

JavaFX

Basics

Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All



Motivations

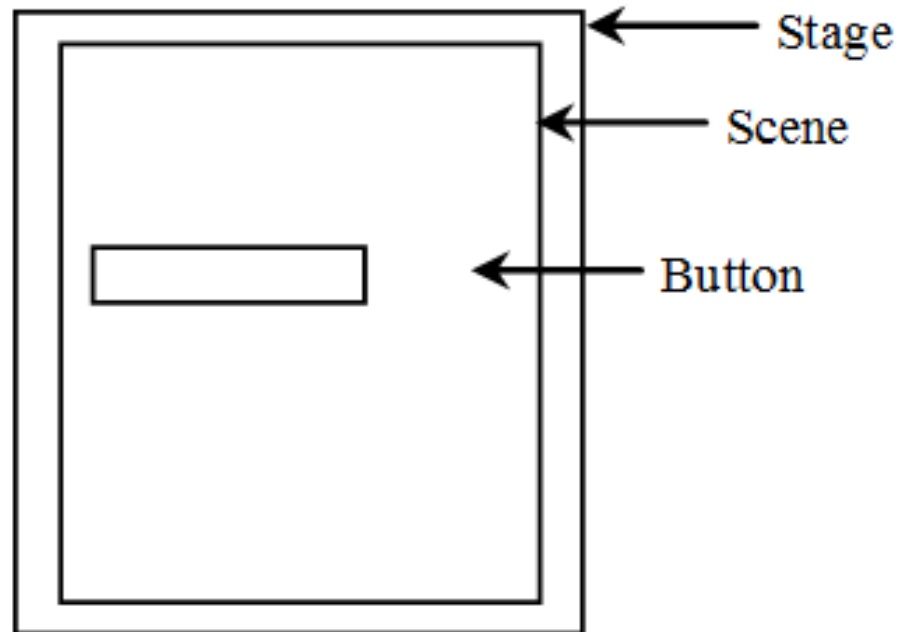
- ❖ **JavaFX** is a new framework for developing Java graphical user interface (**GUI**) programs.
- ❖ The **JavaFX API** is an **excellent** example of how the **OO** principle is applied.
- ❖ This chapter serves two purposes:
 - First, it presents the basics of **JavaFX** programming.
 - Second, it uses **JavaFX** to demonstrate **OOP**.
- ❖ Specifically, this chapter introduces the framework of **JavaFX** and discusses **JavaFX GUI** components and their relationships.

JavaFX vs Swing and AWT

- ❖ When Java was introduced, the **GUI** classes were bundled in a library known as the ***Abstract Windows Toolkit (AWT)***.
 - **AWT** is fine for developing simple graphical user interfaces, but not for developing comprehensive **GUI**.
 - In addition, **AWT** is prone to platform-specific bugs.
- ❖ The **AWT** components were replaced by a more robust, versatile, and flexible library known as ***Swing***.
 - **Swing** components depend less on the target platform and use less of the native **GUI** resource.
- ❖ With the release of Java 8, **Swing** is replaced by a completely new **GUI** platform known as ***JavaFX***.

Basic Structure of JavaFX

- ❖ Application
- ❖ Override the **start (Stage)** method
- ❖ **Stage, Scene, and Nodes**



```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.stage.Stage;
```

```
public class MyJavaFX extends Application {
    @Override // Override the start method in the Application class
    public void start(Stage primaryStage) {
        // Create a button and place it in the scene
        Button btOK = new Button("OK");
        Scene scene = new Scene(btOK, 200, 250);
        primaryStage.setTitle("MyJavaFX"); // Set the stage title
        primaryStage.setScene(scene); // Place the scene in the stage
        primaryStage.show(); // Display the stage
    }
```

Extend Application

Override start

Create a button

Create a scene

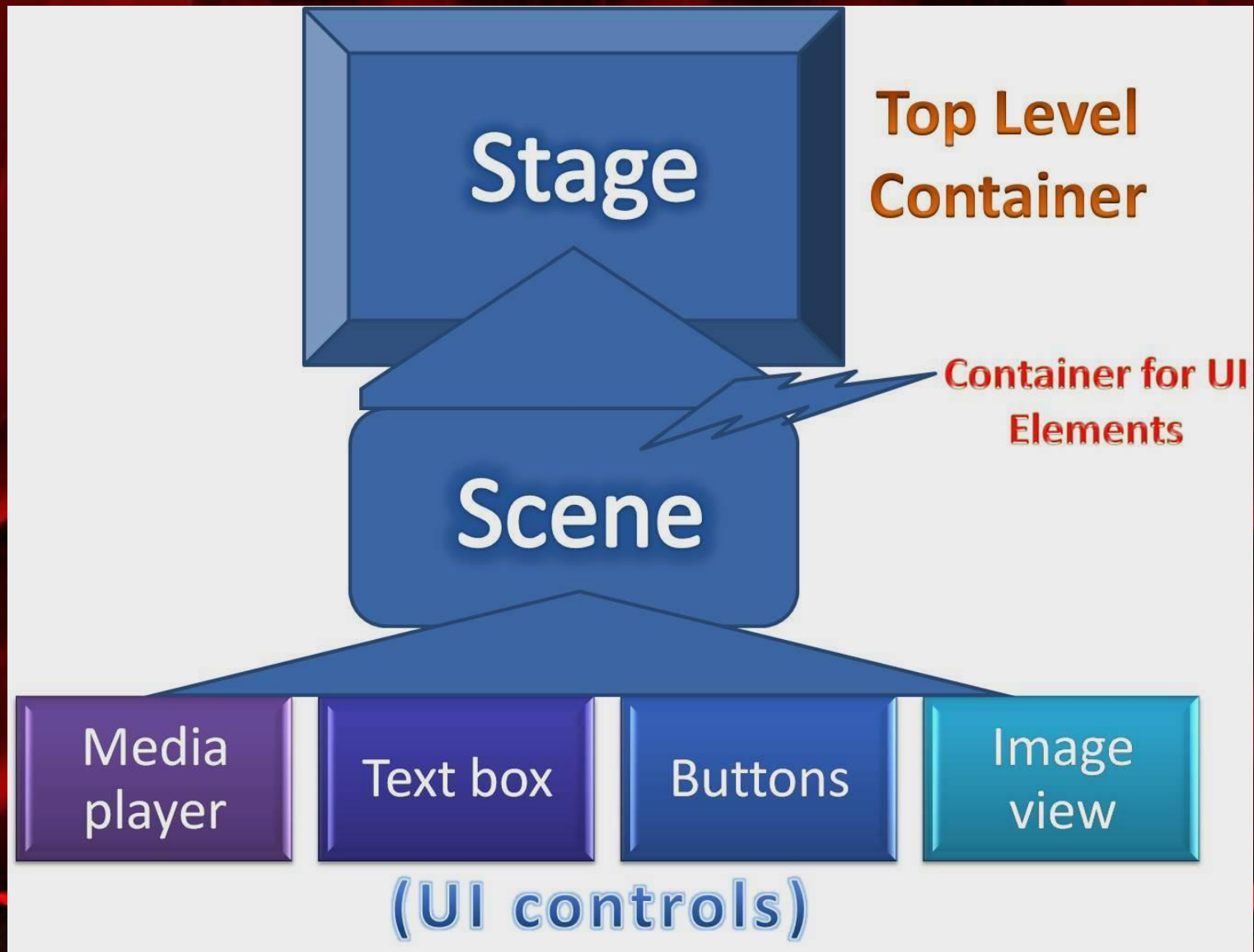
Set a scene

Display stage

```
/**
 * The main method is only needed for the IDE
 * JavaFX support. Not needed for running from
 */
public static void main(String[] args) {
    launch(args);
}
```

Launch application



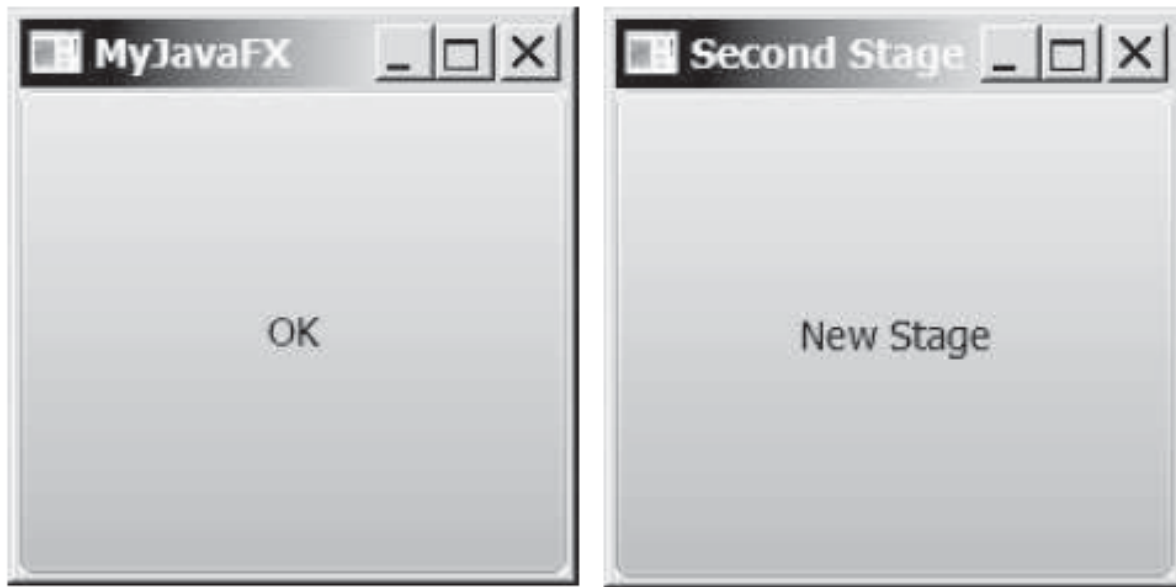


```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.stage.Stage;
```

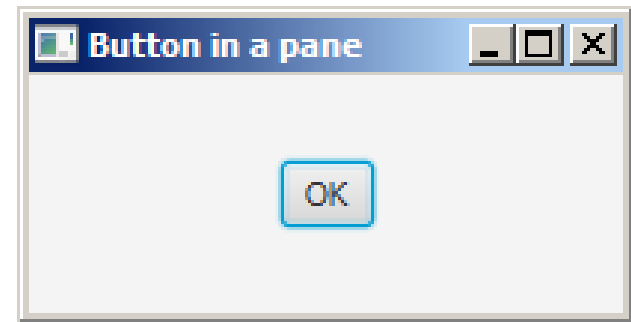
```
public class MultipleStageDemo extends Application {
    public void start(Stage primaryStage) {
        Scene scene = new Scene(new Button("OK"), 200, 250);
        primaryStage.setTitle("MyJavaFX");
        primaryStage.setScene(scene);
        primaryStage.show();
        // Create a new stage
        Stage stage = new Stage();
        stage.setTitle("Second Stage");
        stage.setScene(new Scene(new Button("New Stage"), 100, 100));
        stage.show(); // Display the stage
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

Result is Two Stages/Windows




```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;
```



```
public class ButtonInPane extends Application {
    public void start(Stage primaryStage) {
        StackPane pane = new StackPane();
        pane.getChildren().add(new Button("OK"));
        Scene scene = new Scene(pane, 200, 50);
        primaryStage.setTitle("Button in a pane");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```



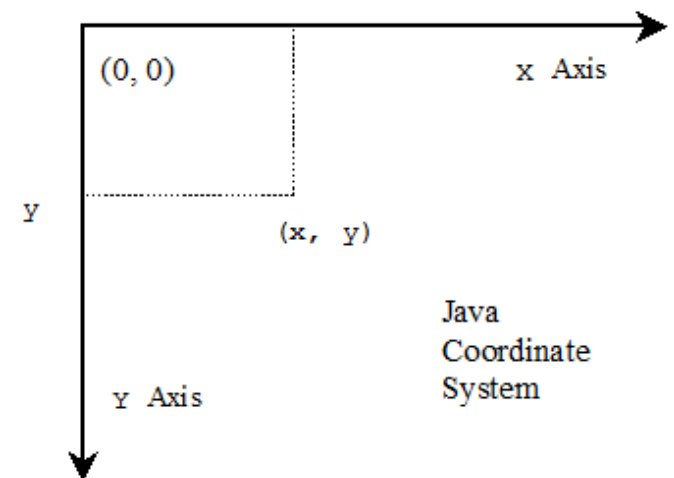
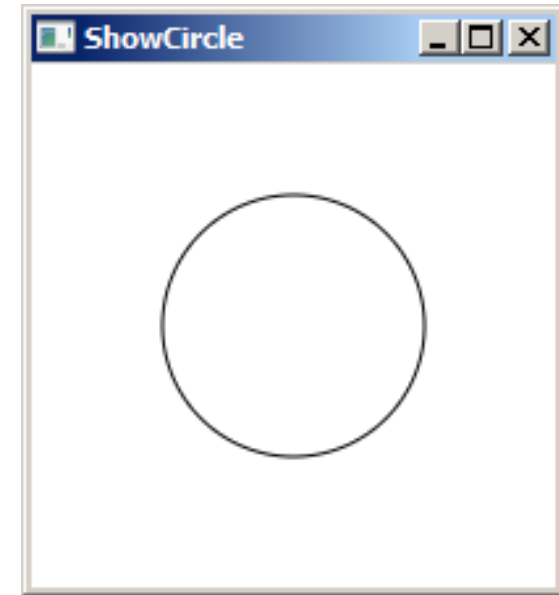
Display a Shape

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.stage.Stage;

public class ShowCircle extends Application {
    public void start(Stage primaryStage) {
        Circle circle = new Circle();
        circle.setCenterX(100);
        circle.setCenterY(100);
        circle.setRadius(50);
        circle.setStroke(Color.BLACK);
        circle.setFill(null);

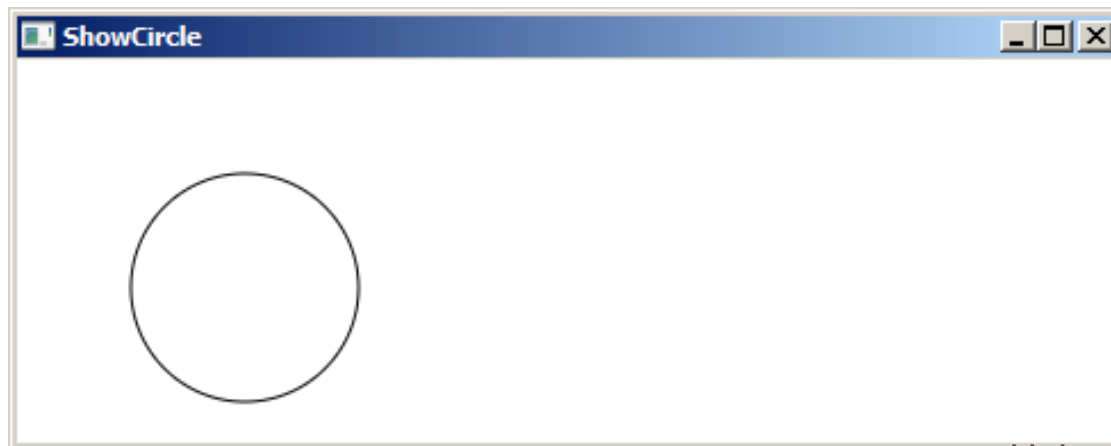
        Pane pane = new Pane();
        pane.getChildren().add(circle);

        Scene scene = new Scene(pane, 200, 200);
        primaryStage.setTitle("ShowCircle");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
}
```



Binding Properties

- ❖ JavaFX introduces a new concept called *binding property* that enables a *target object* to be bound to a *source object*.
 - If the value in the source object changes, the target property is also changed automatically.
- ❖ The target object is simply called a *binding object* or a *binding property*.



Binding Properties

```
public void start(Stage primaryStage) {  
    Pane pane = new Pane();  
    Circle circle = new Circle();  
    circle.centerXProperty().bind(pane.widthProperty().divide(2));  
    circle.centerYProperty().bind(pane.heightProperty().divide(2));  
    circle.setRadius(50);  
    circle.setStroke(Color.BLACK);  
    circle.setFill(Color.WHITE);  
    pane.getChildren().add(circle);  
  
    Scene scene = new Scene(pane, 200, 200);  
    primaryStage.setTitle("ShowCircleCentered");  
    primaryStage.setScene(scene);  
    primaryStage.show();  
}
```

Binding Property:

getter, setter, and property getter

```
public class SomeClassName {  
  
    private PropertyType x;  
  
    /** Value getter method */  
    public propertyValueType getX() { ... }  
  
    /** Value setter method */  
    public void setX(propertyValueType value) { ... }  
  
    /** Property getter method */  
    public PropertyType xProperty() { ... }  
}
```

(a) x is a binding property

```
public class Circle {  
  
    private DoubleProperty centerX;  
  
    /** Value getter method */  
    public double getCenterX() { ... }  
  
    /** Value setter method */  
    public void setCenterX(double value) { ... }  
  
    /** Property getter method */  
    public DoubleProperty centerXProperty() { ... }  
}
```

(b) centerX is binding property

Binding Property

❖ **JavaFX** defines binding properties for primitive types and strings:

Type	Binding Property Type
double	DoubleProperty
float	FloatProperty
long	LongProperty
int	IntegerProperty
boolean	BooleanProperty
String	StringProperty



```
import javafx.beans.property.DoubleProperty;  
import javafx.beans.property.SimpleDoubleProperty;
```

```
public class BindingDemo {  
    public static void main(String[] args) {  
        DoubleProperty d1 = new SimpleDoubleProperty(1);  
        DoubleProperty d2 = new SimpleDoubleProperty(2);  
        d1.bind(d2);  
        System.out.println("d1 is " + d1.getValue()  
            + " and d2 is " + d2.getValue());  
        d2.setValue(70.2);  
        System.out.println("d1 is " + d1.getValue()  
            + " and d2 is " + d2.getValue());  
    }  
}
```

d1 is 2.0 and d2 is 2.0

d1 is 70.2 and d2 is 70.2

Common Properties and Methods for Nodes

❖ The abstract **Node** class defines many properties and methods that are common to all nodes.

- **style**: set a **JavaFX CSS** style

```
circle.setStyle("-fx-stroke: black; -fx-fill: red;");
```

- **rotate**: Rotate a node

```
button.setRotate(80);
```

The Color Class

javafx.scene.paint.Color

-red: double
-green: double
-blue: double
-opacity: double

+Color(r: double, g: double, b: double, opacity: double)

+brighter(): Color

+darker(): Color

+color(r: double, g: double, b: double): Color

+color(r: double, g: double, b: double, opacity: double): Color

+rgb(r: int, g: int, b: int): Color

+rgb(r: int, g: int, b: int, opacity: double): Color

The red value of this Color (between 0.0 and 1.0).

The green value of this Color (between 0.0 and 1.0).

The blue value of this Color (between 0.0 and 1.0).

The opacity of this Color (between 0.0 and 1.0).

Creates a Color with the specified red, green, blue, and opacity values.

Creates a Color that is a brighter version of this Color.

Creates a Color that is a darker version of this Color.

Creates an opaque Color with the specified red, green, and blue values.

Creates a Color with the specified red, green, blue, and opacity values.

Creates a Color with the specified red, green, and blue values in the range from 0 to 255.

Creates a Color with the specified red, green, and blue values in the range from 0 to 255 and a given opacity.

```
Color color = new Color(0.25, 0.14, 0.333, 0.51);
```



The **Font** Class

javafx.scene.text.Font

```
-size: double  
-name: String  
-family: String  
  
+Font(size: double)  
+Font(name: String, size: double)  
+font(name: String, size: double)  
+font(name: String, w: FontWeight, size: double)  
+font(name: String, w: FontWeight, p: FontPosture, size: double)  
+getFamilies(): List<String>  
+getFontNames(): List<String>
```

The size of this font.

The name of this font.

The family of this font.

Creates a **Font** with the specified size.

Creates a **Font** with the specified full font name and size.

Creates a **Font** with the specified name and size.

Creates a **Font** with the specified name, weight, and size.

Creates a **Font** with the specified name, weight, posture, and size.

Returns a list of font family names.

Returns a list of full font names including family and weight.

```
Font font1 = new Font("SansSerif", 16);  
Font font2 = Font.font("Times New Roman", FontWeight.BOLD,  
    FontPosture.ITALIC, 12);
```



```

public class FontDemo extends Application {
    @Override // Override the start method in the Application class
    public void start(Stage primaryStage) {
        // Create a pane to hold the circle
        Pane pane = new StackPane();

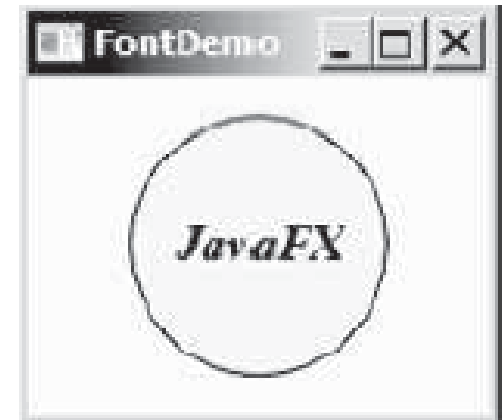
        // Create a circle and set its properties
        Circle circle = new Circle();
        circle.setRadius(50);
        circle.setStroke(Color.BLACK);
        circle.setFill(new Color(0.5, 0.5, 0.5, 0.1));
        pane.getChildren().add(circle); // Add circle to the pane

        // Create a label and set its properties
        Label label = new Label("JavaFX");
        label.setFont(Font.font("Times New Roman",
            FontWeight.BOLD, FontPosture.ITALIC, 20));
        pane.getChildren().add(label);

        // Create a scene and place it in the stage
        Scene scene = new Scene(pane);
        primaryStage.setTitle("FontDemo"); // Set the stage title
        primaryStage.setScene(scene); // Place the scene in the stage

        primaryStage.show(); // Display the stage
    }
}

```



The Image, ImageView Class

javafx.scene.image.Image

-error: ReadOnlyBooleanProperty
-height: ReadOnlyBooleanProperty
-width: ReadOnlyBooleanProperty
-progress: ReadOnlyBooleanProperty
+Image(filenameOrURL: String)

Indicates whether the image is loaded correctly?
The height of the image.
The width of the image.
The approximate percentage of image's loading that is completed.
Creates an Image with contents loaded from a file or a URL

javafx.scene.image.ImageView

-fitHeight: DoubleProperty
-fitWidth: DoubleProperty
-x: DoubleProperty
-y: DoubleProperty
-image: ObjectProperty<Image>
+ImageView()
+ImageView(image: Image)
+ImageView(filenameOrURL:String)

The height of the bounding box within which the image is resized to fit.
The width of the bounding box within which the image is resized to fit.
The x-coordinate of the ImageView origin.
The y-coordinate of the ImageView origin.
The image to be displayed in the image view.
Creates an ImageView.
Creates an ImageView with the specified image.
Creates an ImageView with image loaded from the specified file or URL

Layout Panes

❖ **JavaFX** provides many types of panes for organizing nodes in a container.

<i>Class</i>	<i>Description</i>
Pane	Base class for layout panes. It contains the getChildren() method for returning a list of nodes in the pane.
StackPane	Places the nodes on top of each other in the center of the pane.
FlowPane	Places the nodes row-by-row horizontally or column-by-column vertically.
GridPane	Places the nodes in the cells in a two-dimensional grid.
BorderPane	Places the nodes in the top, right, bottom, left, and center regions.
HBox	Places the nodes in a single row.
VBox	Places the nodes in a single column.



FlowPane

`javafx.scene.layout.FlowPane`

-alignment: `ObjectProperty<Pos>`
-orientation:
 `ObjectProperty<Orientation>`
-hgap: `DoubleProperty`
-vgap: `DoubleProperty`

+`FlowPane()`
+`FlowPane(hgap: double, vgap: double)`
+`FlowPane(orientation: ObjectProperty<Orientation>)`
+`FlowPane(orientation: ObjectProperty<Orientation>, hgap: double, vgap: double)`

The overall alignment of the content in this pane (default: `Pos.LEFT`).
The orientation in this pane (default: `Orientation.HORIZONTAL`).

The horizontal gap between the nodes (default: 0).

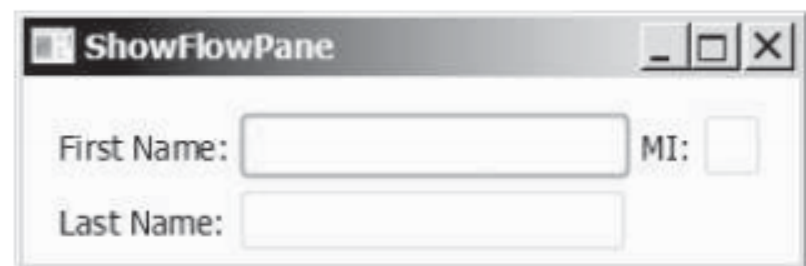
The vertical gap between the nodes (default: 0).

Creates a default `FlowPane`.

Creates a `FlowPane` with a specified horizontal and vertical gap.

Creates a `FlowPane` with a specified orientation.

Creates a `FlowPane` with a specified orientation, horizontal gap and vertical gap.



GridPane

javafx.scene.layout.GridPane

-alignment: ObjectProperty<Pos>
-gridLinesVisible: BooleanProperty
-hgap: DoubleProperty
-vgap: DoubleProperty

+GridPane()
+add(child: Node, columnIndex: int, rowIndex: int): void
+addColumn(columnIndex: int, children: Node...): void
+addRow(rowIndex: int, children: Node...): void
+getColumnIndex(child: Node): int
+setColumnIndex(child: Node, columnIndex: int): void
+getRowIndex(child: Node): int
+setRowIndex(child: Node, rowIndex: int): void
+setHalignment(child: Node, value: HPos): void
+setValignment(child: Node, value: VPos): void

The overall alignment of the content in this pane (default: Pos.LEFT).

Is the grid line visible? (default: false)

The horizontal gap between the nodes (default: 0).

The vertical gap between the nodes (default: 0).

Creates a GridPane.

Adds a node to the specified column and row.

Adds multiple nodes to the specified column.

Adds multiple nodes to the specified row.

Returns the column index for the specified node.

Sets a node to a new column. This method repositions the node.

Returns the row index for the specified node.

Sets a node to a new row. This method repositions the node.

Sets the horizontal alignment for the child in the cell.

Sets the vertical alignment for the child in the cell.

BorderPane

javafx.scene.layout.BorderPane

-top: ObjectProperty<Node>
-right: ObjectProperty<Node>
-bottom: ObjectProperty<Node>
-left: ObjectProperty<Node>
-center: ObjectProperty<Node>

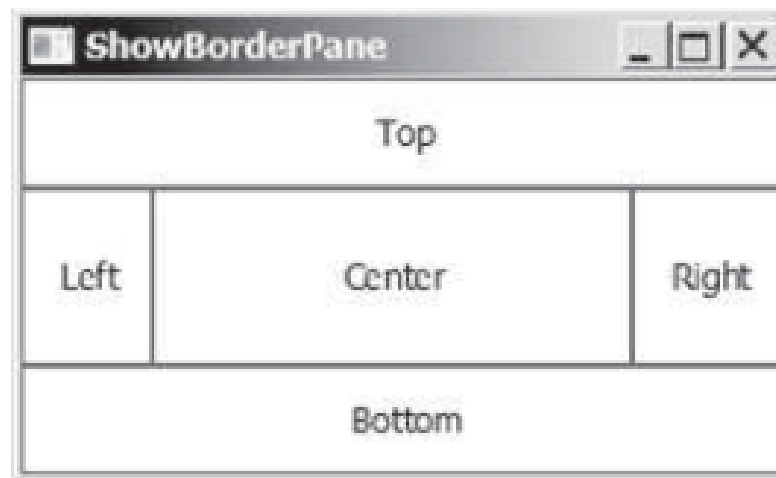
+BorderPane()

+setAlignment(child: Node, pos: Pos)

The node placed in the top region (default: null).
The node placed in the right region (default: null).
The node placed in the bottom region (default: null).
The node placed in the left region (default: null).
The node placed in the center region (default: null).

Creates a BorderPane.

Sets the alignment of the node in the BorderPane.



Hbox , VBox

javafx.scene.layout.HBox

-alignment: ObjectProperty<Pos>
-fillHeight: BooleanProperty
-spacing: DoubleProperty

+HBox()
+HBox(spacing: double)
+setMargin(node: Node, value: Insets): void

The overall alignment of the children in the box (default: Pos.TOP_LEFT).
Is resizable children fill the full height of the box (default: true).
The horizontal gap between two nodes (default: 0).

Creates a default HBox.
Creates an HBox with the specified horizontal gap between nodes.
Sets the margin for the node in the pane.

javafx.scene.layout.VBox

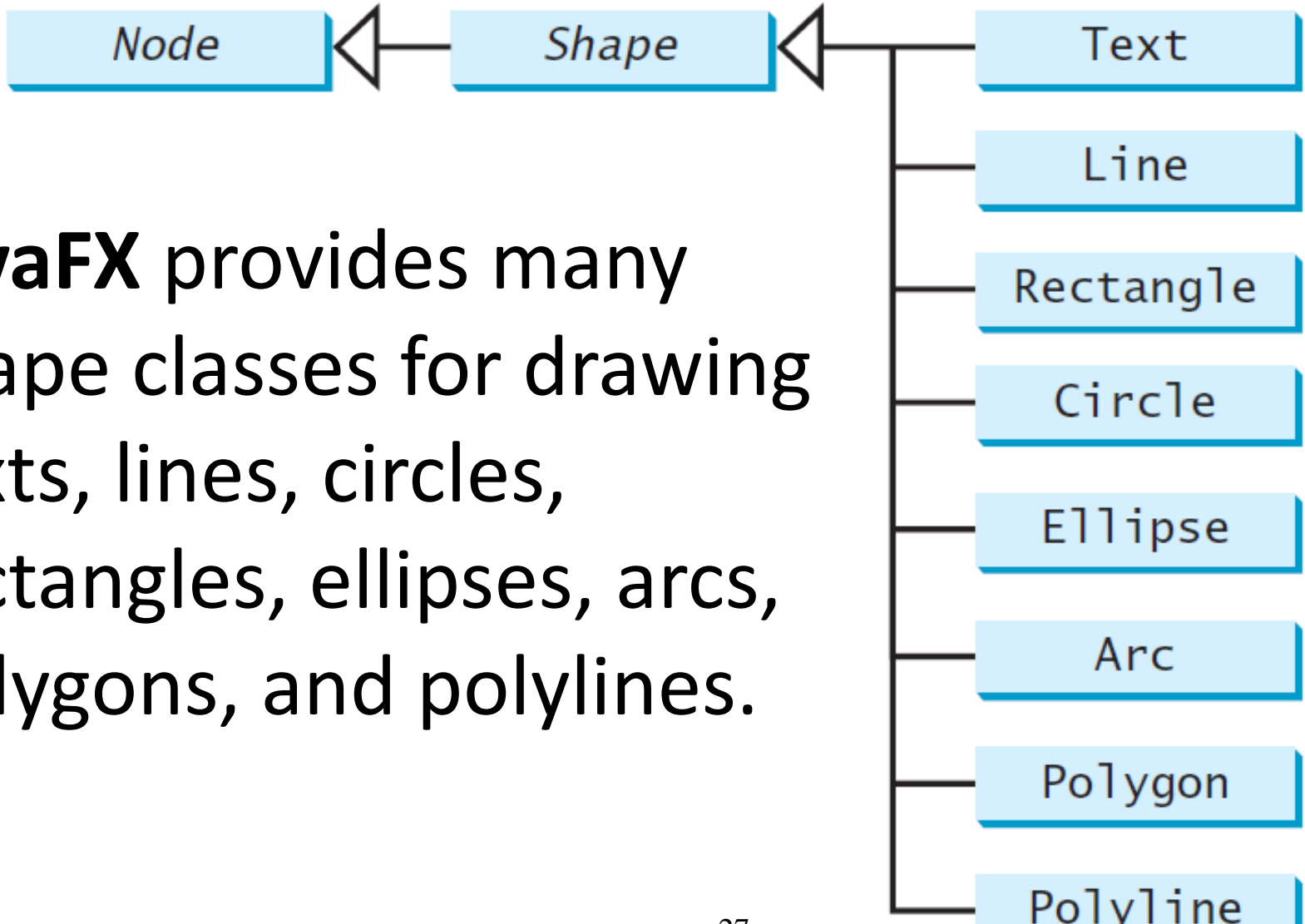
-alignment: ObjectProperty<Pos>
-fillWidth: BooleanProperty
-spacing: DoubleProperty

+VBox()
+VBox(spacing: double)
+setMargin(node: Node, value: Insets): void

The overall alignment of the children in the box (default: Pos.TOP_LEFT).
Is resizable children fill the full width of the box (default: true).
The vertical gap between two nodes (default: 0).

Creates a default VBox.
Creates a VBox with the specified horizontal gap between nodes.
Sets the margin for the node in the pane.

Shapes



- ❖ **JavaFX** provides many shape classes for drawing texts, lines, circles, rectangles, ellipses, arcs, polygons, and polylines.

Text

`javafx.scene.text.Text`

-text: StringProperty
-x: DoubleProperty
-y: DoubleProperty
-underline: BooleanProperty
-strikethrough: BooleanProperty
-font: ObjectProperty

+Text()
+Text(text: String)
+Text(x: double, y: double,
text: String)

Defines the text to be displayed.

Defines the x-coordinate of text (default 0).

Defines the y-coordinate of text (default 0).

Defines if each line has an underline below it (default false).

Defines if each line has a line through it (default false).

Defines the font for the text.

Creates an empty Text.

Creates a Text with the specified text.

Creates a Text with the specified x-, y-coordinates and text.

Line

javafx.scene.shape.Line

-startX: DoubleProperty
-startY: DoubleProperty
-endX: DoubleProperty
-endY: DoubleProperty

+Line()

+Line(startX: double, startY: double, endX: double, endY: double)

The x-coordinate of the start point.
The y-coordinate of the start point.
The x-coordinate of the end point.
The y-coordinate of the end point.

Creates an empty Line.

Creates a Line with the specified starting and ending points.

(0, 0)

(getWidth(), 0)

(startX, startY)

(endX, endY)

(0, getHeight())

(getWidth(), getHeight())

By: Jibreel Bornat



Rectangle

`javafx.scene.shape.Rectangle`

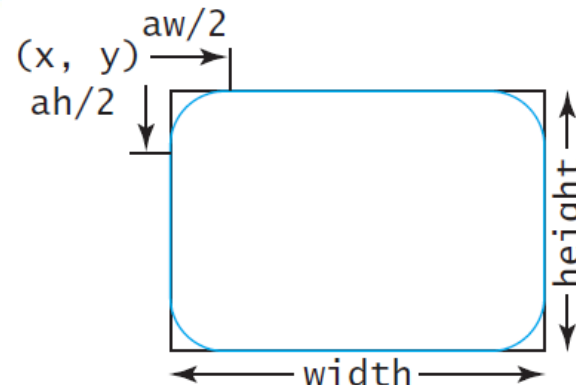
-x: DoubleProperty
-y: DoubleProperty
-width: DoubleProperty
-height: DoubleProperty
-arcWidth: DoubleProperty
-arcHeight: DoubleProperty

+Rectangle()
+Rectangle(x: double, y: double, width: double, height: double)

The x-coordinate of the upper-left corner of the rectangle (default 0).
The y-coordinate of the upper-left corner of the rectangle (default 0).
The width of the rectangle (default: 0).
The height of the rectangle (default: 0).
The arcWidth of the rectangle (default: 0). arcWidth is the horizontal diameter of the arcs at the corner (see Figure 14.31a).
The arcHeight of the rectangle (default: 0). arcHeight is the vertical diameter of the arcs at the corner (see Figure 14.31a).

Creates an empty Rectangle.

Creates a Rectangle with the specified upper-left corner point, width, and height.



(a) `Rectangle(x, y, w, h)`

Circle, Ellipse

`javafx.scene.shape.Circle`

-centerX: DoubleProperty
-centerY: DoubleProperty
-radius: DoubleProperty

+Circle()
+Circle(x: double, y: double)
+Circle(x: double, y: double,
radius: double)

The x-coordinate of the center of the circle (default 0).
The y-coordinate of the center of the circle (default 0).
The radius of the circle (default: 0).

Creates an empty `Circle`.
Creates a `Circle` with the specified center.
Creates a `Circle` with the specified center and radius.

`javafx.scene.shape.Ellipse`

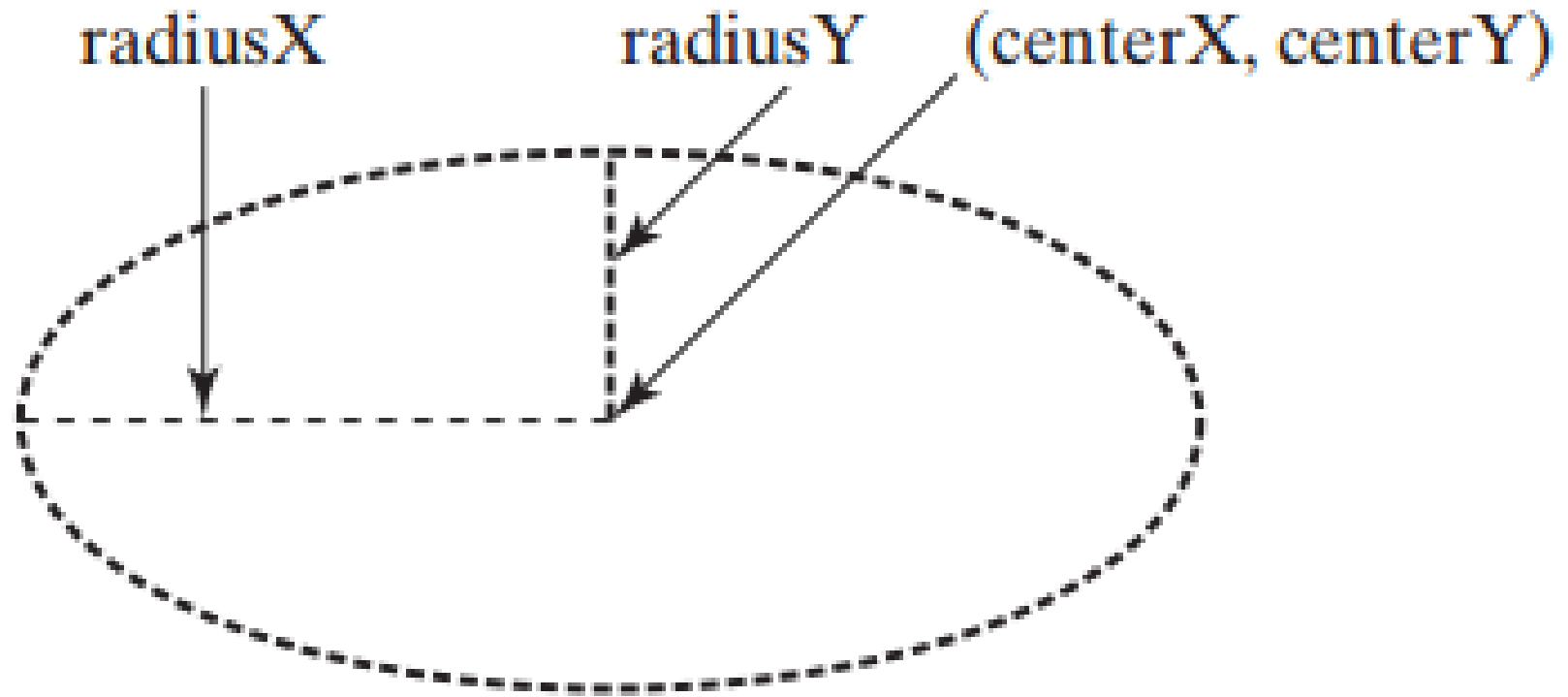
-centerX: DoubleProperty
-centerY: DoubleProperty
-radiusX: DoubleProperty
-radiusY: DoubleProperty

+Ellipse()
+Ellipse(x: double, y: double)
+Ellipse(x: double, y: double,
radiusX: double, radiusY:
double)

The x-coordinate of the center of the ellipse (default 0).
The y-coordinate of the center of the ellipse (default 0).
The horizontal radius of the ellipse (default: 0).
The vertical radius of the ellipse (default: 0).

Creates an empty `Ellipse`.
Creates an `Ellipse` with the specified center.
Creates an `Ellipse` with the specified center and radiuses.

Ellipse



(a) `Ellipse(centerX, centerY,
radiusX, radiusY)`

Arc

`javafx.scene.shape.Arc`

-centerX: DoubleProperty
-centerY: DoubleProperty
-radiusX: DoubleProperty
-radiusY: DoubleProperty
-startAngle: DoubleProperty
-length: DoubleProperty
-type: ObjectProperty<ArcType>

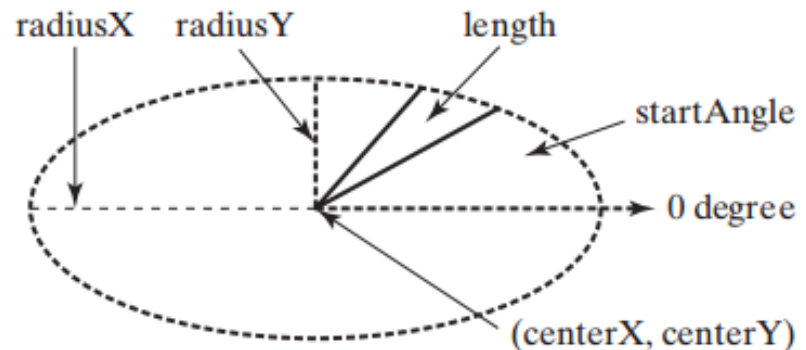
+Arc()

+Arc(x: double, y: double,
radiusX: double, radiusY:
double, startAngle: double,
length: double)

The x-coordinate of the center of the ellipse (default 0).
The y-coordinate of the center of the ellipse (default 0).
The horizontal radius of the ellipse (default: 0).
The vertical radius of the ellipse (default: 0).
The start angle of the arc in degrees.
The angular extent of the arc in degrees.
The closure type of the arc (`ArcType.OPEN`,
`ArcType.CHORD`, `ArcType.ROUND`).

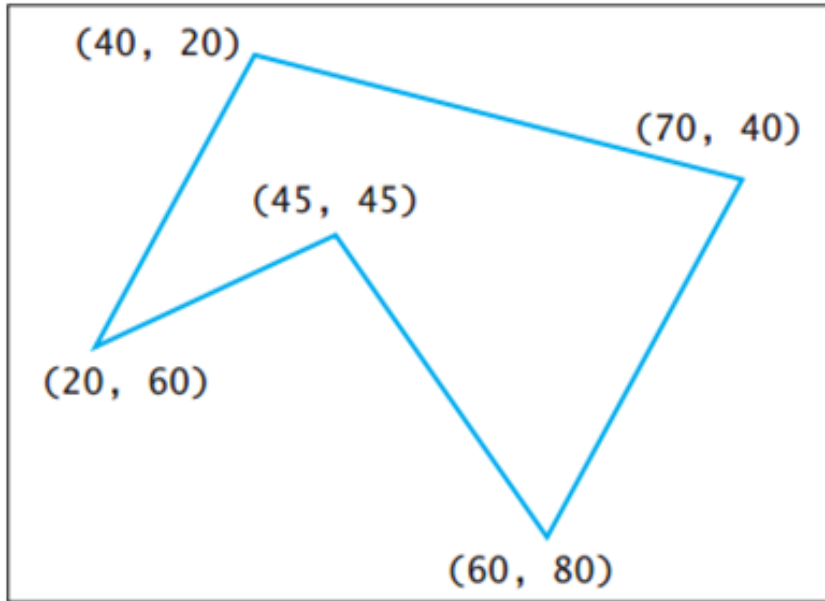
Creates an empty Arc.

Creates an Arc with the specified arguments.

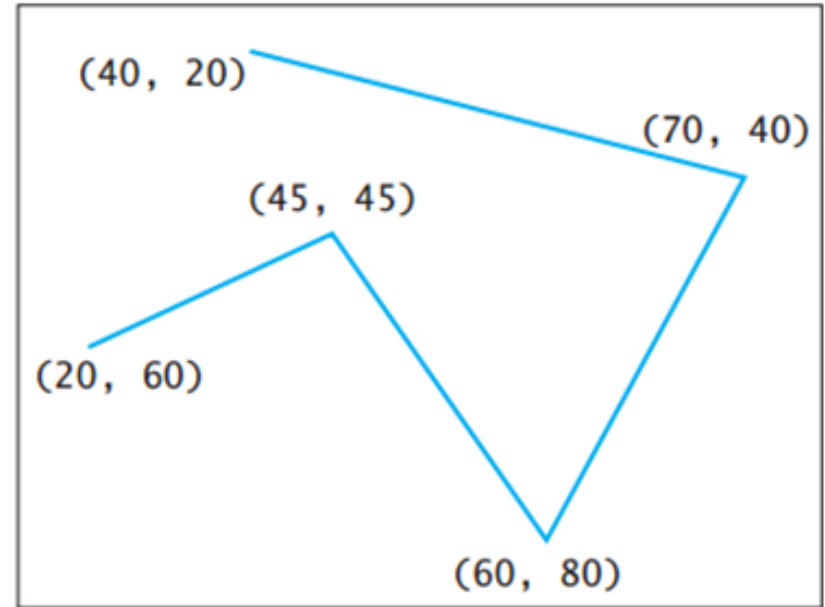


(a) `Arc(centerX, centerY, radiusX,
radiusY, startAngle, length)`

Polygon and Polyline



(a) Polygon



(b) Polyline

`javafx.scene.shape.Polygon`

```
+Polygon()  
+Polygon(double... points)  
+getPoints():  
    ObservableList<Double>
```

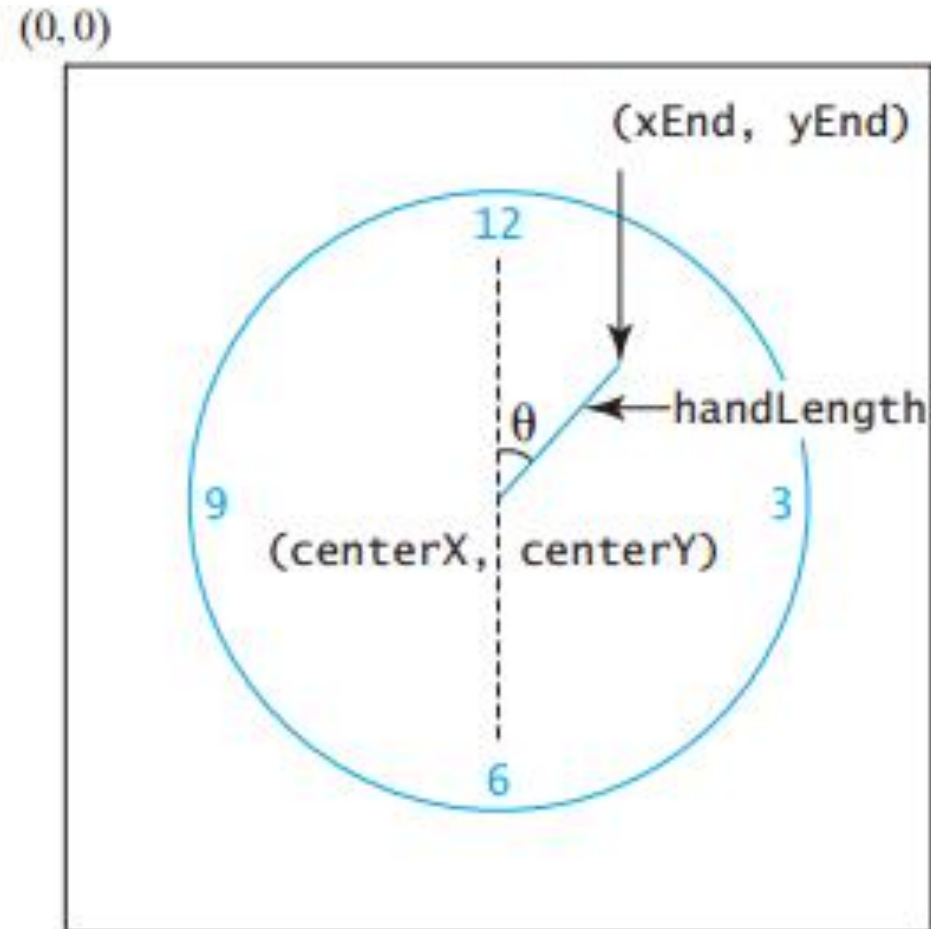
Creates an empty polygon.

Creates a polygon with the given points.

Returns a list of double values as x- and y-coordinates of the points.

Case Study: The **ClockPane** Class

- ❖ This case study develops a class that displays a clock on a pane.



Case Study: The **ClockPane** Class

`javafx.scene.layout.Pane`



ClockPane

-hour: int
-minute: int
-second: int
-w: double
-h: double

+ClockPane()
+ClockPane(hour: int, minute:
int, second: int)
+setCurrentTime(): void

The hour in the clock.

The minute in the clock.

The second in the clock.

The width of the pane that contains the clock.

The height of the pane that contains the clock.

Constructs a default clock for the current time.

Constructs a clock with the specified time.

Sets hour, minute, and second to current time.