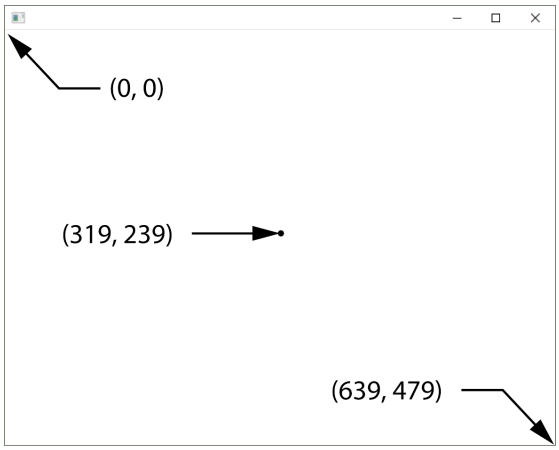
Topic 6: JavaFX

Part c: Drawing Shapes

Drawing Shapes

Screen Coordinate System

- The location of each pixel in an application's window is identified with an *X* coordinate and a *Y* coordinate.
  - The *X* coordinates increase from left to right, and the *Y* coordinates increase from top to bottom.
  - When drawing a line or shape on a component, you must indicate its position using *X* and *Y* coordinates.



The `Shape` Class and its Subclasses

- The `Shape` class (in the `javafx.scene.shape` package) provides the basic functionality for drawing shapes.
- The `Shape` class has several subclasses, each of which draws a specific shape.

Class	Description
`Line`	Draws a line from one point to another.
`Circle`	Draws a circle with its center point located at a specific coordinate, and with a specified radius.
`Rectangle`	Draws a rectangle with a specified width and height.
`Ellipse`	Draws an ellipse with a specified center point, *X* radius, and *Y* radius.
`Arc`	Draws an arc, which is a partial ellipse.
`Polygon`	Draws a polygon with vertices at specified locations.
`Polyline`	Draws a polyline with vertices at specified locations.
`Text`	Draws a string at a specified location.

The `Shape` Class and its Subclasses (2)

- You draw shapes with these classes by following this general procedure:

1. Create an instance of the desired shape class.
2. Repeat step 1 for each shape that you want to draw.
3. Add all of the shape objects that you created to a container.
4. Add the container to the scene graph.

Shape Subclasses

Line, Circle, Rectangle, etc…

The `Line` Class

- Constructors

Constructor	Description
`Line()`	Creates an empty line. Call the Line class's `setStartX` and `setStartY` methods to establish the line's starting point, and the `setEndX` and `setEndY` methods to establish the line's ending point
`Line(`*`startX, startY, endX, endY`*`)`	All of the arguments are doubles. The *`startX`* and *`startY`* arguments are the *X* and *Y* coordinates for the line's starting point. The *`endX`* and *`endY`* arguments are the *X* and *Y* coordinates for the line's ending point.

The `Line` Class (2)

- The following statement creates a line starting at (80, 120) and ending at (400, 520):

```
Line myLine = new Line(80, 120, 400, 520);
```

- No-arg constructor: This creates a line starting at (0, 0) and ending at (200, 200):

```
Line myLine = new Line();
myLine.setStartX(0);
myLine.setStartY(0);
myLine.setEndX(200);
myLine.setEndY(200);
```

Changing the Stroke Color

- The default color of lines and other shapes is black.
- To change a shape's color call the `setStroke` method, which is inherited from the `Shape` class.
- The general format is

```
setStroke(color)
```

- The color argument is usually a `Color` class constant, such as `Color.RED`, `Color.Blue`, etc.
- The `Color` class is in the `javafx.scene.paint` package
- For example:

```
Line myLine = new Line(80, 120, 400, 520);
myLine.setStroke(Color.RED);
```

The `Circle` Class

- The following code creates a circle with its center point at (75, 100), a radius of 50, and filled with the color red:

```
Circle myCircle = new Circle();
myCircle.setCenterX(75);
myCircle.setCenterY(100);
myCircle.setRadius(50);
myCircle.setFill(Color.RED);
```

- The following code draws a black outline of a circle with no fill color:

```
Circle myCircle = new Circle(75, 100, 50);
myCircle.setFill(null);
myCircle.setStroke(Color.BLACK);
```

The `Rectangle` Class

- The following statement creates a rectangle with its upper-left corner at (200, 100), with a width of 75 and a height of 150:

```
Rectangle myRectangle = new
                Rectangle(200, 100, 75, 150);
```

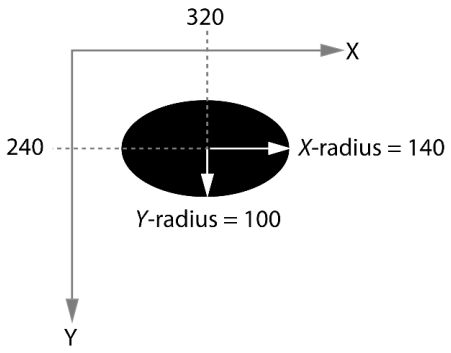
- The following code creates a rectangle with its upper-left corner at (10, 20), a width of 50, a height of 100, and filled with the color dark green:

```
Rectangle myRectangle = new Rectangle();
myRectangle.setX(10);
myRectangle.setY(20);
myRectangle.setWidth(50);
myRectangle.setHeight(100);
myRectangle.setFill(Color.DARKGREEN);
```

The `Ellipse` Class

- The following statement creates an ellipse its center located at (320, 240), an X-radius of 140 pixels, and a Y-radius of 100:

```
Ellipse myEllipse = new
        Ellipse(320, 240, 140, 100);
```



The `Ellipse` Class

- The following code creates an ellipse with its center point at (125, 100), an X-radius of 130, a Y-radius of 90, no fill color, and a stroke color of black:

```
Ellipse myEllipse = new Ellipse();
myEllipse.setCenterX(125);
myEllipse.setCenterY(100);
myEllipse.setRadiusX(130);
myEllipse.setRadiusY(90);
myEllipse.setFill(null);
myellipse.setStroke(Color.BLACK);
```

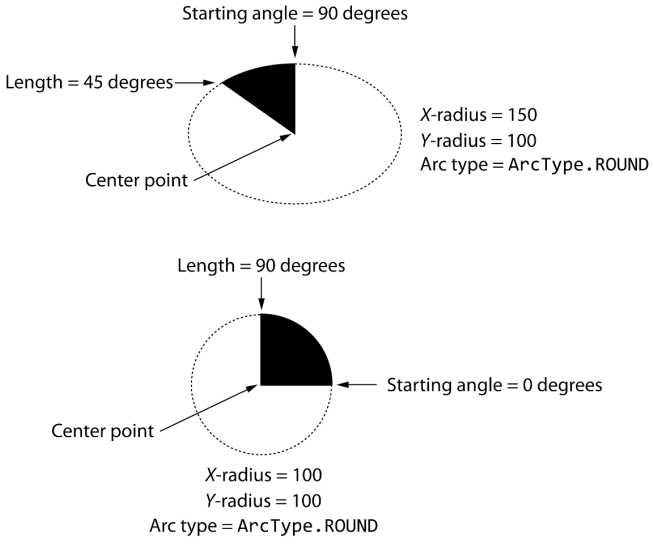
The `Arc` Class

- Constructors

Constructor	Description
`Arc()`	Creates an empty arc. Call the Arc class's `setCenterX` and `setCenterY` methods to establish the arc's center point, the `setRadiusX` method to establish the arc's radius along the *X* axis, the `setRadiusY` method to establish the arc's radius along the *Y* axis, the `setStartAngle` method to establish the arc's starting angle (in degrees), and the `setLength` method to establish the arc's angular extent (in degrees).
`Arc(`*`centerX`*`, `*`centerY`*`, `*`radiusX`*`, `*`radiusY`*`, `*`startAngle`*`, `*`length`*`)`	The arguments are doubles. Creates an arc at the specified center point, with the specified *X* and *Y* radii. The arc begins at the angle specified by *`startAngle`*, and extends counterclockwise. The *`length`* argument specifies the number of degrees that the arc extends from its starting angle.

The `Arc` Class (2)

- Arc Properties



The `Arc` Class (3)

- Types of Arcs

Type	Description
`ArcType.CHORD`	This is the default arc type. A straight line will be drawn from one endpoint of the arc to the other endpoint.
`ArcType.ROUND`	Straight lines will be drawn from each endpoint to the arc's center point. As a result, the arc will be shaped like a pie slice.
`ArcType.OPEN`	No lines will connect the endpoints. Only the arc will be drawn.

The `Arc` Class (4)

- The following code creates an arc with its center point at (160, 120), an X-radius of 100, a Y-radius of 100, beginning at 0 degrees, with a length of 34 degrees, filled with the color red. The arc will resemble a pie-slice because the type of arc is `ArcType.ROUND`:

```
Arc myArc = new Arc(160.0, 120.0, 100, 100.0, 0.0, 45.0);
myArc.setFill(Color.RED);
myArc.setType(ArcType.ROUND);
```

The `Polygon` Class

- Example

```
Polygon diamond = new Polygon(160.0, 20.0,  // Top
                              300.0, 120.0, // Right
                              160.0, 220.0, // Bottom
                              20.0,  120.0);// Left
```

The `Text` Class

- Constructors

Constructor	Description
`Text()`	Creates an empty Text object. Call the Text class's `setX` and `setY` methods to establish the `Text` object's location, and the `setText` method to establish the string that the object should display.
`Text(x, y, text)`	The *x* and *y* arguments, which are doubles, are the *XY* coordinates of the object's bottom-left corner. The *`text`* argument is the string that the object will display.
`Text(text)`	The *`text`* argument is the string that the object will display. Call the `Text` class's `setX` and `setY` methods to establish the `Text` object's location.

The `Text` Class (2)

- The following code draws the string “Hello World”, starting at the coordinates 100, 50:

```
Text myText = new Text(100.0, 50.0, "Hello World");
```

- You can set the font with the `setFont` method. This method accepts a `Font` object as its argument.
- The `Font` class is in the `javafx.scene.text` package.
- When you instantiate the `Font` class, you pass the name of a font and the font’s size, in points, as arguments to the constructor:

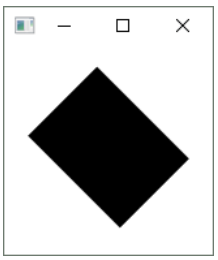
```
myText.setFont(new Font("Serif", 36));
myText.setStroke(Color.BLACK);
myText.setFill(Color.RED);
```

Rotating Nodes

- The `Node` class provides a method named `setRotate` that rotates a node about its center.
- Because the `setRotate` method is in the `Node` class, it can be used to rotate any node in your scene graph

```
// Constants for the rectangle
final double X = 30.0, Y = 40.0;
final double WIDTH = 100.00, HEIGHT = 75.0;
final double ANGLE = 45.0;

// Create a rectangle.
Rectangle box = new Rectangle(X, Y, WIDTH, HEIGHT);
box.setRotate(ANGLE);
```



Scaling Nodes

- The `Node` class also provides methods named `setScaleX` and `setScaleY` that scale a node in its X and Y dimensions.
- Because these methods are in the `Node` class, they can be used to scale any node in your scene graph.

Scaling Nodes (2)

```
// Constants for the text
final double X1 = 30.0, Y1 = 100.0;
final double X2 = 30.0, Y2 = 130.0;
final double X3 = 30.0, Y3 = 150.0;
final double FONT_SIZE = 38;
final double SCALE_HALF = 0.5;
final double SCALE_QTR = 0.25;

Text text1 = new Text(X1, Y1, "Hello World");
text1.setFont(new Font("SansSerif", FONT_SIZE));

Text text2 = new Text(X2, Y2, "Hello World");
text2.setFont(new Font("SansSerif", FONT_SIZE));
text2.setScaleX(SCALE_HALF);
text2.setScaleY(SCALE_HALF);

Text text3 = new Text(X3, Y3, "Hello World");
text3.setFont(new Font("SansSerif", FONT_SIZE));
text3.setScaleX(SCALE_QTR);
text3.setScaleY(SCALE_QTR);
```