BIRZEIT UNIVERSITY

COMPUTER SCIENCE DEPARTMENT FACULTY OF ENGINEERING AND TECHNOLOGY

ADVANCED PROGRAMMING COMP231

Instructor :Murad Njoum
Office : Masri322

Chapter 14 JavaFX Basics

Motivations

JavaFX is a **new framework** for developing Java **GUI programs**.

The **JavaFX API** is an excellent example of how the object-oriented principle is applied.

This chapter serves **two purposes**.

First, it presents the basics of JavaFX programming.

Second, it uses JavaFX to demonstrate OOP.

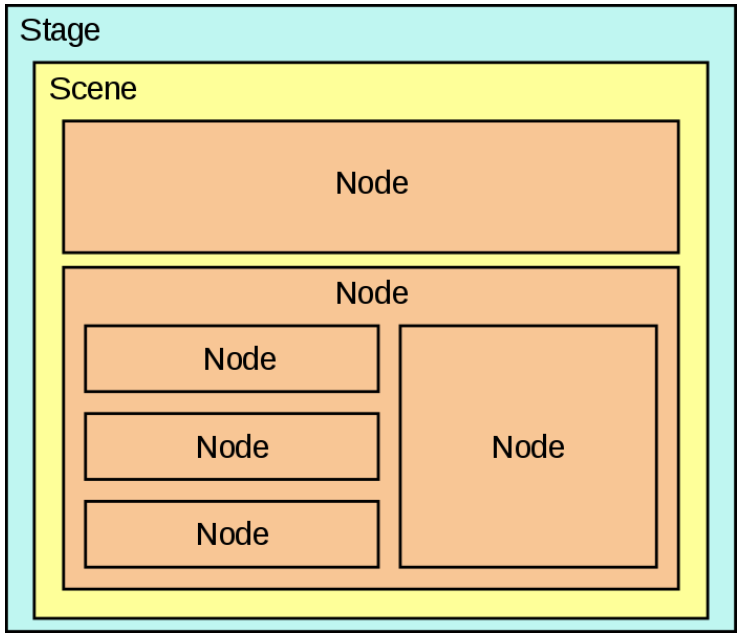
Specifically, this chapter introduces the framework of JavaFX and discusses **JavaFX GUI components** and their relationships.

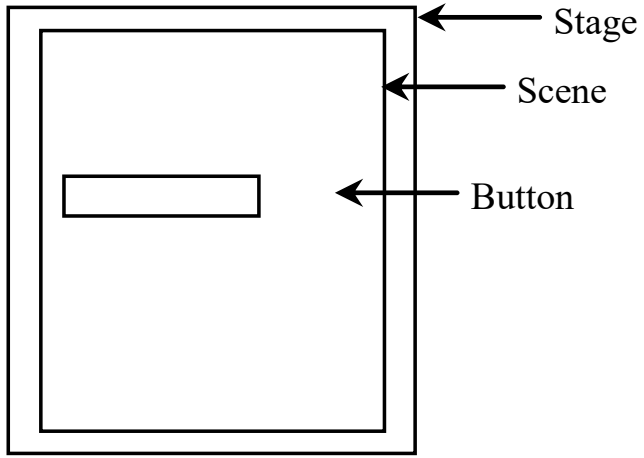
JavaFX vs Swing and AWT

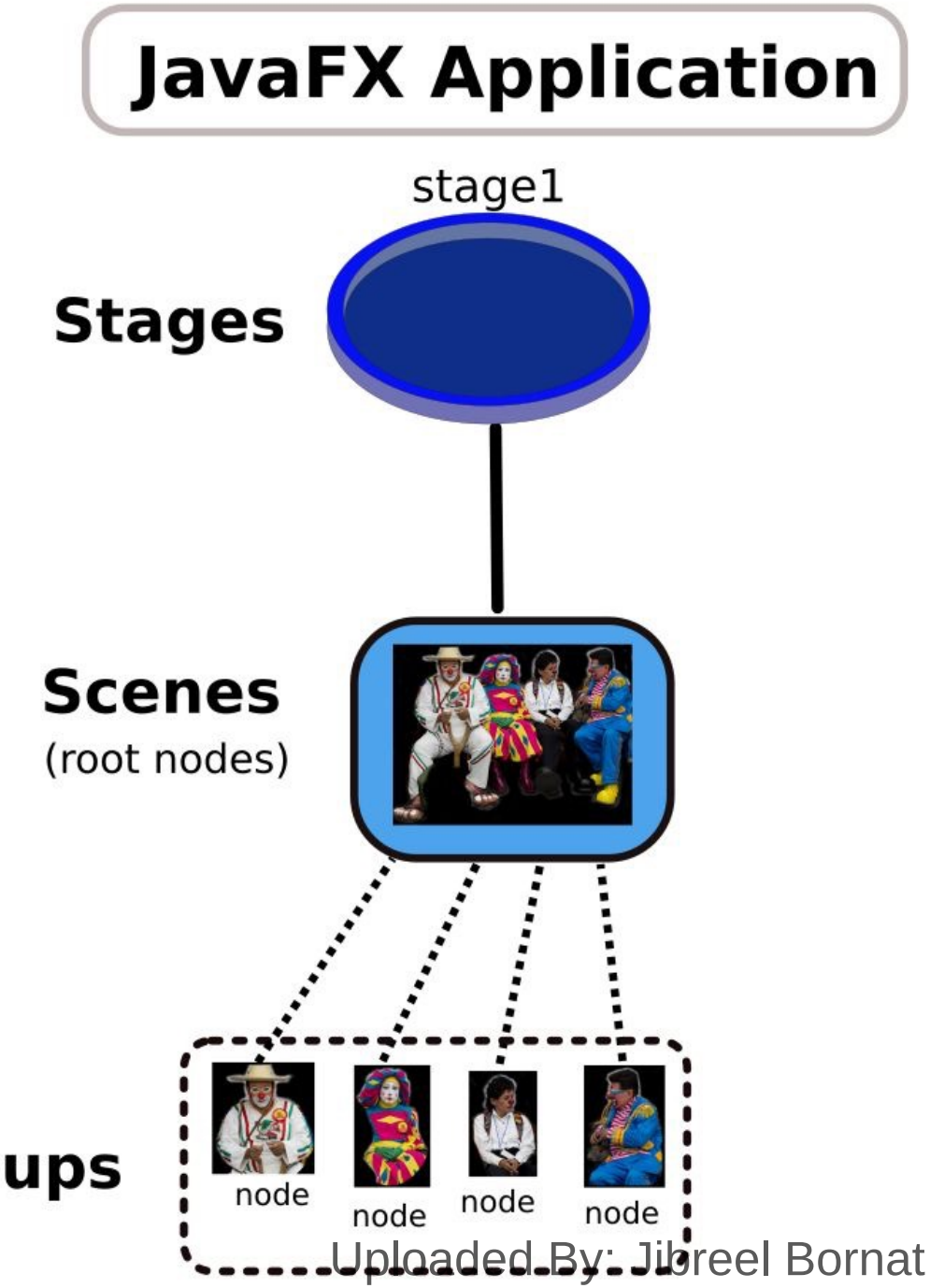
- When Java was introduced, the GUI classes were bundled in a library known as the **Abstract Windows Toolkit (AWT).**
- AWT is fine for developing simple graphical user interfaces, but not for **developing comprehensive GUI projects**.
- In addition, AWT is prone to platform-specific bugs. The AWT user-interface components were replaced by a more robust, versatile, and flexible library known as **Swing components.**
- Swing components are painted directly on **canvases using Java code.** 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.**
- Swing and AWT are replaced by the JavaFX platform for developing **rich Internet applications**.

Basic Structure of JavaFX

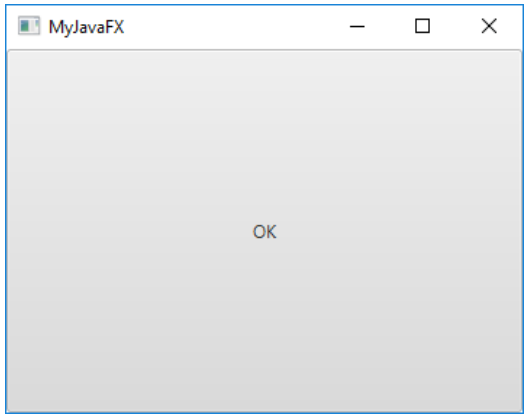
- Application
- Override the start(Stage)
- Stage, Scene, and Nodes



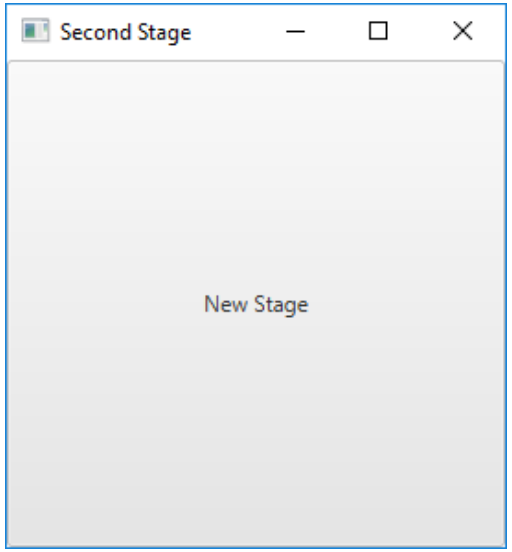




```java
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.stage.Stage;

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

 //add second stage add code here
  }

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



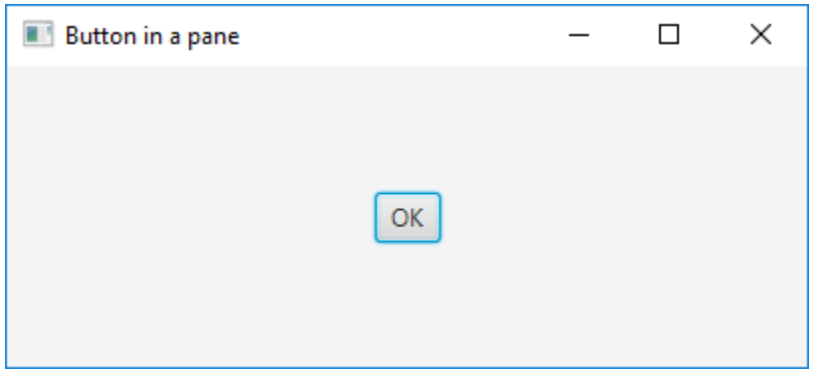
1
2
3
4
5
6
7



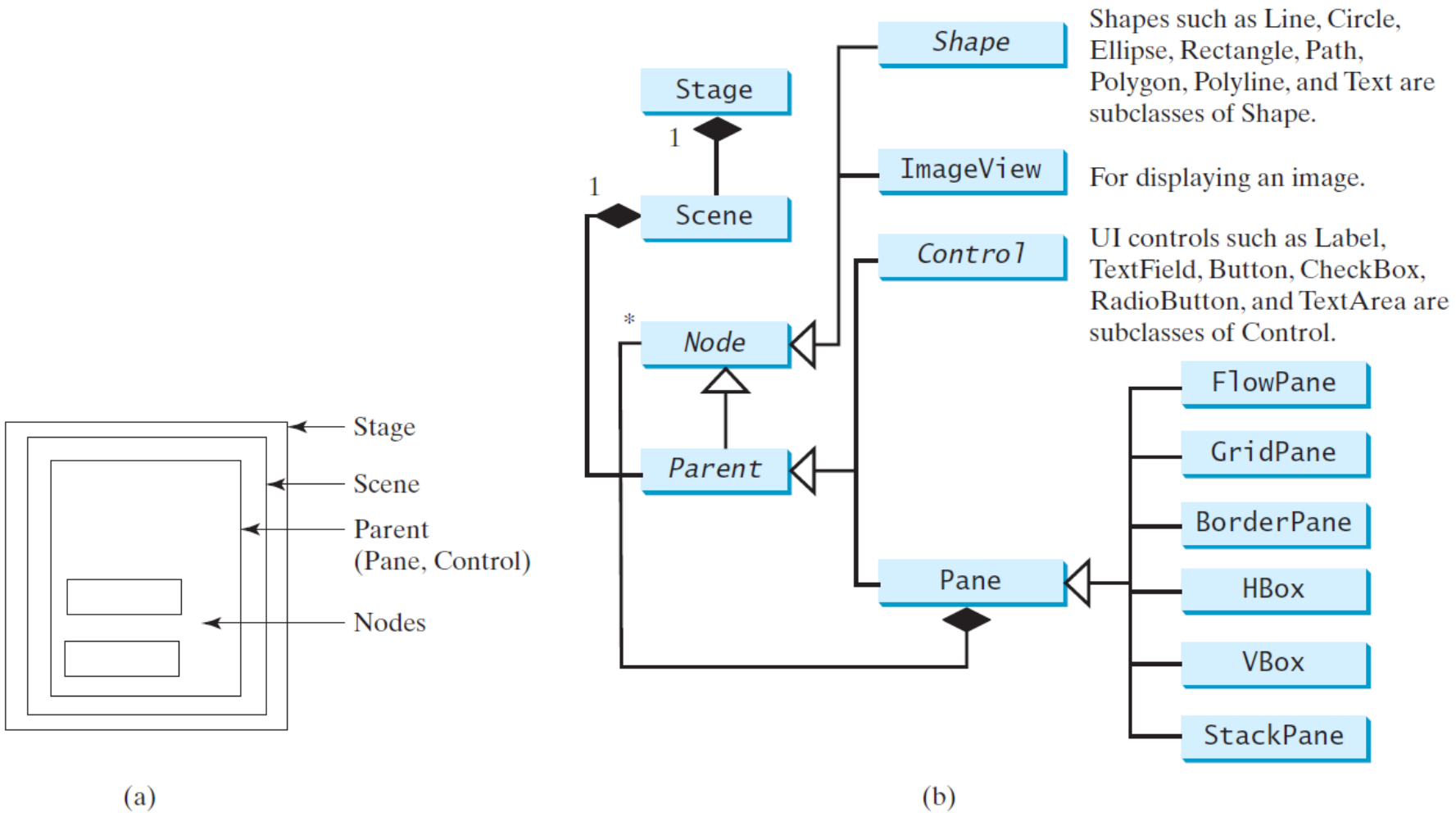

```
Stage stage = new Stage(); // Create a new stage
stage.setTitle("Second Stage"); // Set the stage title
// Set a scene with a button in the stage
stage.setScene(new Scene(new Button("New Stage"), 200, 250));
stage.show(); // Display the stage
```

StackPane is a container which can contain different interface components, subcomponents stacked up to others, and at a certain moment, you can only see the subcomponent lying on the top of Stack.

```
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 {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // Create a scene and place a button in the scene
    StackPane pane = new StackPane();
    pane.getChildren().add(new Button("OK"));
//getChildren method is used to get the children components(such as checkboxes, buttons) in a container
    Scene scene = new Scene(pane, 400, 150);
    primaryStage.setTitle("Button in a pane"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }

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



Panes, UI Controls, and Shapes



Each JavaFX Node (subclass) instance can only be added to the JavaFX scene. In other words, each Node instance **can only appear in one place in** the scene graph. If you try to add the same Node instance, or Node subclass instance, to the scene graph more than once, **JavaFX will throw an exception**!

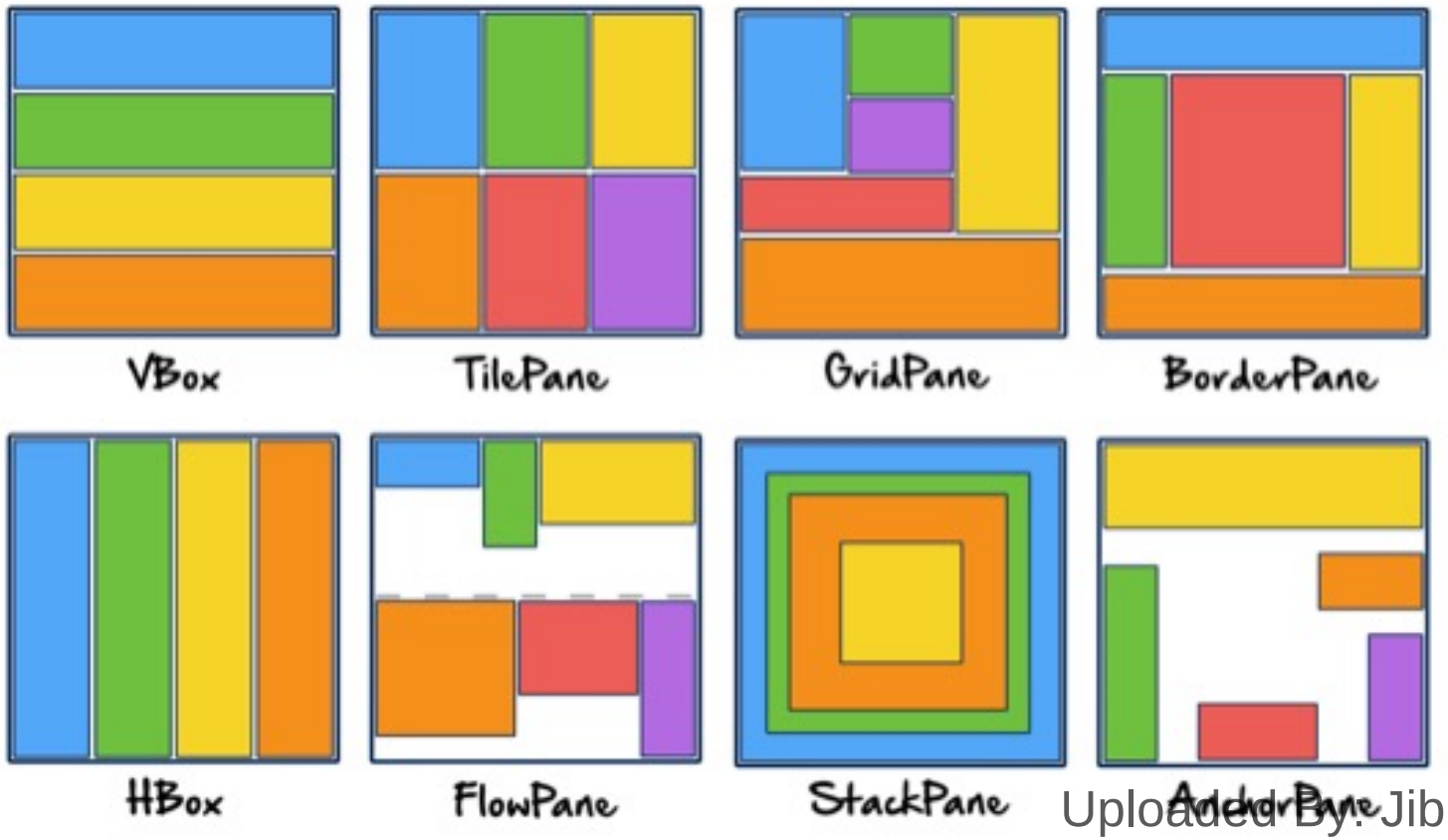
Layout Panes

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

Class	*Description*
`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.

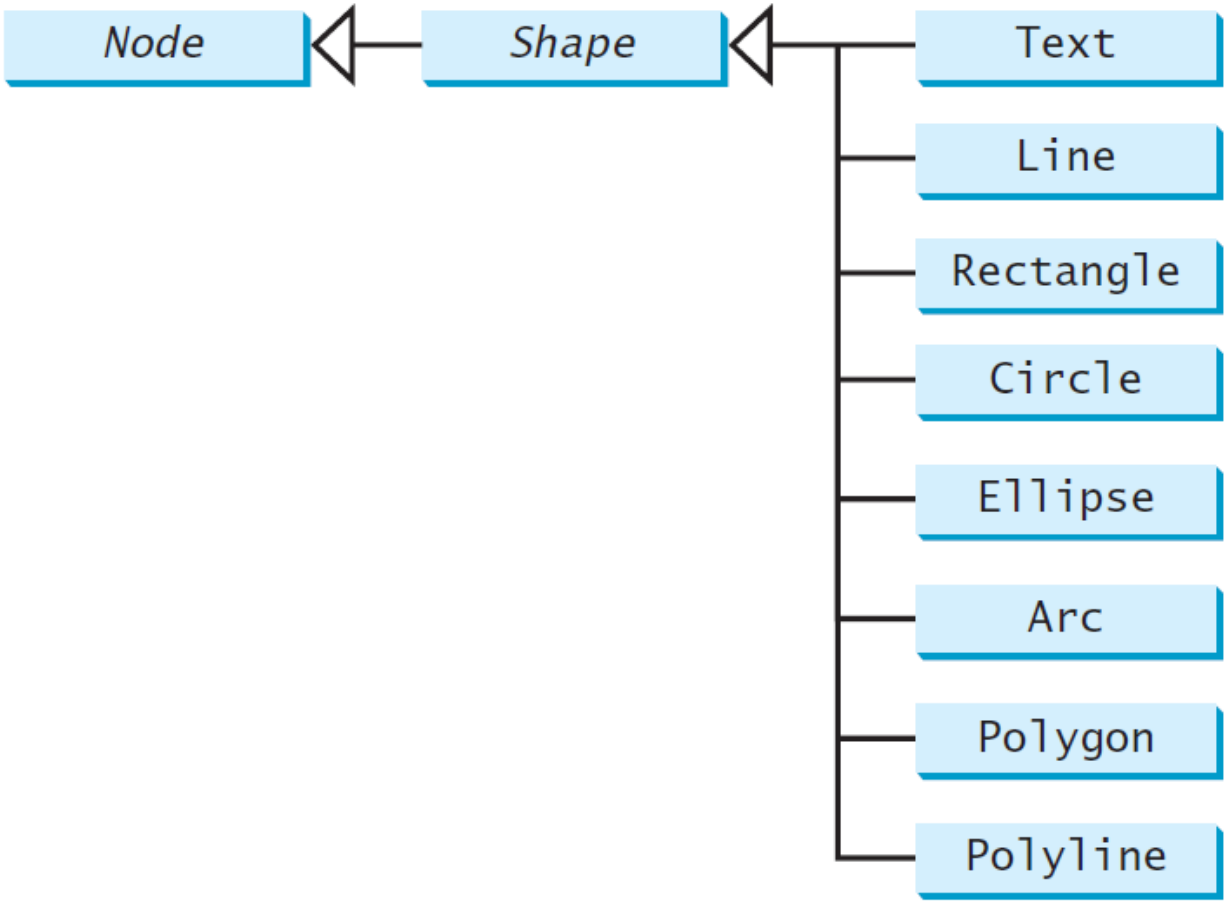
Example 1-1 Create a Border Pane

```
BorderPane border = new BorderPane();
HBox hbox = addHBox()
border.setTop(hbox);
border.setLeft(addVBox());
addStackPane(hbox);        // Add stack to HBox in top region

border.setCenter(addGridPane());
border.setRight(addFlowPane());
```



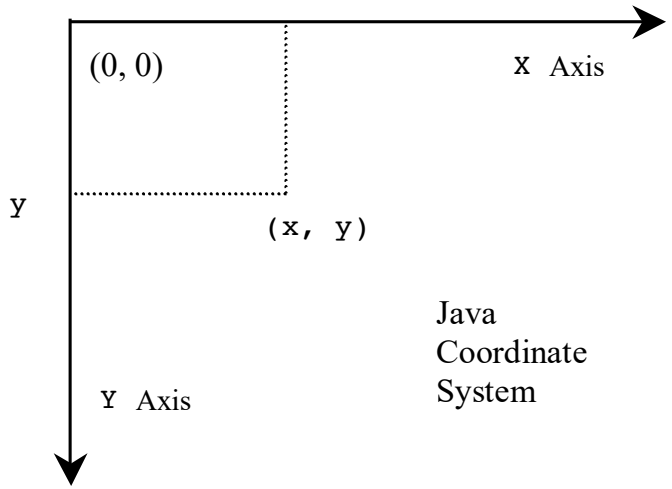
Shapes

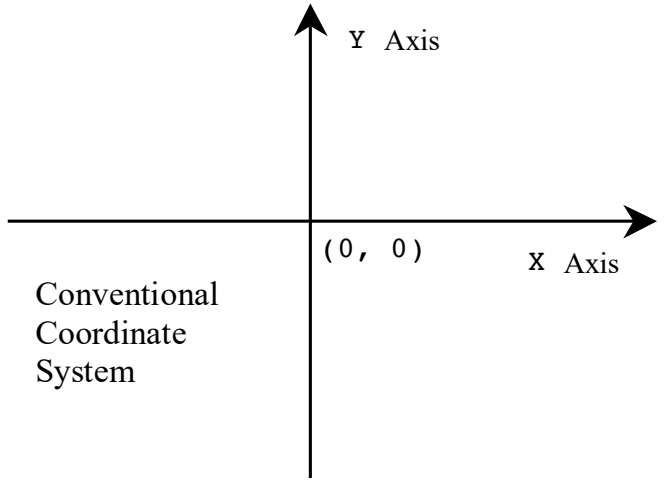
JavaFX provides many shape classes for drawing texts, lines, circles, rectangles, ellipses, arcs, polygons, and polylines.



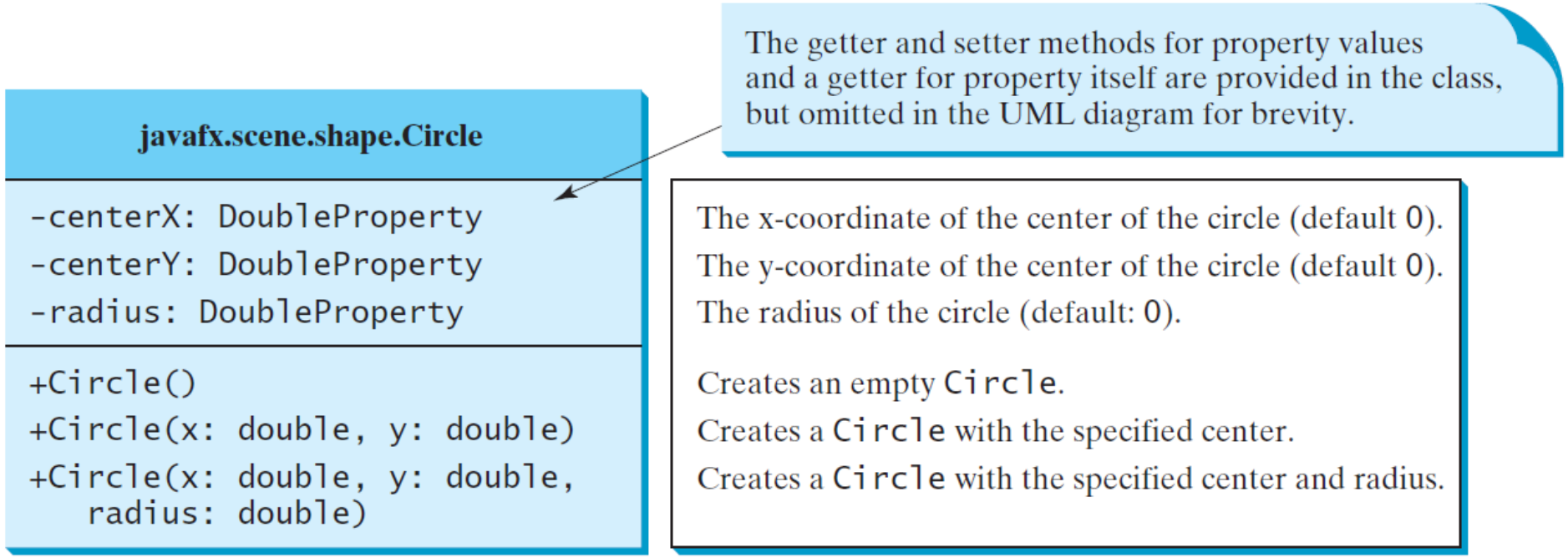
Display a Shape

This example displays a circle in the center of the pane.

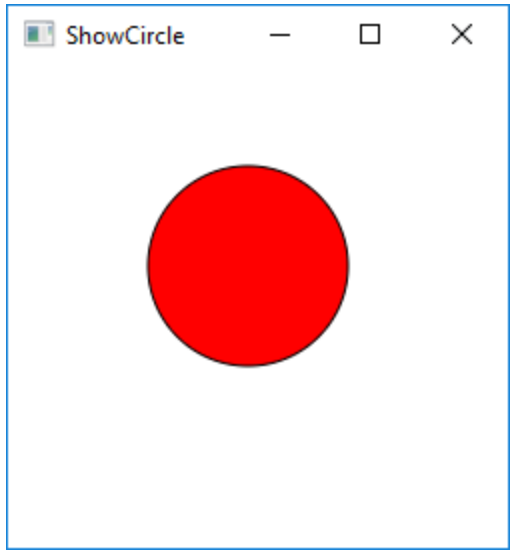




Circle


javafx.scene.shape.Circle
-centerX: DoubleProperty
-centerY: DoubleProperty
-radius: DoubleProperty
+Circle()
+Circle(x: double, y: double)
+Circle(x: double, y: double, radius: double)
The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.
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.

```java
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 {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // Create a circle and set its properties
    Circle circle = new Circle();
    circle.setCenterX(120);
    circle.setCenterY(100);
    circle.setRadius(50);
    circle.setStroke(Color.BLACK);
    circle.setFill(Color.RED);

    // Create a pane to hold the circle
    Pane pane = new Pane();
    pane.getChildren().add(circle);

    // Create a scene and place it in the stage
    Scene scene = new Scene(pane, 250, 250);
    primaryStage.setTitle("ShowCircle"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }

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



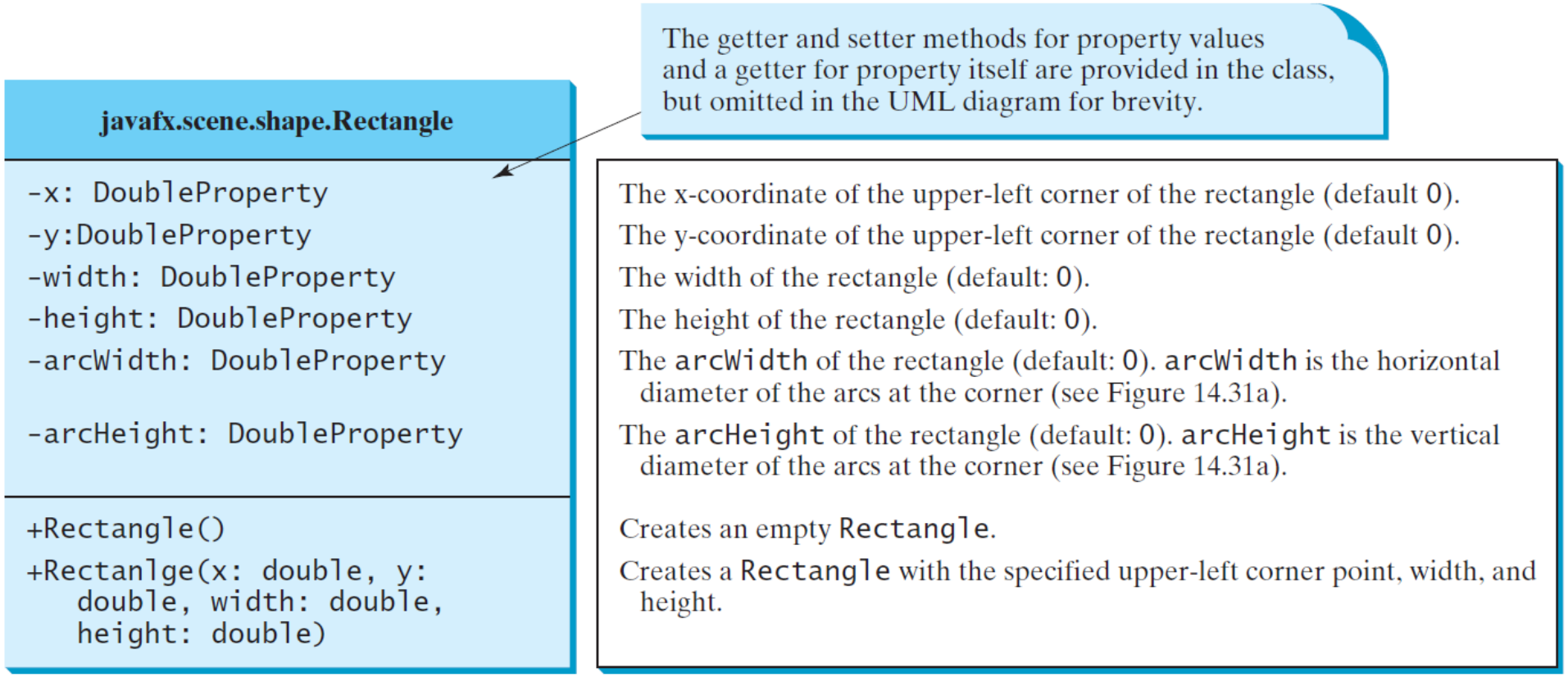
```
import javafx.beans.property.DoubleProperty; import
javafx.beans.property.SimpleDoubleProperty;

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

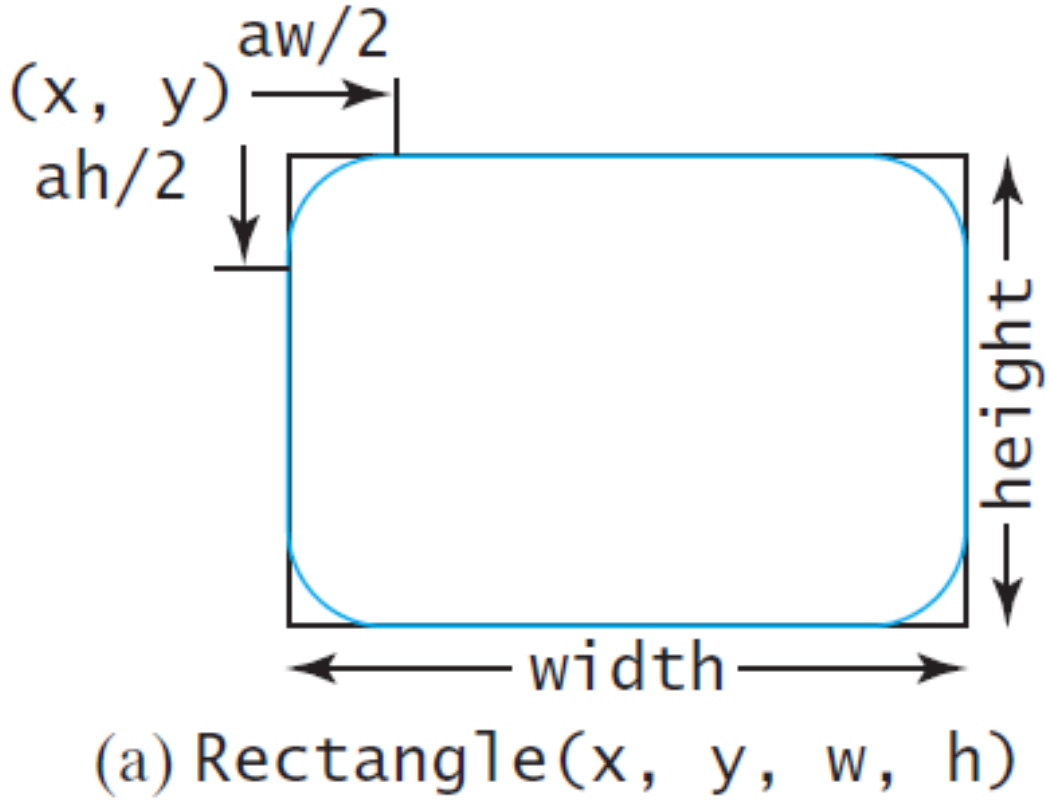
Binding Circle:

```
Circle c=new Circle(150,150,100,Color.GOLD);


c.centerXProperty().bind(root.widthProperty().divide(2));
c.centerYProperty().bind(root.heightProperty().divide(2));
```

Rectangle


javafx.scene.shape.Rectangle
-x: DoubleProperty
-y:DoubleProperty
-width: DoubleProperty
-height: DoubleProperty
-arcWidth: DoubleProperty
-arcHeight: DoubleProperty
+Rectangle()
+Rectanlge(x: double, y: double, width: double, height: double)
The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.
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.

Rectangle Example

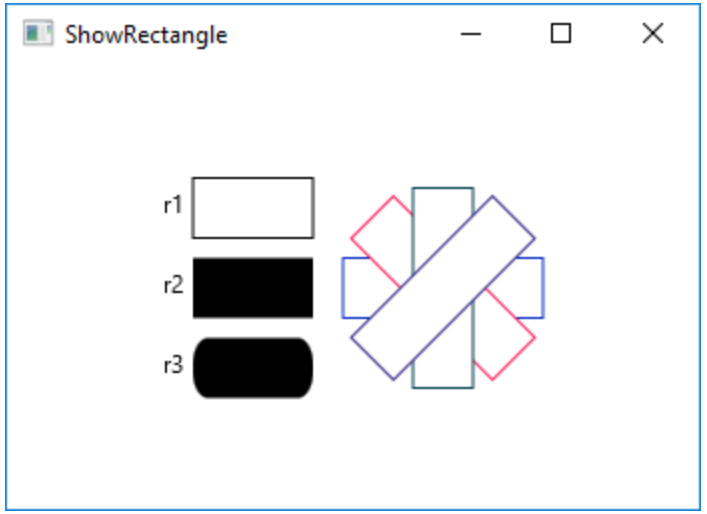


(a) Rectangle(x, y, w, h)

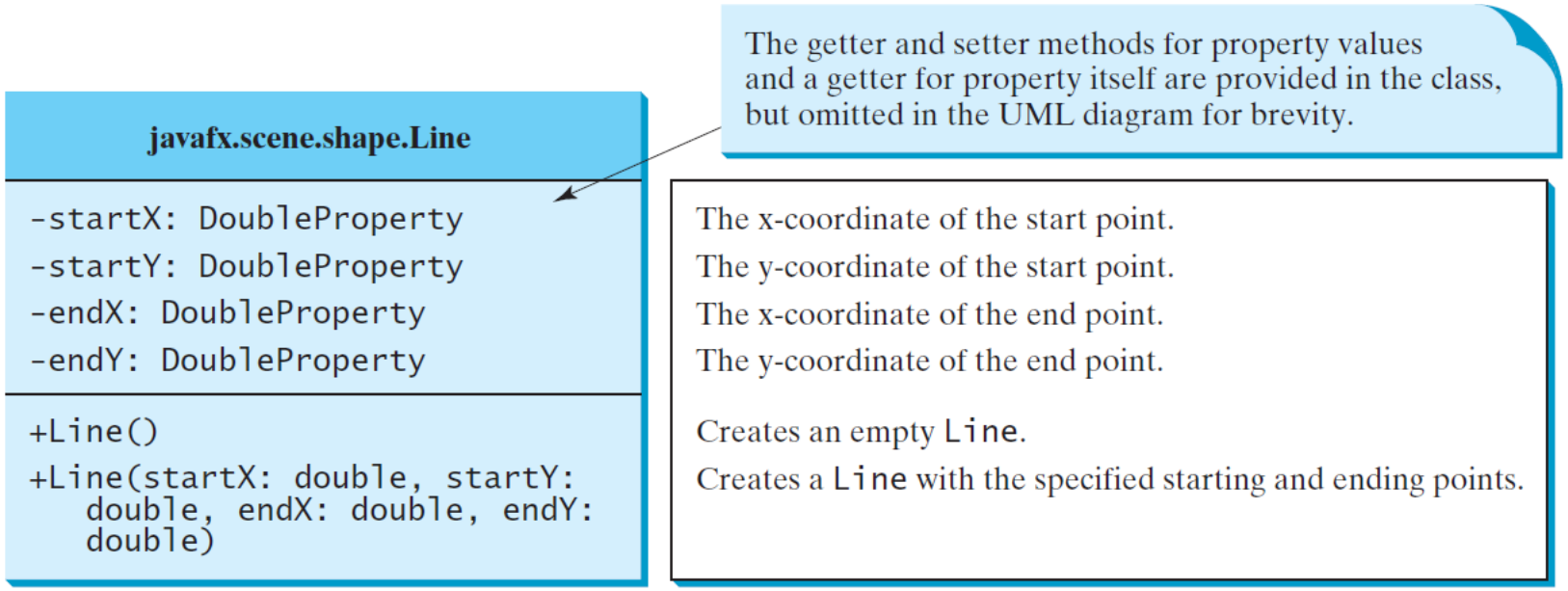
ShowRectangle Run

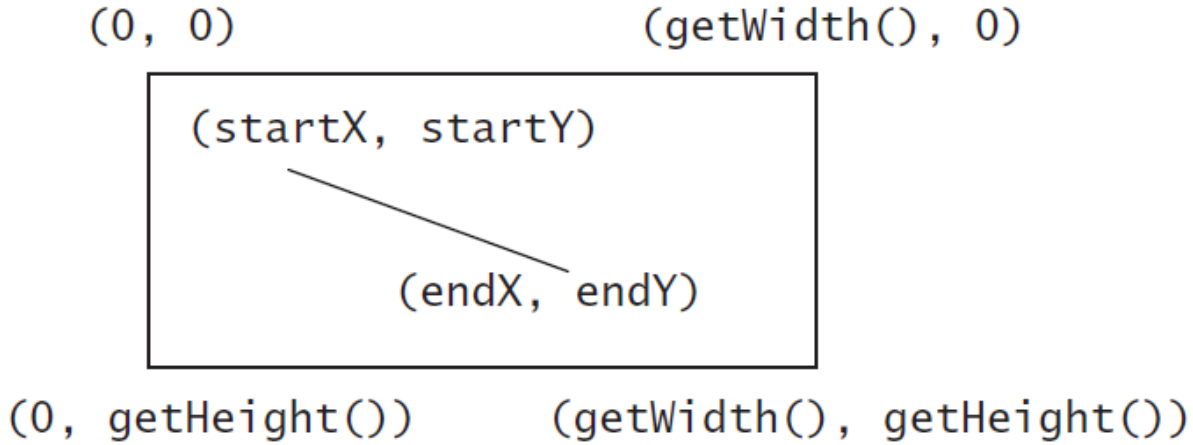
```
public class ShowRectangle extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // Create rectangles
    Rectangle r1 = new Rectangle(25, 10, 60, 30);
    r1.setStroke(Color.BLACK);
    r1.setFill(Color.WHITE);
    Rectangle r2 = new Rectangle(25, 50, 60, 30);
    Rectangle r3 = new Rectangle(25, 90, 60, 30);
    r3.setArcWidth(15);
    r3.setArcHeight(25);

    // Create a group and add nodes to the group
    Group group = new Group();
    group.getChildren().addAll(new Text(10, 27, "r1"), r1,
      new Text(10, 67, "r2"), r2, new Text(10, 107, "r3"), r3);
```

```
    for (int i = 0; i < 4; i++) {
      Rectangle r = new Rectangle(100, 50, 100, 30);
      r.setRotate(i * 360 / 8);
      r.setStroke(Color.color(Math.random(), Math.random(),
        Math.random()));
      r.setFill(Color.WHITE);
      group.getChildren().add(r);
    }

    // Create a scene and place it in the stage
    Scene scene = new Scene(new BorderPane(group), 250, 150);
    primaryStage.setTitle("ShowRectangle"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }

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



Line





ShowLine Run

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

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

```
class LinePane extends Pane {
  public LinePane() {
    Line line1 = new Line(10, 10, 30, 50);
     line1.setStrokeWidth(5);
    line1.setStroke(Color.GREEN);
    getChildren().add(line1);

    Line line2 = new Line(20, 20, 150, 150);
    line2.setStrokeWidth(5);
    line2.setStroke(Color.GREEN);
    getChildren().add(line2);
  }
}
```

Ellipse

The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

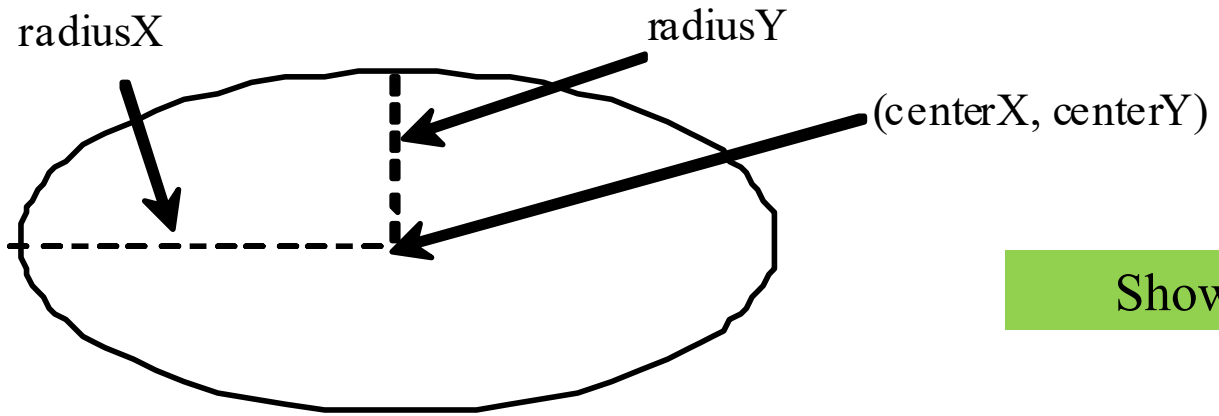
javafx.scene.shape.Ellipse

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

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

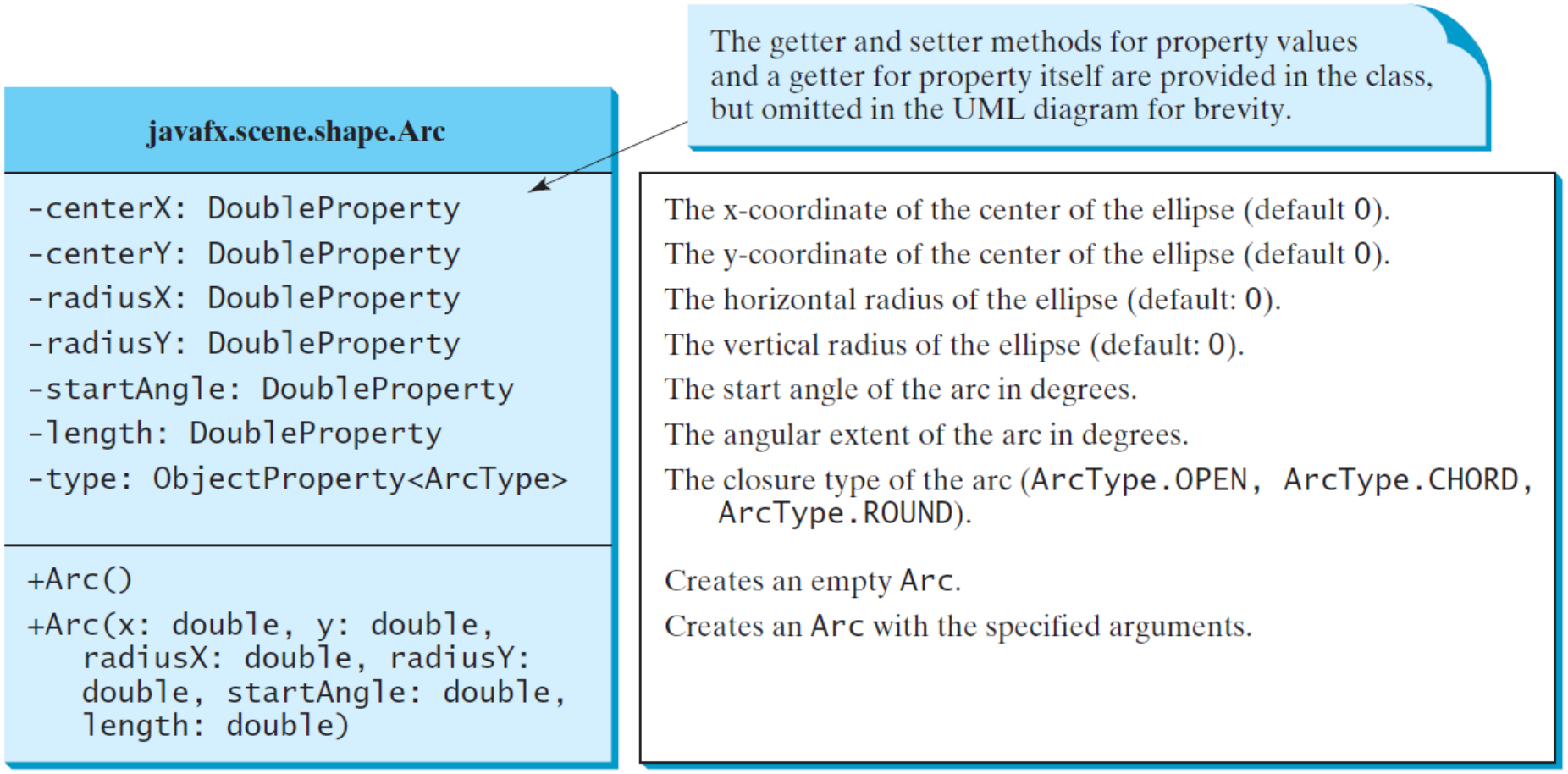
The x-coordinate of the center of the ellipse (default 0).
The y-coordinate of the center of the ellipse (default 0).
The horizontal radius of the ellipse (default: 0).
The vertical radius of the ellipse (default: 0).

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

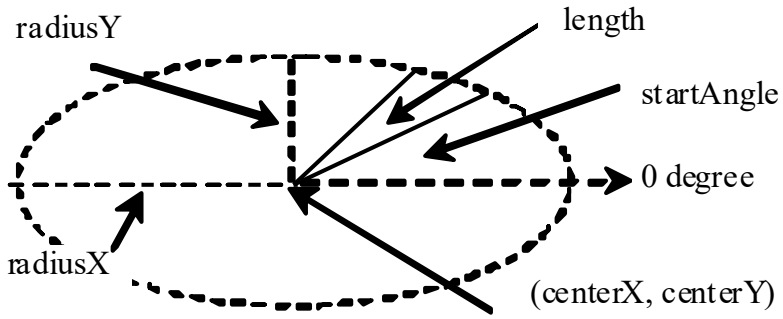


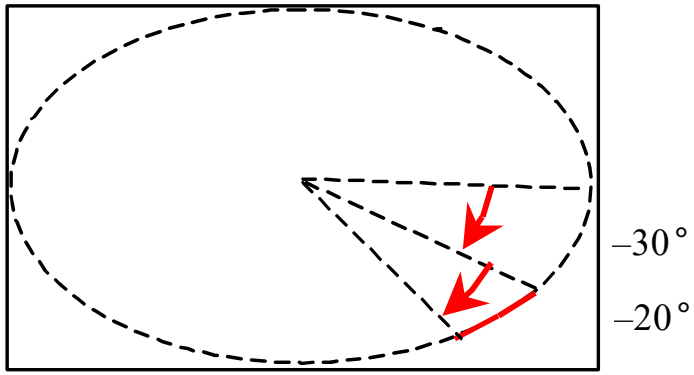
ShowEllipse Run

Arc

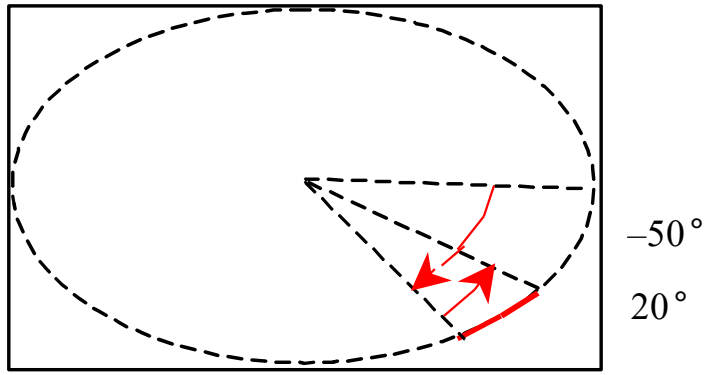


Arc Examples



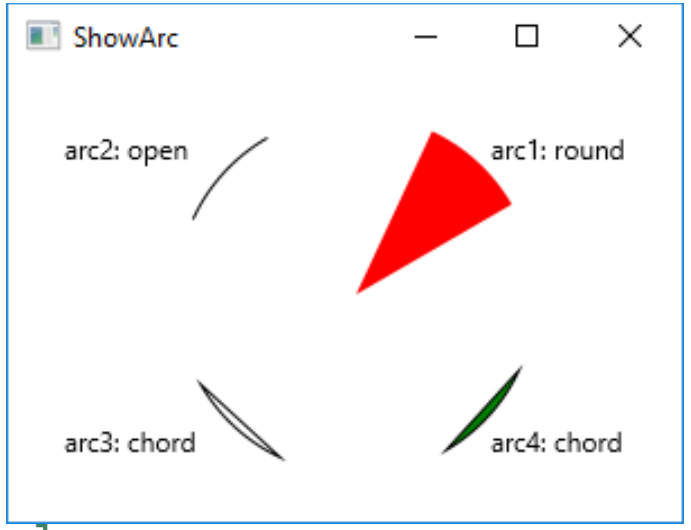


(a) Negative starting angle –30° and negative spanning angle –20°



(b) Negative starting angle –50° and positive spanning angle 20°

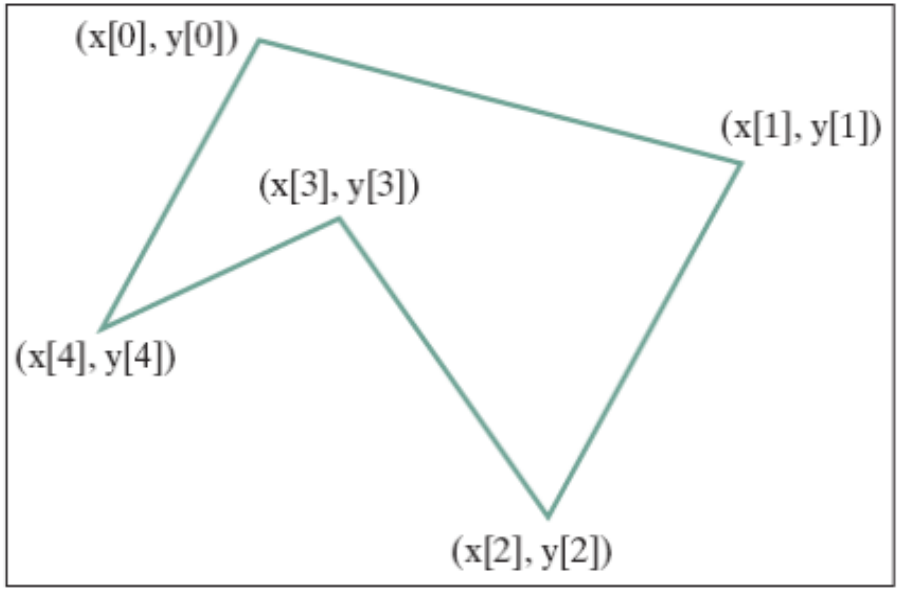
ShowArc Run



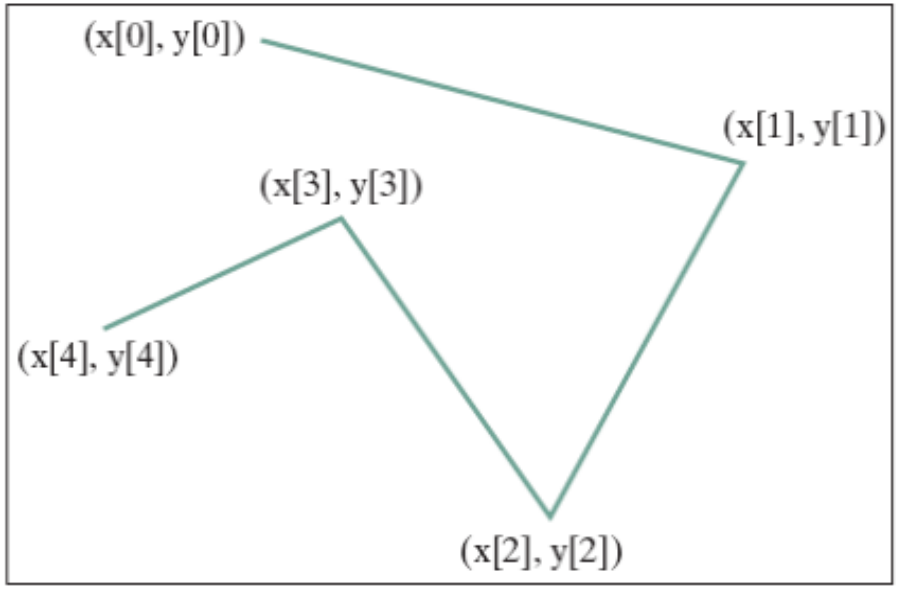

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.Group;
import javafx.scene.layout.BorderPane;
import javafx.scene.paint.Color;
import javafx.stage.Stage;
import javafx.scene.shape.Arc;
import javafx.scene.shape.ArcType;
import javafx.scene.text.Text;

public class ShowArc extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    Arc arc1 = new Arc(150, 100, 80, 80, 30, 35);
   // Create an arc (x,y,r.x,r.y, start degree,length degree)
    arc1.setFill(Color.RED); // Set fill color
    arc1.setType(ArcType.ROUND); // Set arc type

    Arc arc2 = new Arc(150, 100, 80, 80, 30 + 90, 35);
    arc2.setFill(Color.WHITE);
    arc2.setType(ArcType.OPEN);
    arc2.setStroke(Color.BLACK);

    Arc arc3 = new Arc(150, 100, 80, 80, 30 + 180, 35);
    arc3.setFill(Color.WHITE);
    arc3.setType(ArcType.CHORD);
    arc3.setStroke(Color.BLACK);
```

```
    Arc arc4 = new Arc(150, 100, 80, 80, 30 + 270, 35);
        arc4.setFill(Color.GREEN);
        arc4.setType(ArcType.CHORD);
        arc4.setStroke(Color.BLACK);

        // Create a group and add nodes to the group
        Group group = new Group();
        group.getChildren().addAll(new Text(210, 40, "arc1: round"),
          arc1, new Text(20, 40, "arc2: open"), arc2,
          new Text(20, 170, "arc3: chord"), arc3,
          new Text(210, 170, "arc4: chord"), arc4);

        // Create a scene and place it in the stage
        Scene scene = new Scene(new BorderPane(group), 300, 200);
        primaryStage.setTitle("ShowArc"); // Set the stage title
        primaryStage.setScene(scene); // Place the scene in the stage
        primaryStage.show(); // Display the stage
      }

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

Polygon and Polyline

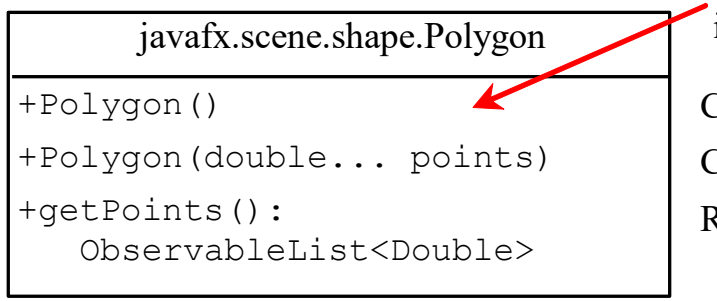


(a) Polygon



(b) Polyline

Polygon



The `getter` and `setter` methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

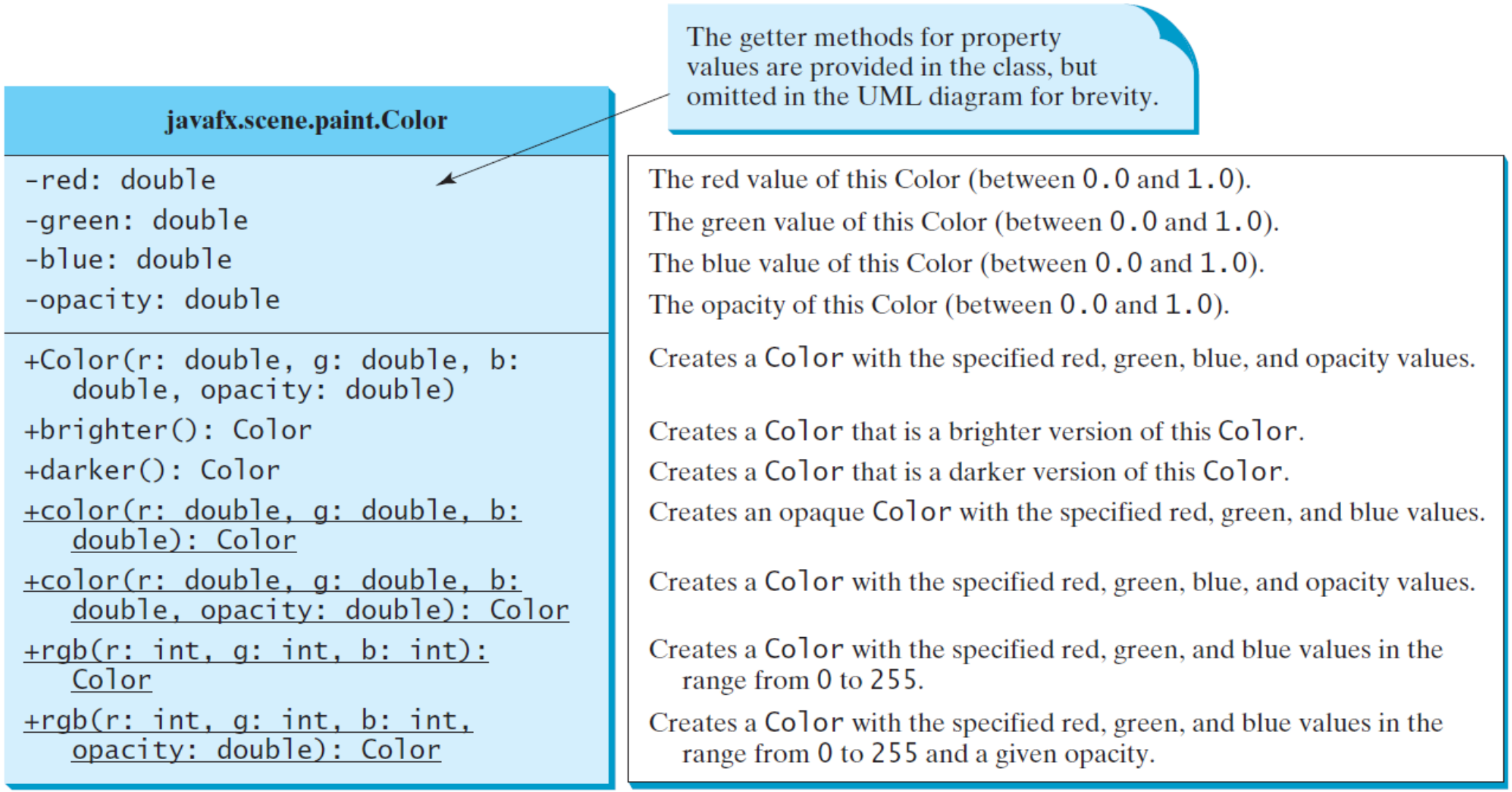
Creates an empty polygon.

Creates a polygon with the given points.

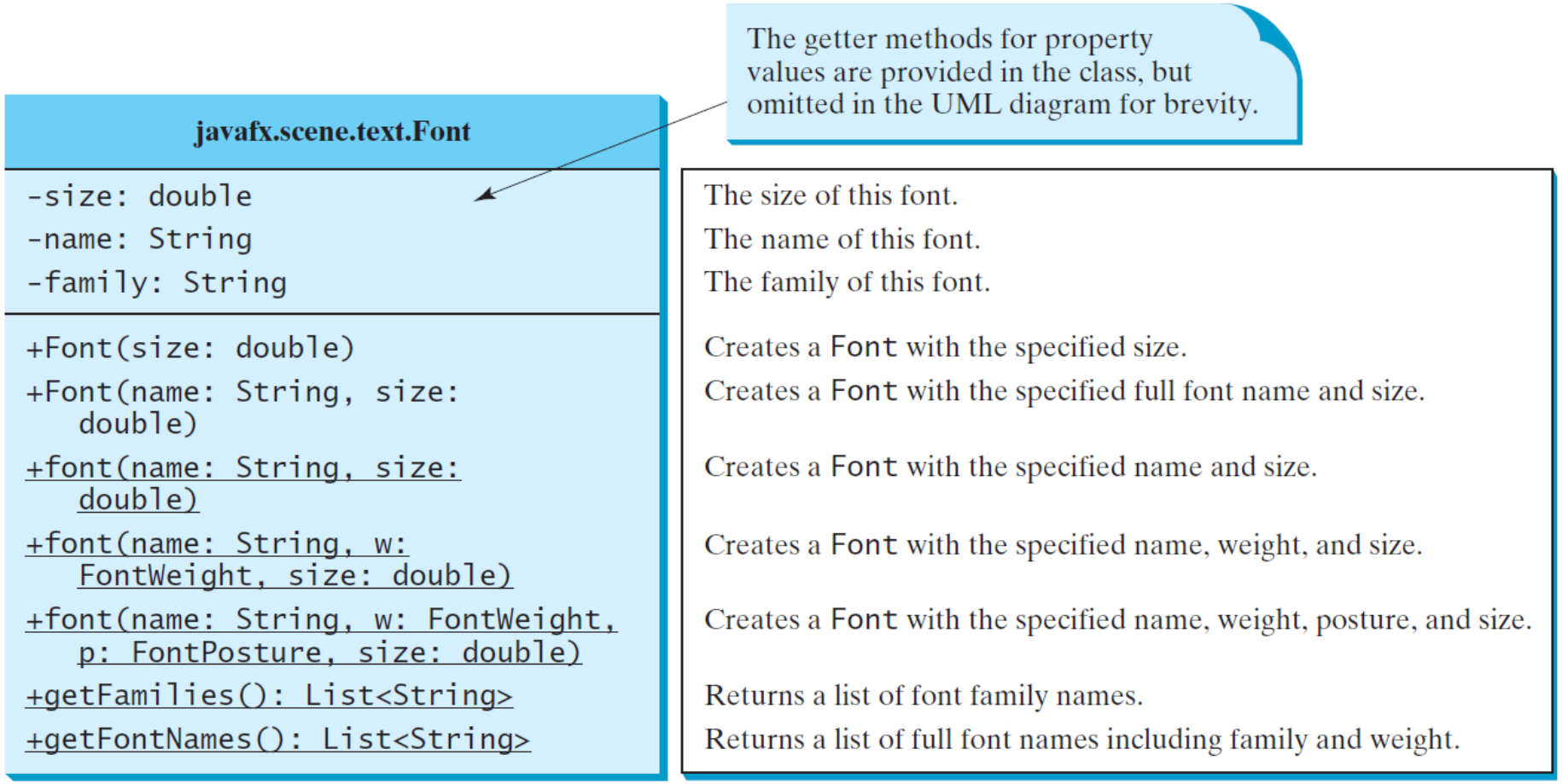
Returns a list of double values as x- and y-coordinates of the points.

ShowPolygon Run

The Color Class



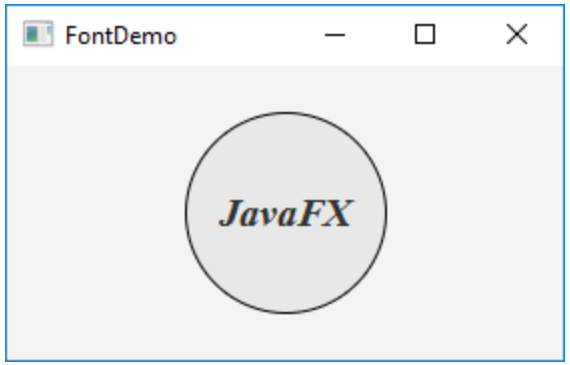
The Font Class



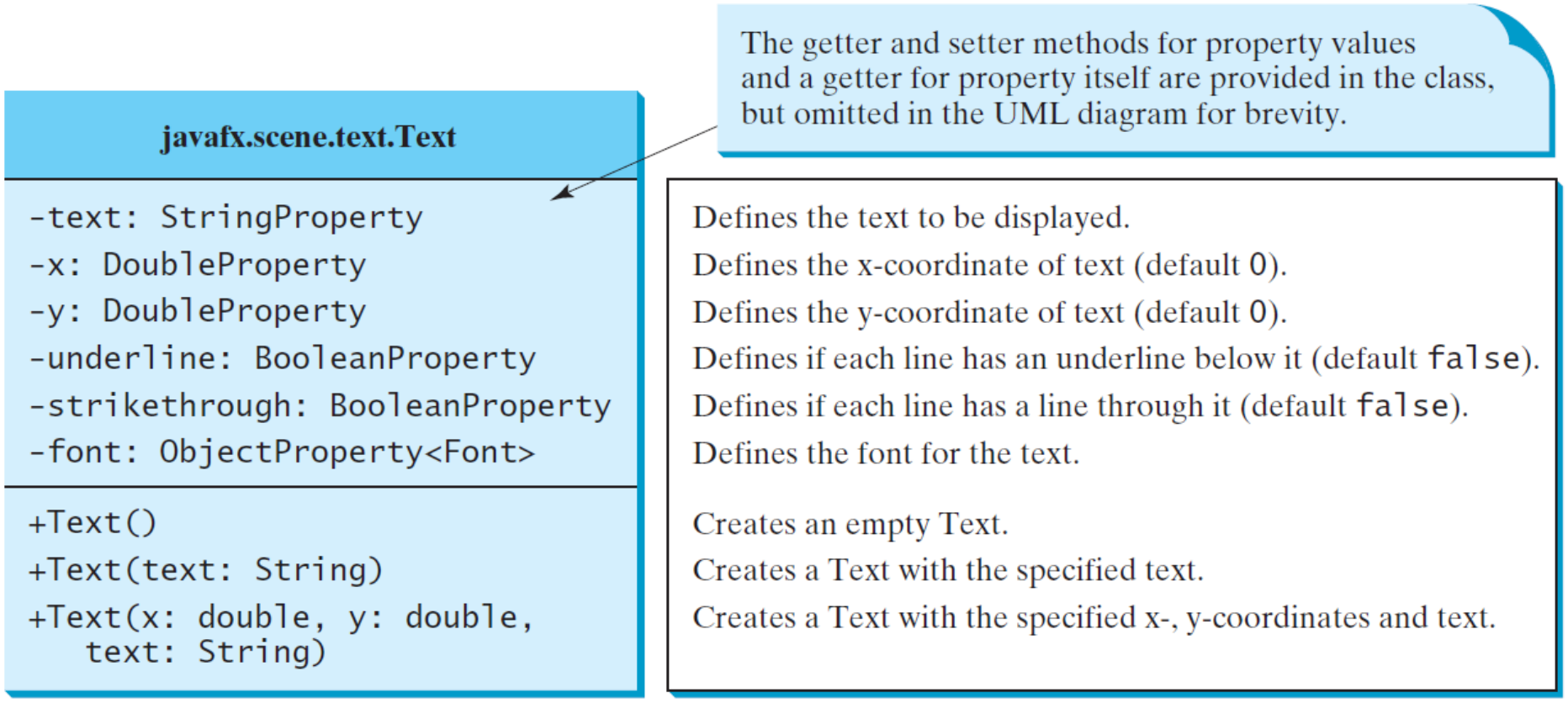
FontDemo Run

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

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
```

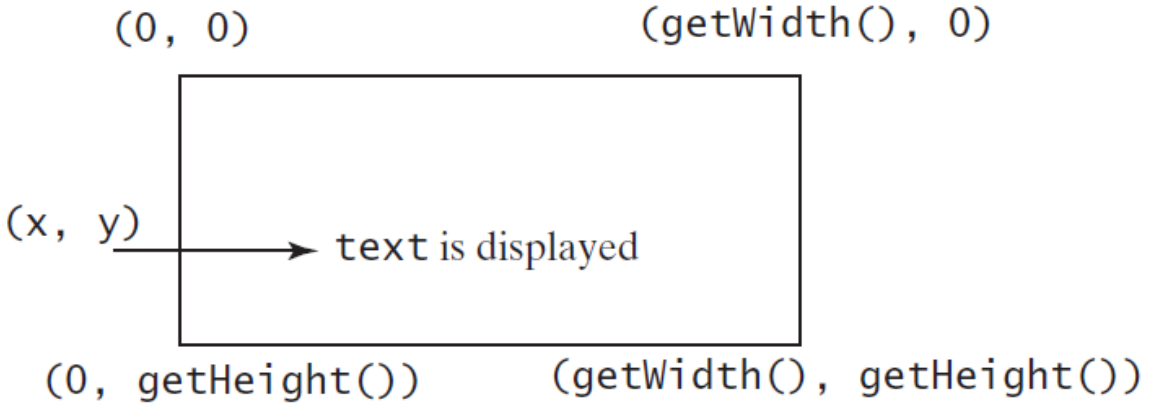
```
// Create a label and set its properties
    Label label = new Label("JavaFX");
    label.setFont(Font.font("Times New Roman",
      FontWeight.BOLD, FontPosture.ITALIC, 20));
    pane.getChildren().add(label);

    // 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
  }

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



Text



Text Example



(a) Text(x, y, text)

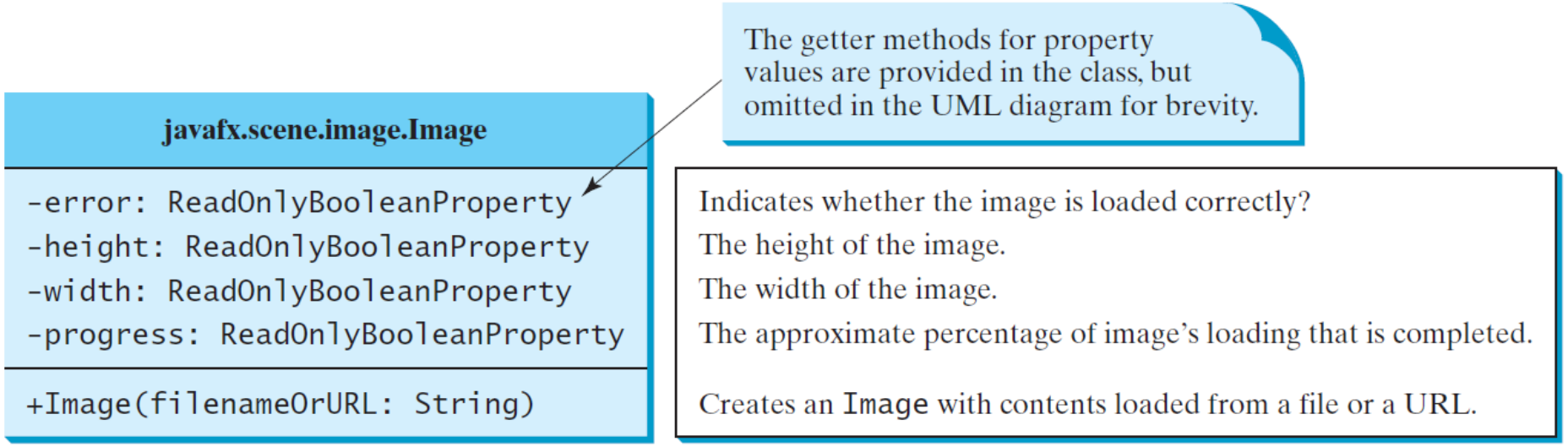
(b) *Three* Text *objects are displayed*

ShowText Run

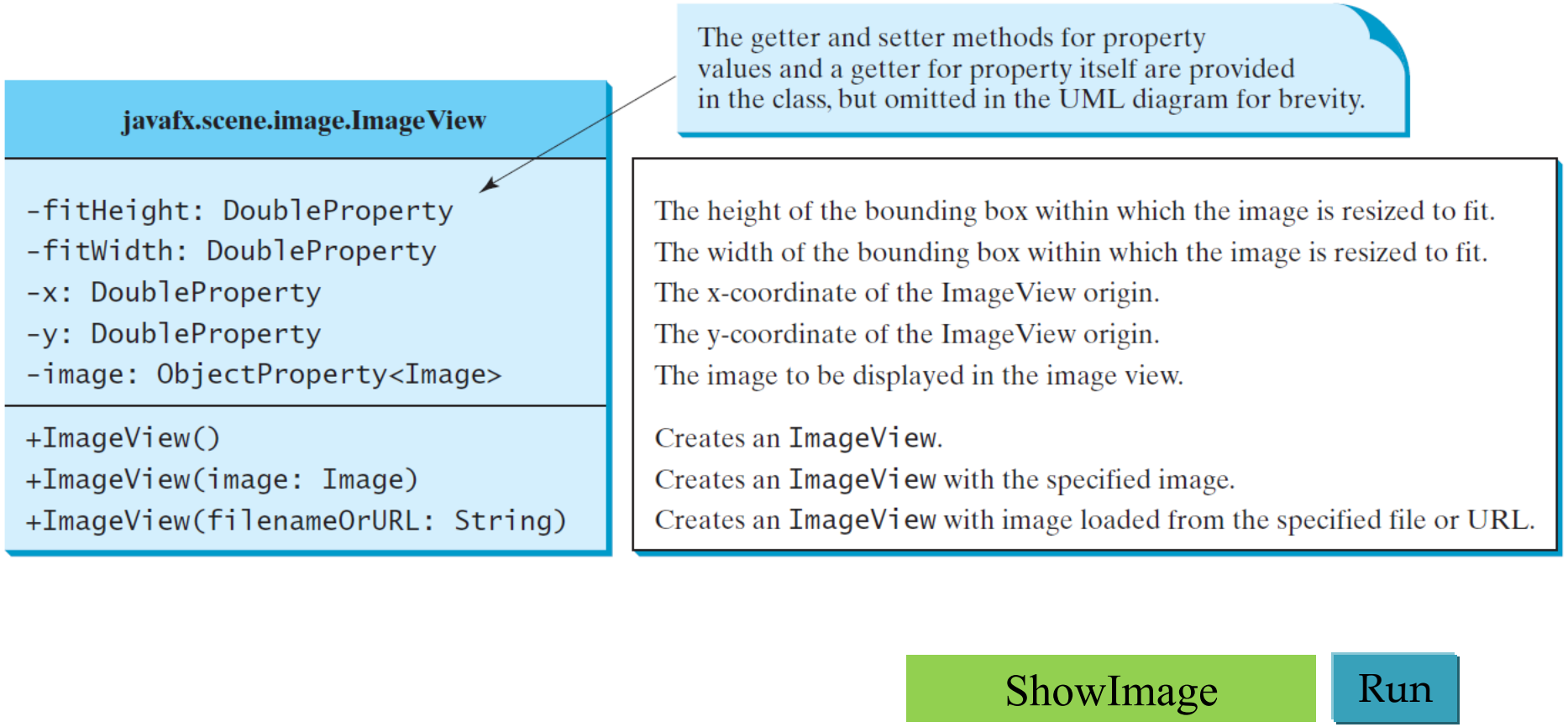
```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.geometry.Insets;
import javafx.stage.Stage;
import javafx.scene.text.Text;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.scene.text.FontPosture;

public class ShowText extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // Create a pane to hold the texts
    Pane pane = new Pane();
    pane.setPadding(new Insets(5, 5, 5, 5));
    Text text1 = new Text(20, 20, "Programming is fun");
    text1.setFont(Font.font("Courier", FontWeight.BOLD,
      FontPosture.ITALIC, 15));
    pane.getChildren().add(text1);
```

```
 Text text2 = new Text(60, 60, "Programming is fun\nDisplay text");
    pane.getChildren().add(text2);

    Text text3 = new Text(10, 100, "Programming is fun\nDisplay text");
    text3.setFill(Color.RED);
    text3.setUnderline(true);
    text3.setStrikethrough(true);
    pane.getChildren().add(text3);

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

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

The Image Class



The ImageView Class



ShowImage Run

BorderPane

The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.layout.BorderPane

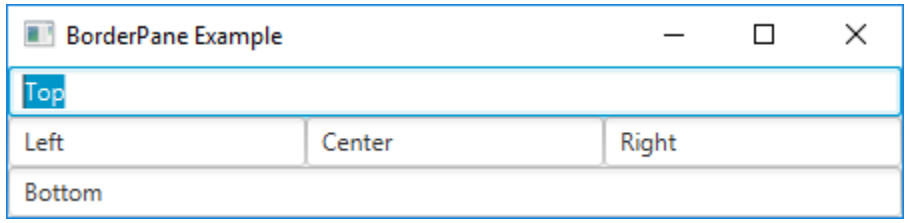
```
-top: ObjectProperty<Node>
-right: ObjectProperty<Node>
-bottom: ObjectProperty<Node>
-left: ObjectProperty<Node>
-center: ObjectProperty<Node>
```

```
+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`).

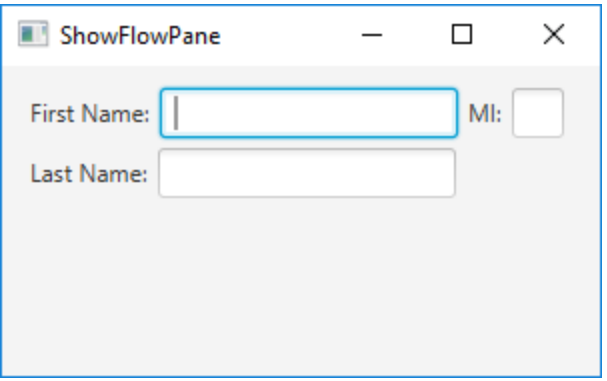
Creates a `BorderPane`.

Sets the alignment of the node in the `BorderPane`.




```
BorderPane bPane = new BorderPane();

    //Setting the top, bottom, center, right
and left nodes to the pane
    bPane.setTop(new TextField("Top"));
    bPane.setBottom(new TextField("Bottom"));
    bPane.setLeft(new TextField("Left"));
    bPane.setRight(new TextField("Right"));
    bPane.setCenter(new TextField("Center"));
```

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 getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

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).

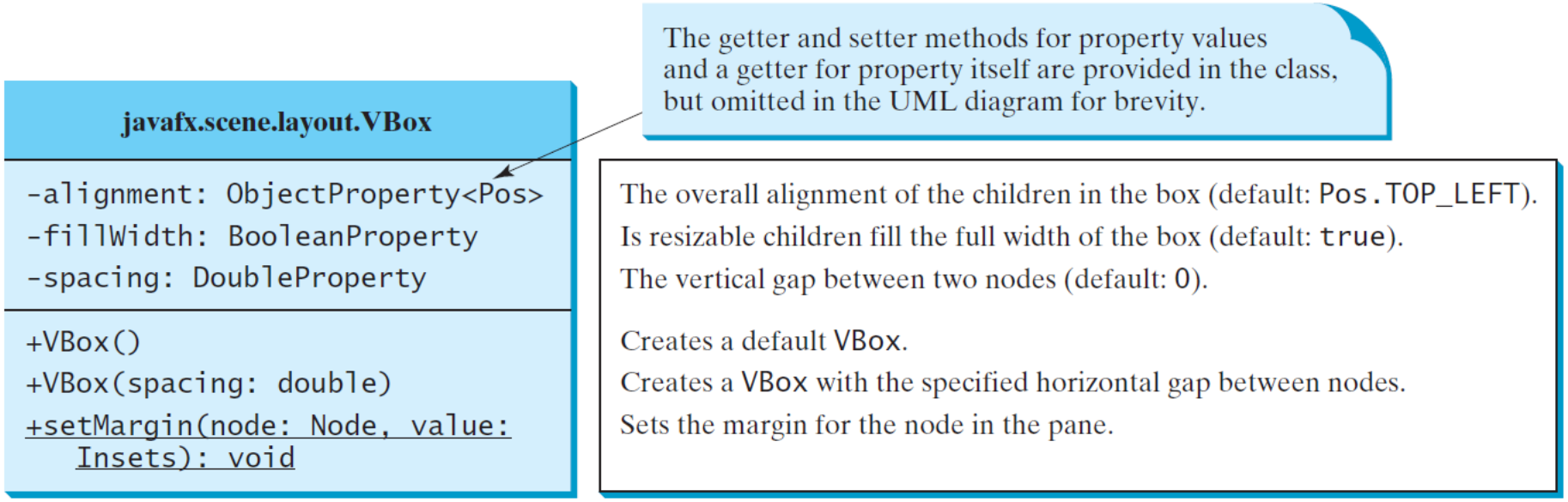
Creates a default `FlowPane`.
Creates a `FlowPane` with a specified horizontal and vertical gap.

Creates a `FlowPane` with a specified orientation.

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

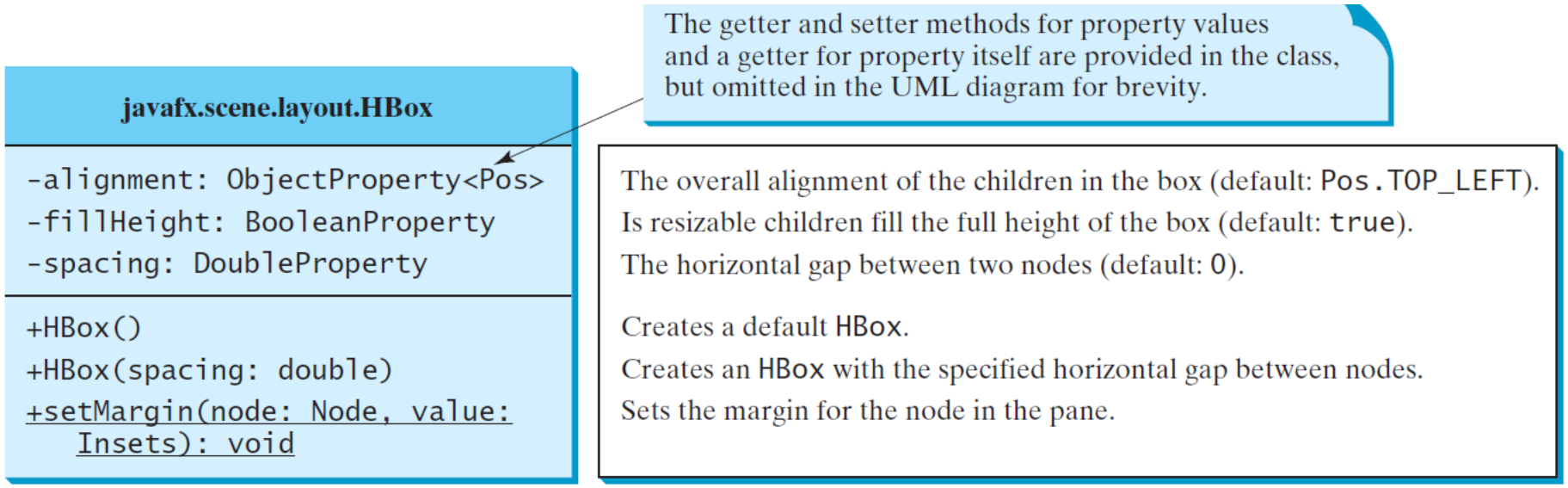
```
FlowPane pane = new FlowPane();
    pane.setPadding(new Insets(11, 12, 13, 14));
    pane.setHgap(5);
    pane.setVgap(5);

    // Place nodes in the pane
    pane.getChildren().addAll(new Label("First Name:"),
      new TextField(), new Label("MI:"));
    TextField tfMi = new TextField();
    tfMi.setPrefColumnCount(1);
    pane.getChildren().addAll(tfMi, new Label("Last Name:"),
      new TextField());
```

VBox



ShowHBoxVBox Run

HBox


javafx.scene.layout.HBox
-alignment: ObjectProperty<Pos>
-fillHeight: BooleanProperty
-spacing: DoubleProperty
+HBox()
+HBox(spacing: double)
+setMargin(node: Node, value: Insets): void
The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.
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.

```
public class ShowHBoxVBox extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // Create a border pane
    BorderPane pane = new BorderPane();
private HBox getHBox() {
    HBox hBox = new HBox(15);
    hBox.setPadding(new Insets(15, 15, 15, 15));
    hBox.setStyle("-fx-background-color: gold");
    hBox.getChildren().add(new Button("Computer Science"));
    hBox.getChildren().add(new Button("Chemistry"));
    ImageView imageView = new ImageView(new Image("image/us.gif"));
    hBox.getChildren().add(imageView);
    return hBox;
  }
    // Place nodes in the pane
    pane.setTop(getHBox());
    pane.setLeft(getVBox());
Scene scene = new Scene(pane);
    primaryStage.setTitle("ShowHBoxVBox"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }
```

```
  private VBox getVBox() {
    VBox vBox = new VBox(15);
    vBox.setPadding(new Insets(15, 5, 5, 5));
    vBox.getChildren().add(new Label("Courses"));

    Label[] courses = {new Label("CSCI 1301"), new Label("CSCI 1302"),
        new Label("CSCI 2410"), new Label("CSCI 3720")};

    for (Label course: courses) {
      VBox.setMargin(course, new Insets(0, 0, 0, 15));
      vBox.getChildren().add(course);
    }

    return vBox;
  }

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

Buidling an example in class :