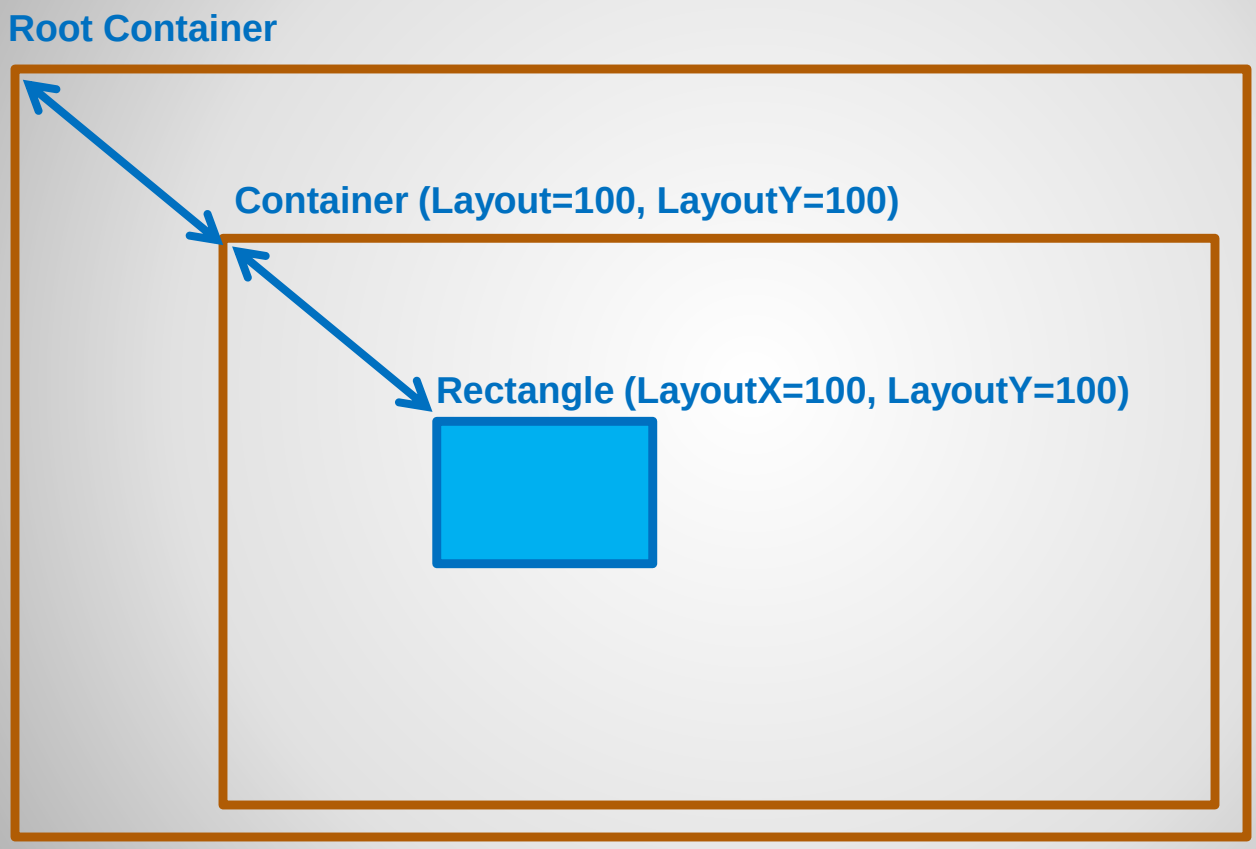
Pane - LayoutX and LayoutY

- LayoutX and LayoutY are JavaFX node properties that set the location of a node (JavaFX control).
- A node's LayoutX and LayoutY values are relative to the container that they are a child of.
- If you use a Pane, you can set these properties manually on its child nodes.

- Layouts that inherit from Pane (VBox, BorderPane etc…) will set the LayoutX and LayoutY values automatically on its children to achieve their specific layout objectives. For example, a VBox will align all controls one after another vertically.
- You can only change LayoutX and LayoutY on Pane or Group (parent manages position in other containers).

Pane – LayoutX and LayoutY

- The actual position of the rectangle is 200, 200.
- The rectangle's LayoutX and LayoutY are relative to its parent container.



Pane – LayoutX and LayoutY

JavaFX 2D Shapes

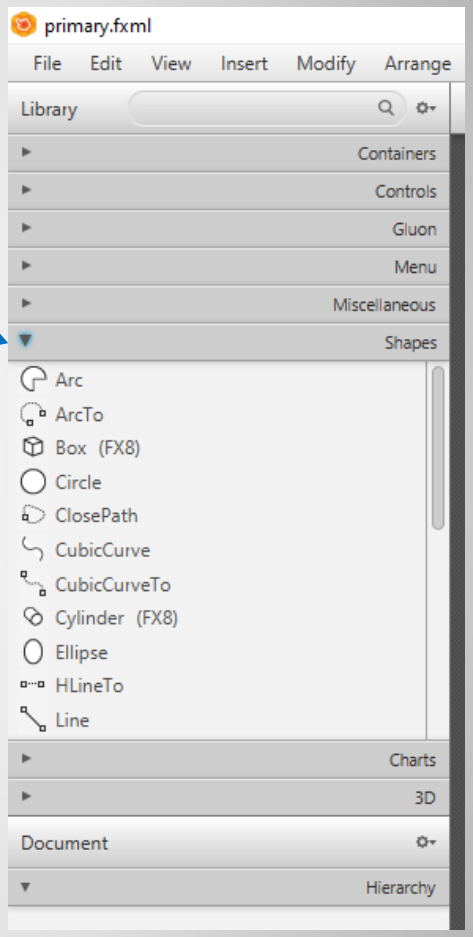
- You can add 2D Shapes to a JavaFX application in different ways.
- Ways to add 2D shapes:
  - Scene Builder
  - FXML
  - Java code

Add Shape - Scene Builder

- Drag and drop shapes on to the window

Open the Shapes area

Drag and drop the shapes you want



Add Shape - Scene Builder

Add Shape - FXML

- Add the shapes you want as children of the Pane.
- The FXML below adds a rectangle and a circle to the Pane.

```
<Pane … other attributes here…>
   <children>
      <Rectangle layoutX="336.0" layoutY="129.0" height="200.0"  width="200.0" />
      <Circle layoutX="149.0" layoutY="140.0" radius="100.0" />
   </children>
</Pane>
```

Note: If you look at the FXML generated by Scene Builder you will see that it adds additional attributes to the FXML.

Add Shape - FXML

Add Shape - Java Code

- The following code adds a rectangle to a pane.

```
import javafx.scene.layout.Pane;
import javafx.scene.shape.Rectangle;

public class PrimaryController {
   @FXML
   public Pane paneShapes;

   void myMethod() {
      Rectangle r = new Rectangle(10, 10, 50, 50);
      paneShapes.getChildren().add(r);
   }
}
```

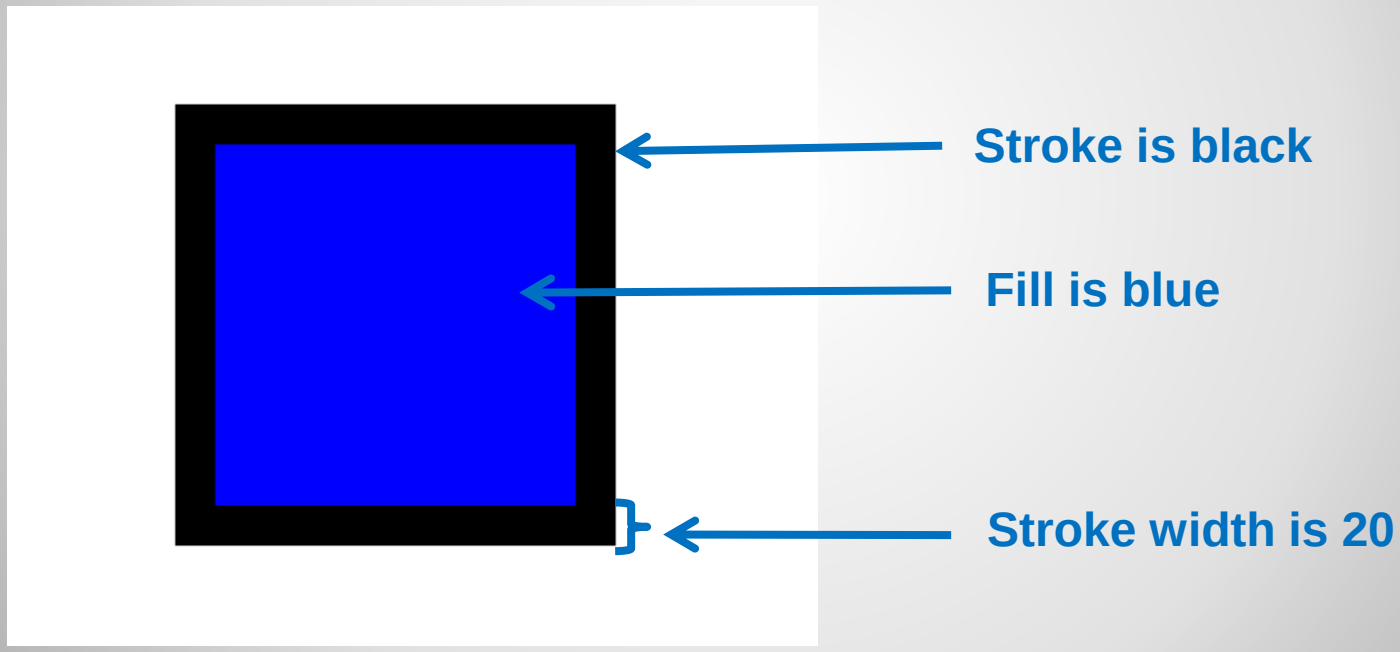
fx:id for the Pane that shapes will be added to

Create an instance of Rectangle

Add the Rectangle as a child of the Pane

Add Shape - Java Code

- Fill – The color that is used to fill the inside of the shape.
- Stroke – The border line that defines the shape.
- Stroke Width – The width of the stroke.

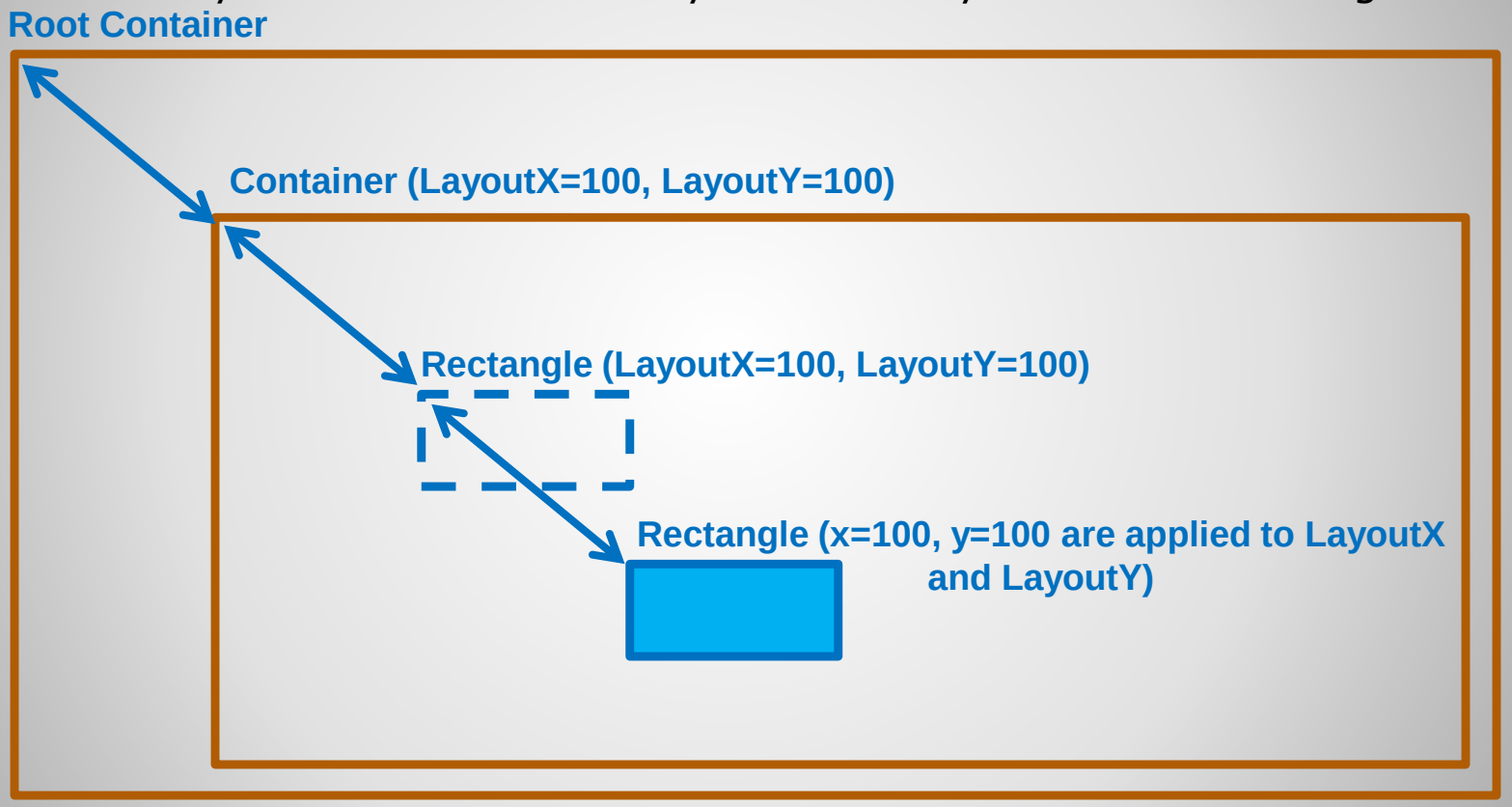


Common Shape Properties

- Rectangle – Set the x, y, width, height.
  - x and y are relative to LayoutX and LayoutY.
- Circle – Set the x, y, radius.
  - x and y are relative to LayoutX and LayoutY.
- Line – Set the startx, starty, endx, endy.
  - startx and starty are relative to LayoutX and LayoutY.
- Relative Positioning – The x and y for Rectangle and Circle are relative to LayoutX and LayoutY.
  - For example, if LayoutX=200 and x is 100 then the final x coordinate will be 300. The x position is applied relative to LayoutX. For a line, startx functions similarly to x on the Rectangle and Circle.

Some Common Shapes

- The actual position of the rectangle is 300, 300.
- The rectangle's LayoutX and LayoutY are relative to its parent container.
- The x and y are relative to the LayoutX and LayoutY of the rectangle.



Rectangle – x and y

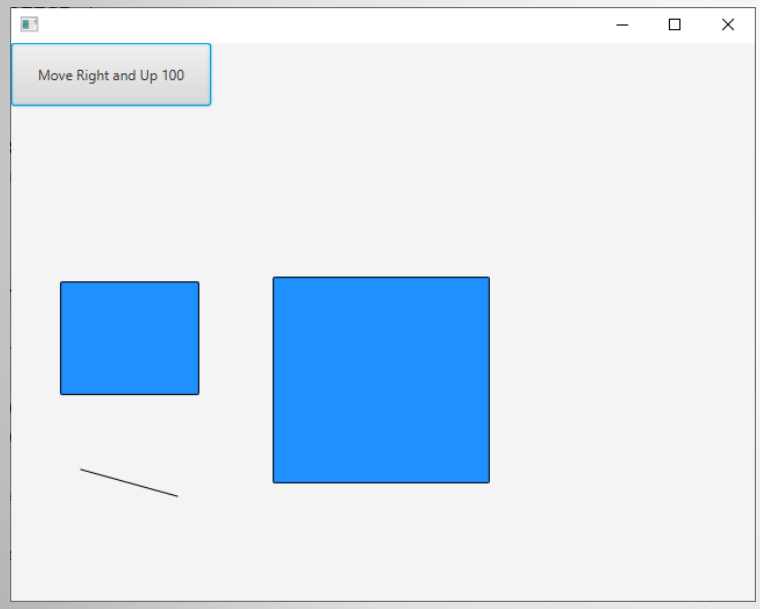
Transformations

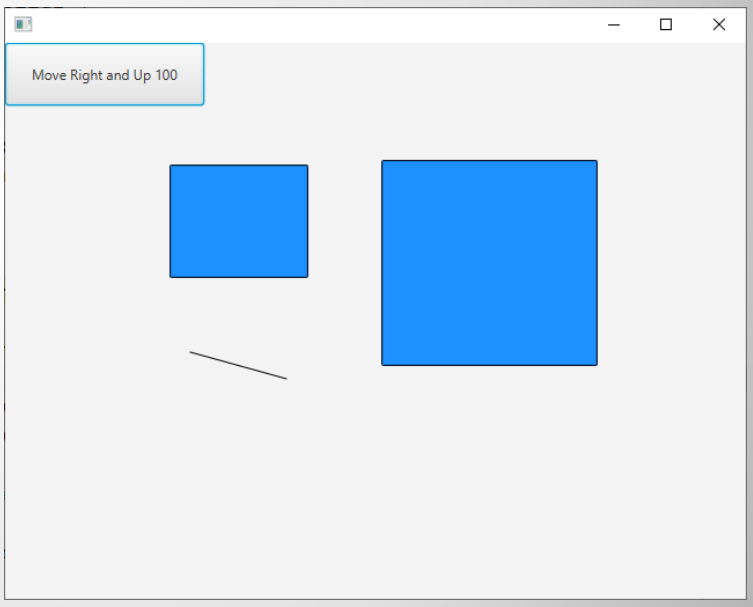
- Change graphical objects. Uses mathematical matrix operations behind the scenes.

- We will cover the following transformations:
  - Translate
  - Scale
  - Rotate

Transformations

- Translate
- Moves a graphical object to a new location.
- For example, the objects are moved to the right by 100 and up by 100.





Translate

Translate One Node

- The Java Translate class is used for translations.
- Each node keeps a list of transformations that are applied to it.
- The example below adds a translate transformation to a rectangle node.

```
Translate translate = new Translate();

translate.setX(100);
translate.setY(-100);

myRectangle.getTransforms().add(translate);
```

Positive x moves object to the right

Negative y moves object up

Add the Translate transformation instance to the rectangle node (assmes there is a rectangle with an fx:id of myRectangle)

Translate All Nodes

- You can apply a translation to all nodes in a Pane.
- The example below applies the translation to all nodes of the Pane named paneMain (this assumes you set the fx:id of the Pane to paneMain).

```
Translate translate = new Translate();

translate.setX(100);
translate.setY(-100);

for (Node n : paneMain.getChildren())
{
   n.getTransforms().add(translate);
}
```

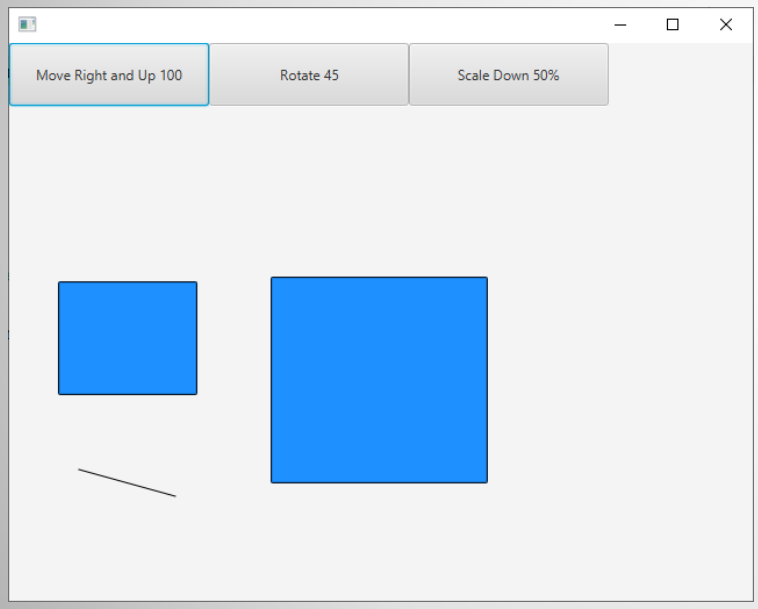
Positive x moves object to the right

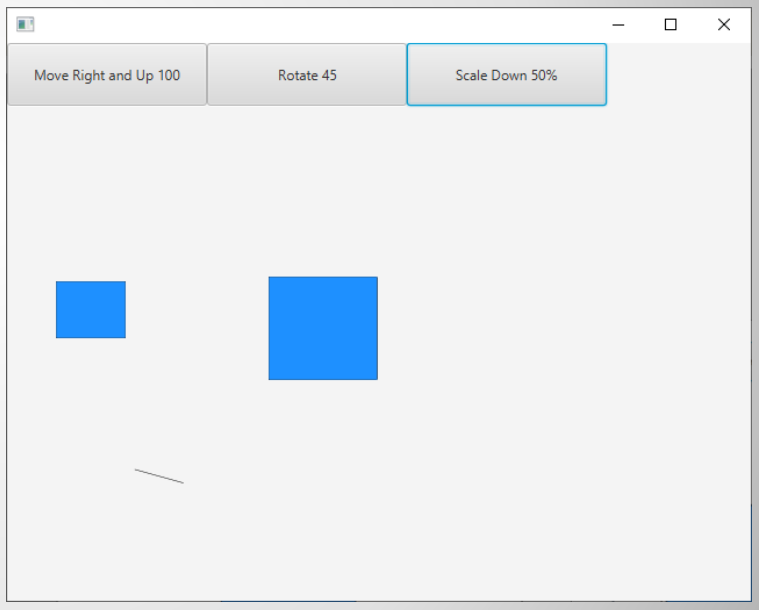
Negative y moves object up

Apply transform to all nodes in the Pane (assumes the pane has an fx:id of paneMain)

Add the Translate transformation instance to the current node

- Scale
- Change an objects size.
- For example, the objects are being scaled down by 50%.





Scale

Scale

- The Java Scale class is used for scaling.
- Use setPivotX to prevent the shape from "moving" when scaling. The "moving" occurs because of scaling at the center.

```
Scale scale = new Scale();

scale.setX(.5);
scale.setY(.5);
scale.setPivotX(rect.getX());
scale.setPivotY(rect.getY());

rect.getTransforms().add(scale);

for (Node n : paneMain.getChildren()) {
   n.getTransforms().add(scale);
}
```

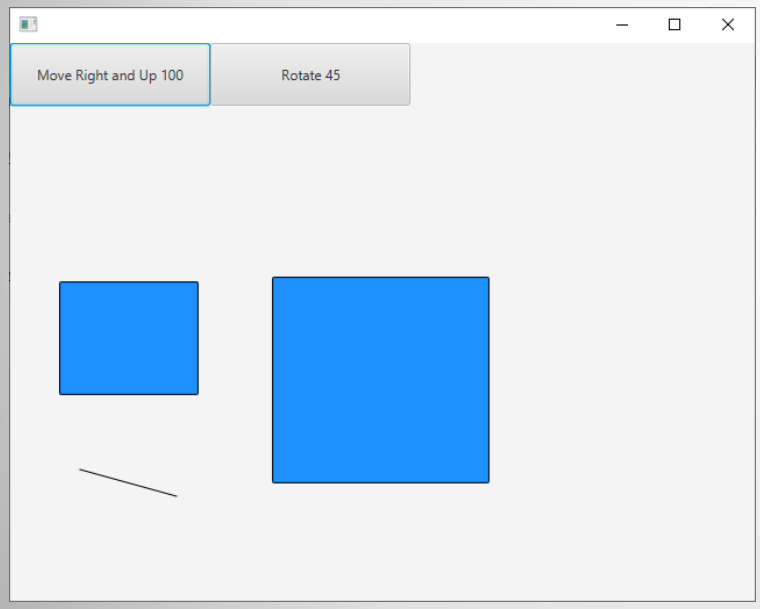
Scale the x and y down by 50%

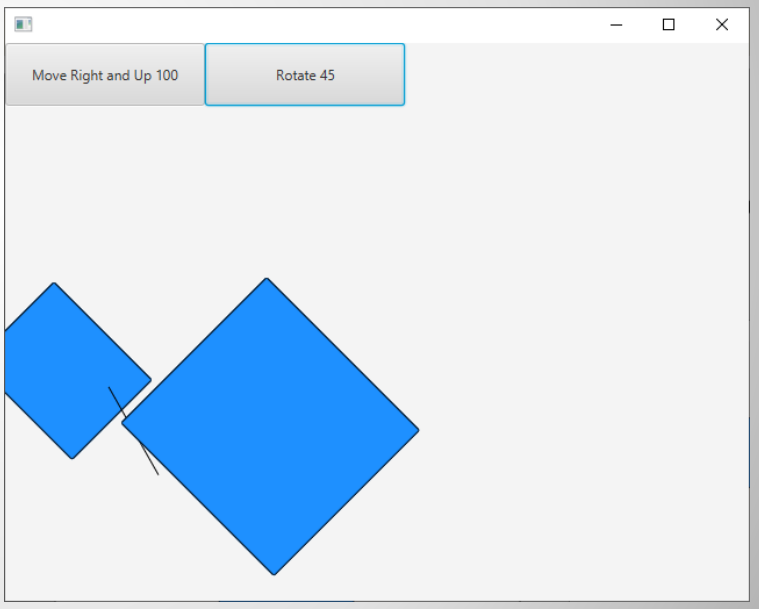
Scale from the top/left location of the rectangle (not the center)

Add the Scale transformation only to the rectangle

Alternatively, you can add the Scale transformation to all node's in a Pane (assumes paneMain exists)

- Rotate
- Turns an object on its axis.
- For example, the objects are being rotated 45 degrees.





Rotate

Rotate

- The Java Rotate class is used for rotations.

```
Rotate rotate = new Rotate();

rotate.setAngle(45);

for (Node n : paneMain.getChildren())
{
   n.getTransforms().add(rotate);
}
```

Rotate 45 degrees

Add the Translate transformation instance to the current node

Mouse Click Event (on Pane)

- You can add event handlers for mouse clicks on a Pane.
- Here is a sample event handler (should be defined on the controller class):

```
@FXML
private void handleOnMouseClicked(MouseEvent event)
{
   System.out.println("Mouse click handled: " + event.toString());
}
```

- The MouseEvent parameter contains information about where the mouse click happened (x and y positions, which mouse buttons are pressed etc…).
- You can set this method as the event handler for On Mouse Clicked (for the Pane) in Scene Builder.

Handling Mouse Clicks on Pane

End of Slides