# Chapter 3 JavaFX Basics

# Objectives

- To distinguish between JavaFX, Swing, and AWT
- To write a simple JavaFX program and understand the relationship among stages, scenes, and nodes
- To create user interfaces using panes
- To use the common properties **style** and **rotate** for nodes
- To create colors using the **Color** class
- To create fonts using the **Font** class
- To create images using the **Image** class and to create image views using the **ImageView** class
- To layout nodes using **Pane**, **StackPane**, **FlowPane**, **GridPane**, **BorderPane**, **HBox**, and **VBox**

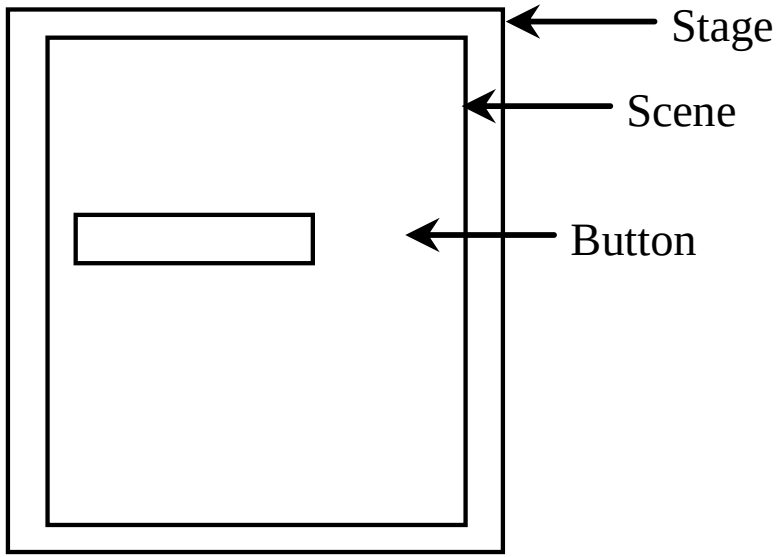
# JavaFX vs Swing and AWT

Swing and AWT are replaced by the JavaFX platform for developing rich Internet applications.

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*.

# Basic Structure of JavaFX

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



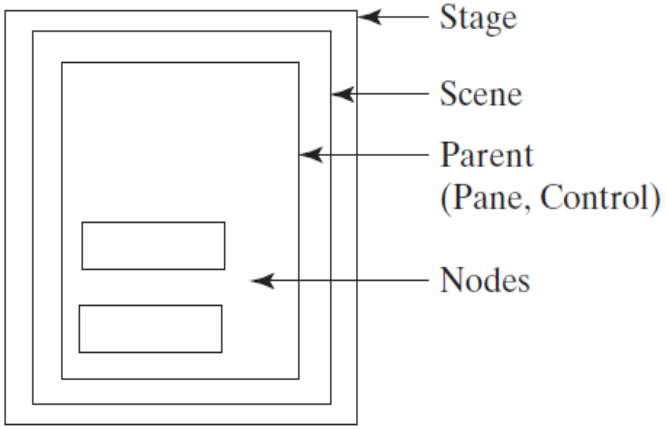
# Basic Structure of JavaFX


```
1  import javafx.application.Application;
2  import javafx.scene.Scene;
3  import javafx.scene.control.Button;
4  import javafx.stage.Stage;
5
6  public class MyJavaFX extends Application {
7    @Override // Override the start method in the Application class
8    public void start(Stage primaryStage) {
9      // Create a scene and place a button in the scene
10     Button btOK = new Button("OK");
11     Scene scene = new Scene(btOK, 200, 250);
12     primaryStage.setTitle("MyJavaFX"); // Set the stage title
13     primaryStage.setScene(scene); // Place the scene in the stage
14     primaryStage.show(); // Display the stage
15   }
16
17   /**
18    * The main method is only needed for the IDE with limited
19    * JavaFX support. Not needed for running from the command line.
20    */
21   public static void main(String[] args) {
22     Application.launch(args);
23   }
24 }
```

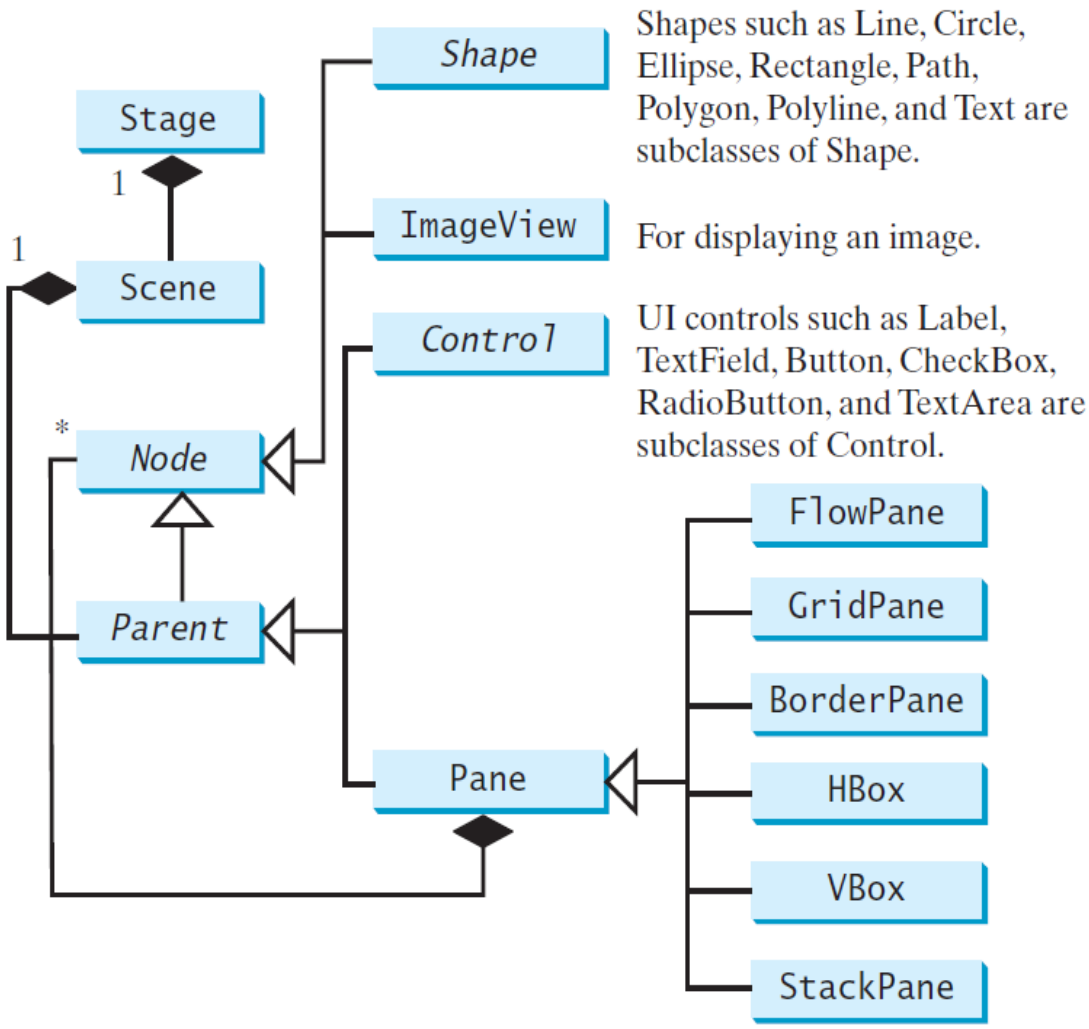
# Basic Structure of JavaFX


```
1  import javafx.application.Application;
2  import javafx.scene.Scene;
3  import javafx.scene.control.Button;
4  import javafx.stage.Stage;
5
6  public class MultipleStageDemo extends Application {
7    @Override // Override the start method in the Application class
8    public void start(Stage primaryStage) {
9      // Create a scene and place a button in the scene
10     Scene scene = new Scene(new Button("OK"), 200, 250);
11     primaryStage.setTitle("MyJavaFX"); // Set the stage title
12     primaryStage.setScene(scene); // Place the scene in the stage
13     primaryStage.show(); // Display the stage
14
15     Stage stage = new Stage(); // Create a new stage
16     stage.setTitle("Second Stage"); // Set the stage title
17     // Set a scene with a button in the stage
18     stage.setScene(new Scene(new Button("New Stage"), 100, 100));
19     stage.show(); // Display the stage
20   }
21 }
```

# Panes, UI Controls, and Shapes




(a)



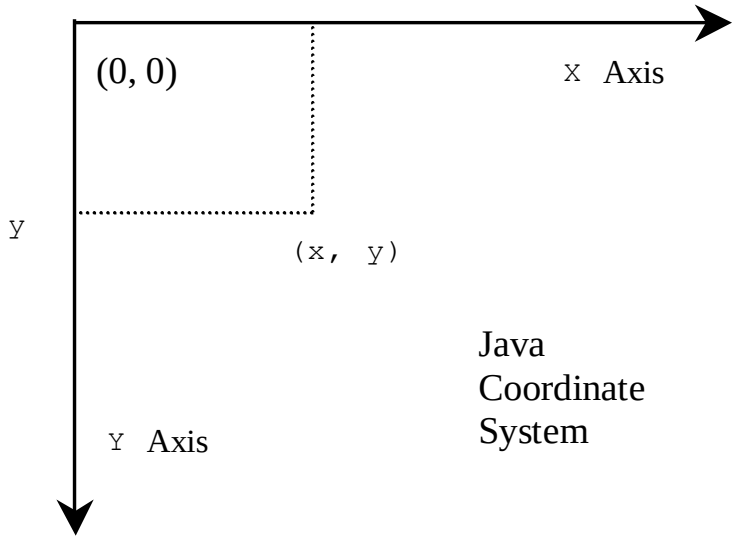

(b)

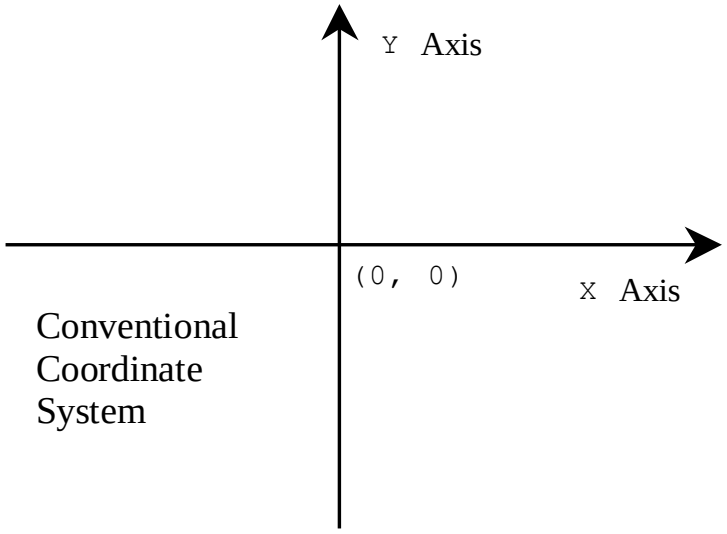
# Panes, UI Controls, and Shapes


```
1  import javafx.application.Application;
2  import javafx.scene.Scene;
3  import javafx.scene.control.Button;
4  import javafx.stage.Stage;
5  import javafx.scene.layout.StackPane;
6
7  public class ButtonInPane extends Application {
8    @Override // Override the start method in the Application class
9    public void start(Stage primaryStage) {
10     // Create a scene and place a button in the scene
11     StackPane pane = new StackPane();
12     pane.getChildren().add(new Button("OK"));
13     Scene scene = new Scene(pane, 200, 50);
14     primaryStage.setTitle("Button in a pane"); // Set the stage title
15     primaryStage.setScene(scene); // Place the scene in the stage
16     primaryStage.show(); // Display the stage
17   }
18 }
```

# Display a Shape

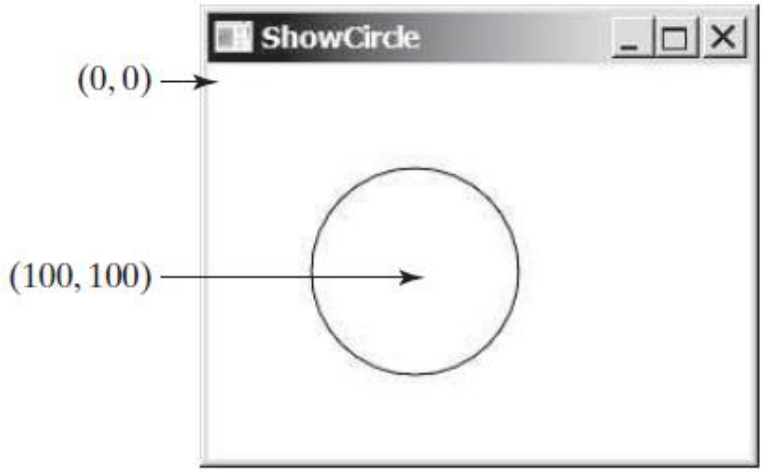
This example displays a circle in the center of the pane.



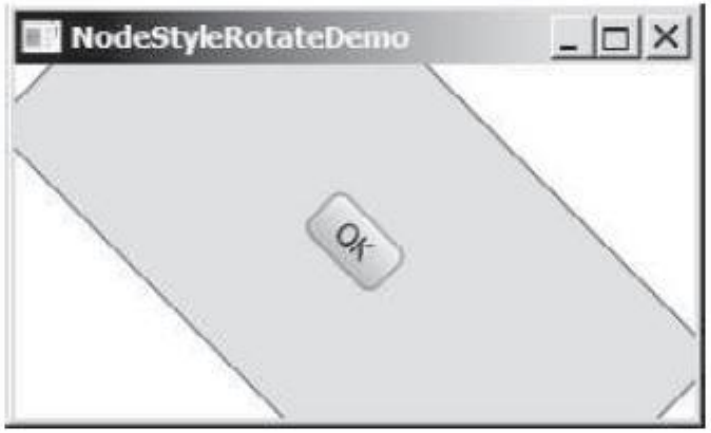
# Display a Shape

```
1  import javafx.application.Application;
2  import javafx.scene.Scene;
3  import javafx.scene.layout.Pane;
4  import javafx.scene.paint.Color;
5  import javafx.scene.shape.Circle;
6  import javafx.stage.Stage;
7
8  public class ShowCircle extends Application {
9    @Override // Override the start method in the Application class
10   public void start(Stage primaryStage) {
11     // Create a circle and set its properties
12     Circle circle = new Circle();
13     circle.setCenterX(100);
14     circle.setCenterY(100);
15     circle.setRadius(50);
16     circle.setStroke(Color.BLACK);
17     circle.setFill(Color.WHITE);
18
19     // Create a pane to hold the circle
20     Pane pane = new Pane();
21     pane.getChildren().add(circle);
22
23     // Create a scene and place it in the stage
24     Scene scene = new Scene(pane, 200, 200);
25     primaryStage.setTitle("ShowCircle"); // Set the stage title
26     primaryStage.setScene(scene); // Place the scene in the stage
27     primaryStage.show(); // Display the stage
28   }
29 }
```

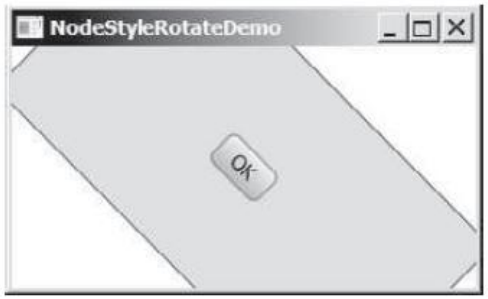


# Common Properties and Methods for Nodes

- style: set a JavaFX CSS style
- rotate: Rotate a node

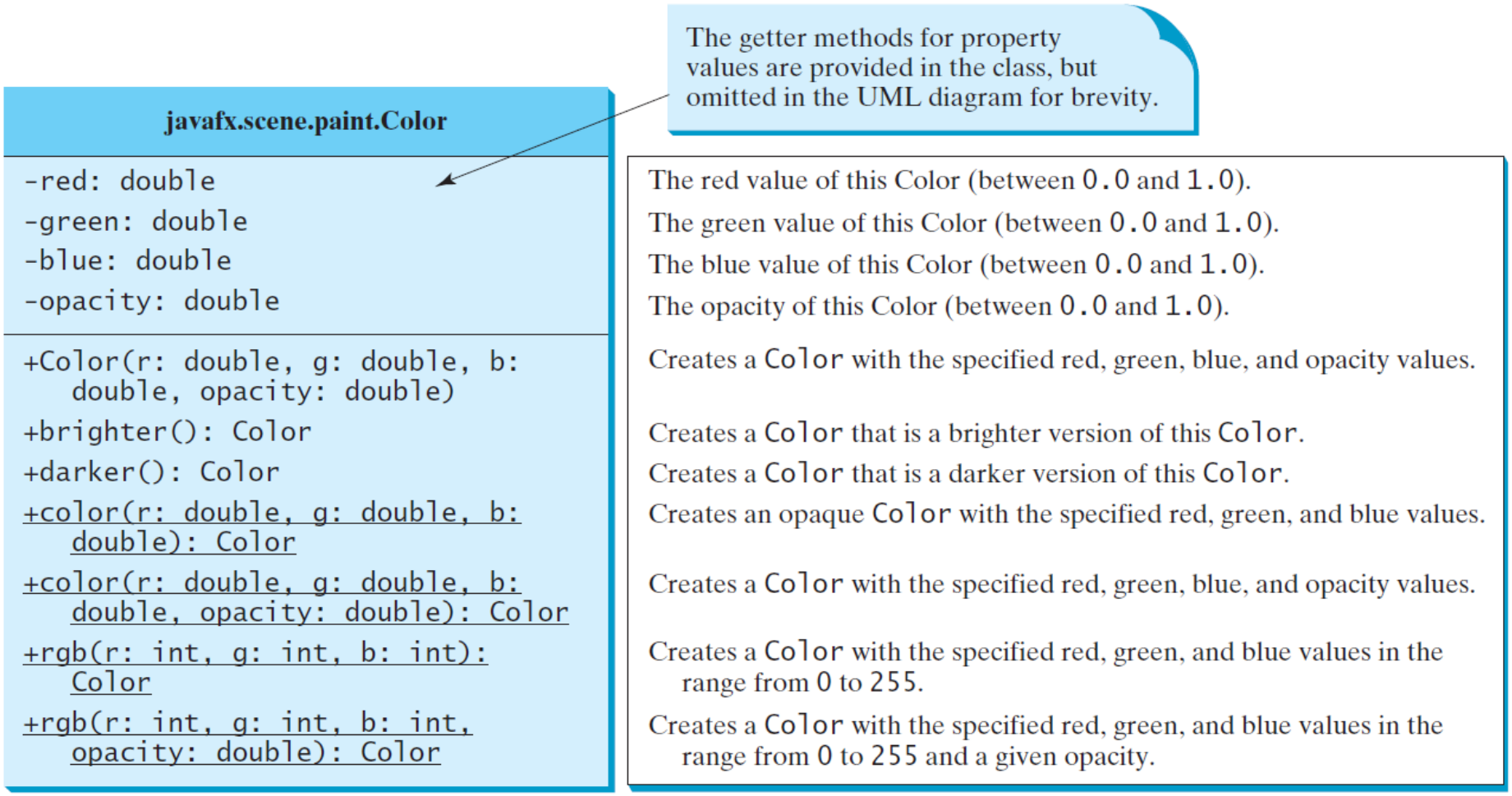


# Common Properties and Methods for Nodes

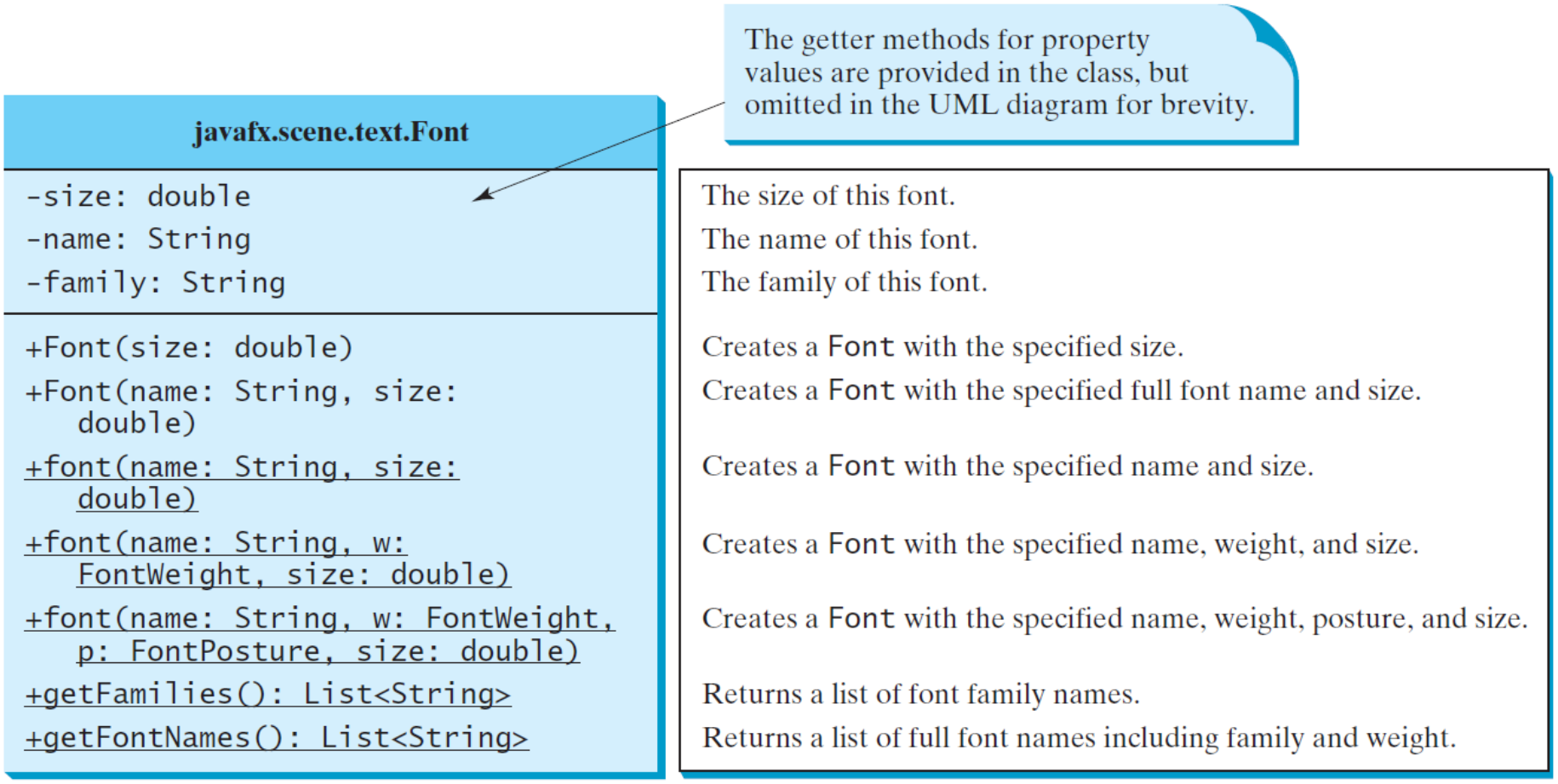



```
1  import javafx.application.Application;
2  import javafx.scene.Scene;
3  import javafx.scene.control.Button;
4  import javafx.stage.Stage;
5  import javafx.scene.layout.StackPane;
6
7  public class NodeStyleRotateDemo extends Application {
8    @Override // Override the start method in the Application class
9    public void start(Stage primaryStage) {
10     // Create a scene and place a button in the scene
11     StackPane pane = new StackPane();
12     Button btOK = new Button("OK");
13     btOK.setStyle("-fx-border-color: blue;");
14     pane.getChildren().add(btOK);
15
16     pane.setRotate(45);
17     pane.setStyle(
18       "-fx-border-color: red; -fx-background-color: lightgray;");
19
20     Scene scene = new Scene(pane, 200, 250);
21     primaryStage.setTitle("NodeStyleRotateDemo"); // Set the stage title
22     primaryStage.setScene(scene); // Place the scene in the stage
23     primaryStage.show(); // Display the stage
24   }
25 }
```

# The Color Class



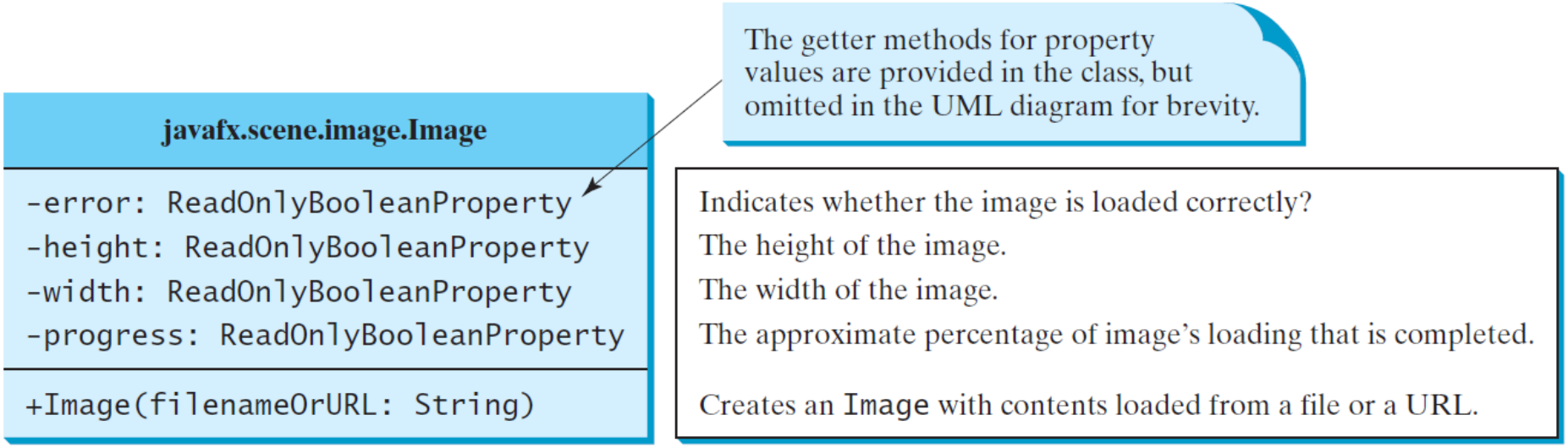
# The Font Class



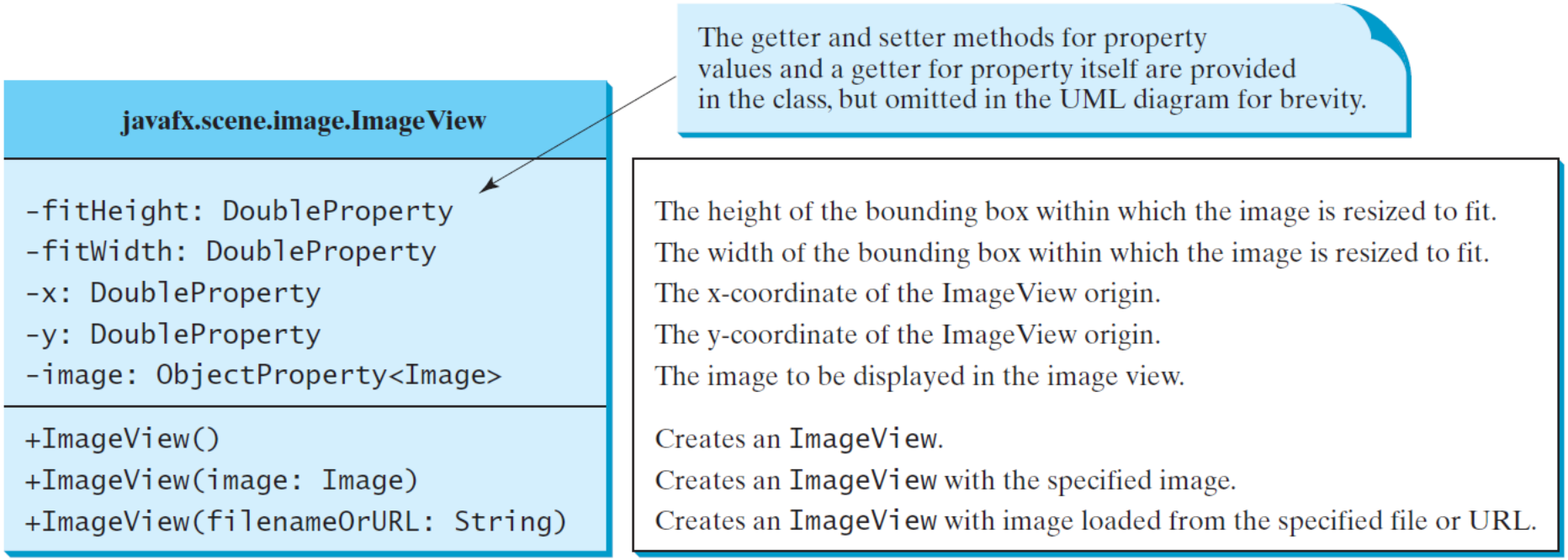
# The Font Class

```
1  import javafx.application.Application;
2  import javafx.scene.Scene;
3  import javafx.scene.layout.*;
4  import javafx.scene.paint.Color;
5  import javafx.scene.shape.Circle;
6  import javafx.scene.text.*;
7  import javafx.scene.control.*;
8  import javafx.stage.Stage;
9
10 public class FontDemo extends Application {
11   @Override // Override the start method in the Application class
12   public void start(Stage primaryStage) {
13     // Create a pane to hold the circle
14     Pane pane = new StackPane();
15
16     // Create a circle and set its properties
17     Circle circle = new Circle();
18     circle.setRadius(50);
19     circle.setStroke(Color.BLACK);
20     circle.setFill(new Color(0.5, 0.5, 0.5, 0.1));
21     pane.getChildren().add(circle); // Add circle to the pane
22
23     // Create a label and set its properties
24     Label label = new Label("JavaFX");
25     label.setFont(Font.font("Times New Roman",
26       FontWeight.BOLD, FontPosture.ITALIC, 20));
27     pane.getChildren().add(label);
28
29     // Create a scene and place it in the stage
30     Scene scene = new Scene(pane);
31     primaryStage.setTitle("FontDemo"); // Set the stage title
32     primaryStage.setScene(scene); // Place the scene in the stage
33     primaryStage.show(); // Display the stage
34   }
35 }
```

# The Image Class



# The ImageView Class



# The ImageView Class

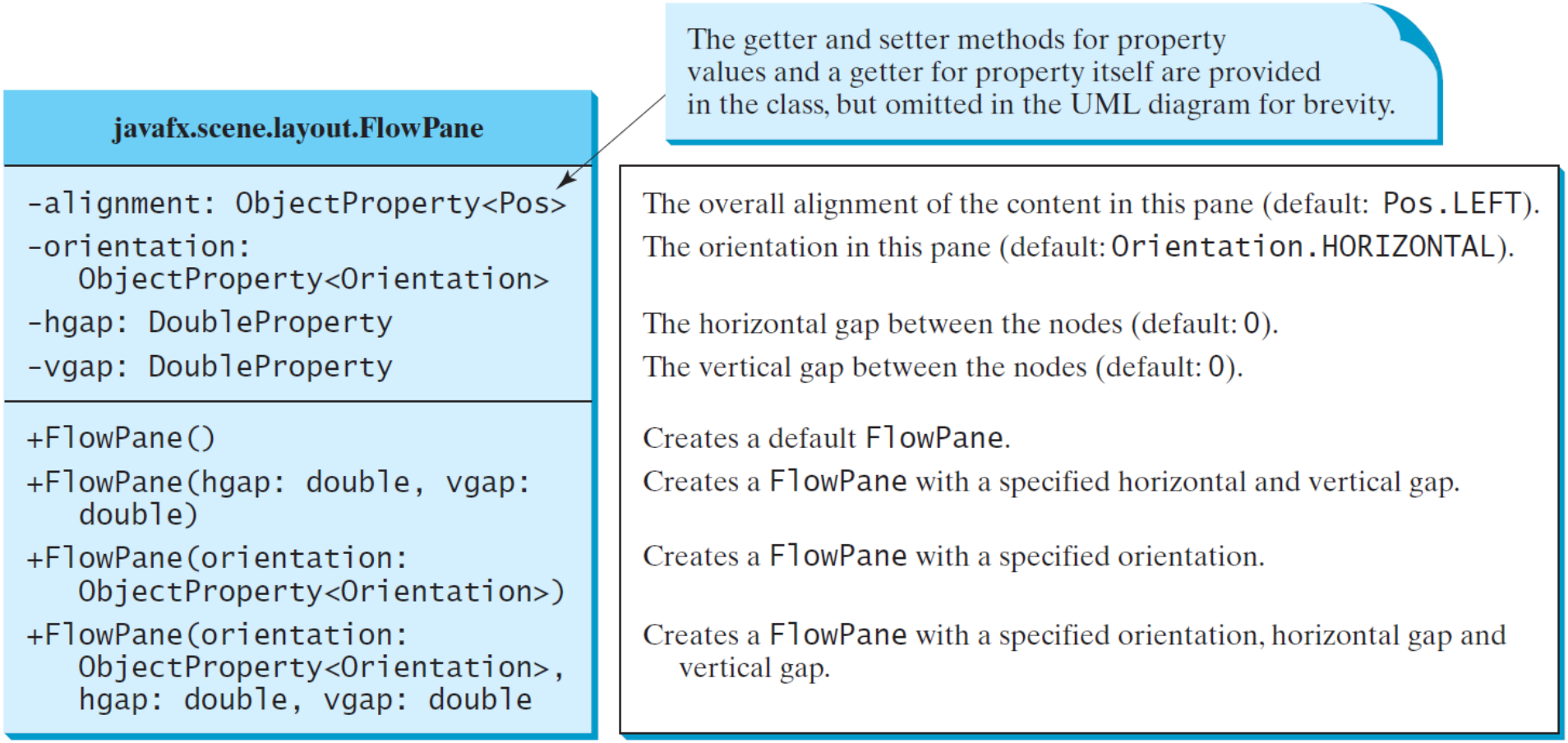
```
1  import java.io.FileInputStream;
2  import java.io.FileNotFoundException;
3  import javafx.application.Application;
4  import javafx.scene.Scene;
5  import javafx.scene.layout.HBox;
6  import javafx.scene.layout.Pane;
7  import javafx.geometry.Insets;
8  import javafx.stage.Stage;
9  import javafx.scene.image.Image;
10 import javafx.scene.image.ImageView;
11
12 public class ShowImage extends Application {
13   @Override // Override the start method in the Application class
14   public void start(Stage primaryStage) throws FileNotFoundException {
15     // Create a pane to hold the image views
16     Pane pane = new HBox(10);
17     pane.setPadding(new Insets(5, 5, 5, 5));
18     FileInputStream imageStream = new FileInputStream("image/us.gif");
19     Image image = new Image(imageStream);
20
21     pane.getChildren().add(new ImageView(image));
22
23     ImageView imageView2 = new ImageView(image);
24     imageView2.setFitHeight(100);
25     imageView2.setFitWidth(100);
26     pane.getChildren().add(imageView2);
27
28     ImageView imageView3 = new ImageView(image);
29     imageView3.setRotate(90);
30     pane.getChildren().add(imageView3);
31
32     // Create a scene and place it in the stage
33     Scene scene = new Scene(pane);
34     primaryStage.setTitle("ShowImage"); // Set the stage title
35     primaryStage.setScene(scene); // Place the scene in the stage
36     primaryStage.show(); // Display the stage
37   }
38 }
```

# Layout Panes

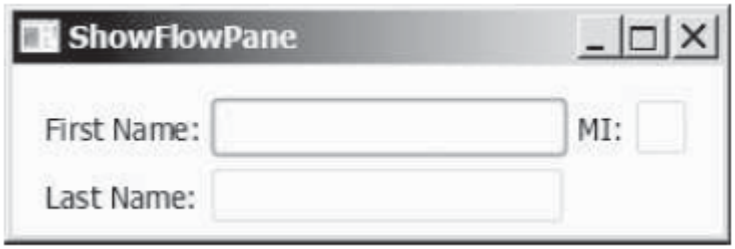
JavaFX provides many types of panes for organizing nodes in a container.

| *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. |

# 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




```
1  import javafx.application.Application;
2  import javafx.geometry.Insets;
3  import javafx.scene.Scene;
4  import javafx.scene.control.Label;
5  import javafx.scene.control.TextField;
6  import javafx.scene.layout.FlowPane;
7  import javafx.stage.Stage;
8
9  public class ShowFlowPane extends Application {
10   @Override // Override the start method in the Application class
11   public void start(Stage primaryStage) {
12     // Create a pane and set its properties
13     FlowPane pane = new FlowPane();
14     pane.setPadding(new Insets(11, 12, 13, 14));
15     pane.setHgap(5);
16     pane.setVgap(5);
17
18     // Place nodes in the pane
19     pane.getChildren().addAll(new Label("First Name:"),
20       new TextField(), new Label("MI:"));
21     TextField tfMi = new TextField();
22     tfMi.setPrefColumnCount(1);
23     pane.getChildren().addAll(tfMi, new Label("Last Name:"),
24       new TextField());
25
26     // Create a scene and place it in the stage
27     Scene scene = new Scene(pane, 200, 250);
28     primaryStage.setTitle("ShowFlowPane"); // Set the stage title
29     primaryStage.setScene(scene); // Place the scene in the stage
30     primaryStage.show(); // Display the stage
31   }
32 }
```

# GridPane

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.

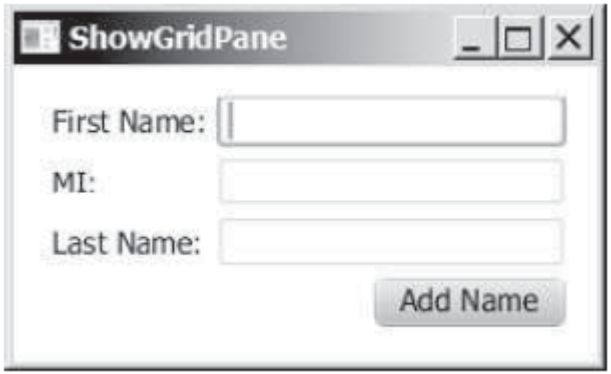
**javafx.scene.layout.GridPane**

- `-alignment: ObjectProperty<Pos>` — The overall alignment of the content in this pane (default: `Pos.LEFT`).
- `-gridLinesVisible: BooleanProperty` — Is the grid line visible? (default: `false`)
- `-hgap: DoubleProperty` — The horizontal gap between the nodes (default: 0).
- `-vgap: DoubleProperty` — The vertical gap between the nodes (default: 0).

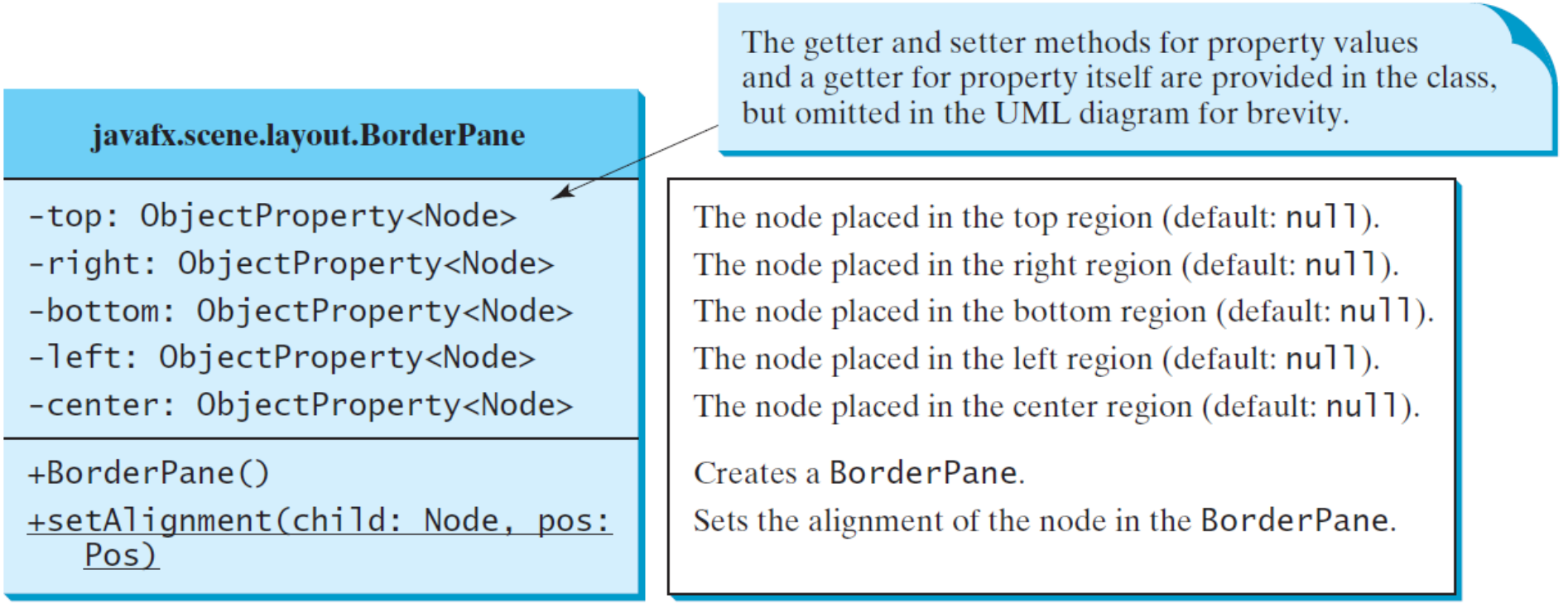
- `+GridPane()` — Creates a `GridPane`.
- `+add(child: Node, columnIndex: int, rowIndex: int): void` — Adds a node to the specified column and row.
- `+addColumn(columnIndex: int, children: Node...): void` — Adds multiple nodes to the specified column.
- `+addRow(rowIndex: int, children: Node...): void` — Adds multiple nodes to the specified row.
- `+getColumnIndex(child: Node): int` — Returns the column index for the specified node.
- `+setColumnIndex(child: Node, columnIndex: int): void` — Sets a node to a new column. This method repositions the node.
- `+getRowIndex(child:Node): int` — Returns the row index for the specified node.
- `+setRowIndex(child: Node, rowIndex: int): void` — Sets a node to a new row. This method repositions the node.
- `+setHalighnment(child: Node, value: HPos): void` — Sets the horizontal alignment for the child in the cell.
- `+setValighnment(child: Node, value: VPos): void` — Sets the vertical alignment for the child in the cell.

# GridPane

```
1  import javafx.application.Application;
2  import javafx.geometry.HPos;
3  import javafx.geometry.Insets;
4  import javafx.geometry.Pos;
5  import javafx.scene.Scene;
6  import javafx.scene.control.Button;
7  import javafx.scene.control.Label;
8  import javafx.scene.control.TextField;
9  import javafx.scene.layout.GridPane;
10 import javafx.stage.Stage;
11
12 public class ShowGridPane extends Application {
13   @Override // Override the start method in the Application class
14   public void start(Stage primaryStage) {
15     // Create a pane and set its properties
16     GridPane pane = new GridPane();
17     pane.setAlignment(Pos.CENTER);
18     pane.setPadding(new Insets(11.5, 12.5, 13.5, 14.5));
19     pane.setHgap(5.5);
20     pane.setVgap(5.5);
21
22     // Place nodes in the pane
23     pane.add(new Label("First Name:"), 0, 0);
24     pane.add(new TextField(), 1, 0);
25     pane.add(new Label("MI:"), 0, 1);
26     pane.add(new TextField(), 1, 1);
27     pane.add(new Label("Last Name:"), 0, 2);
28     pane.add(new TextField(), 1, 2);
29     Button btAdd = new Button("Add Name");
30     pane.add(btAdd, 1, 3);
31     GridPane.setHalignment(btAdd, HPos.RIGHT);
32
33     // Create a scene and place it in the stage
34     Scene scene = new Scene(pane);
35     primaryStage.setTitle("ShowGridPane"); // Set the stage title
36     primaryStage.setScene(scene); // Place the scene in the stage
37     primaryStage.show(); // Display the stage
38   }
39 }
```

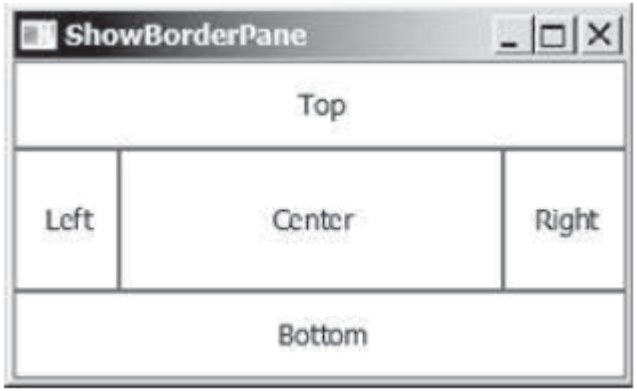


# BorderPane

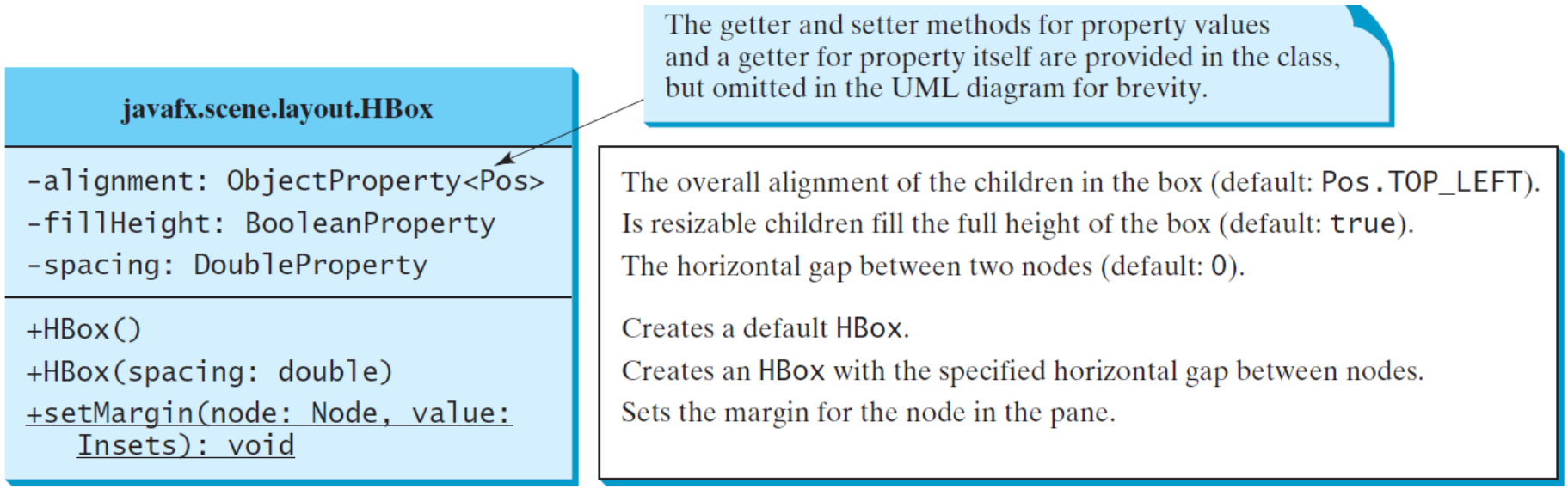

javafx.scene.layout.BorderPane
-top: ObjectProperty<Node>
-right: ObjectProperty<Node>
-bottom: ObjectProperty<Node>
-left: ObjectProperty<Node>
-center: ObjectProperty<Node>
+BorderPane()
+setAlignment(child: Node, pos: Pos)
The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.
The 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

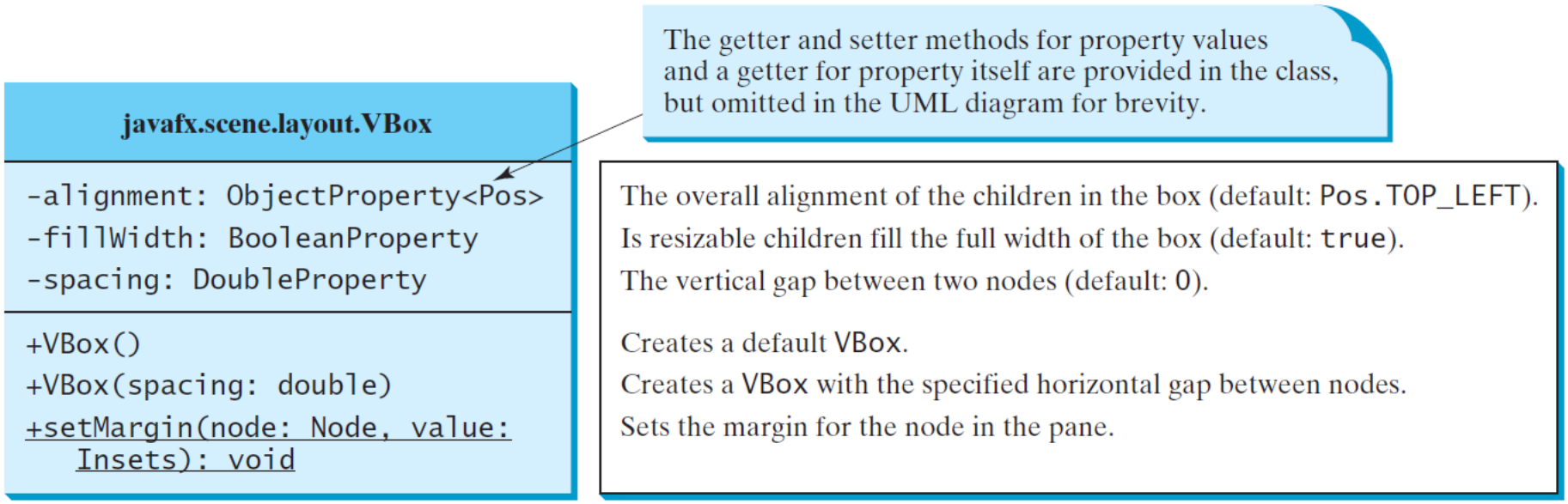
```
1  import javafx.application.Application;
2  import javafx.geometry.Insets;
3  import javafx.scene.Scene;
4  import javafx.scene.control.Label;
5  import javafx.scene.layout.BorderPane;
6  import javafx.scene.layout.StackPane;
7  import javafx.stage.Stage;
8
9  public class ShowBorderPane extends Application {
10   @Override // Override the start method in the Application class
11   public void start(Stage primaryStage) {
12     // Create a border pane
13     BorderPane pane = new BorderPane();
14
15     // Place nodes in the pane
16     pane.setTop(new CustomPane("Top"));
17     pane.setRight(new CustomPane("Right"));
18     pane.setBottom(new CustomPane("Bottom"));
19     pane.setLeft(new CustomPane("Left"));
20     pane.setCenter(new CustomPane("Center"));
21
22     // Create a scene and place it in the stage
23     Scene scene = new Scene(pane);
24     primaryStage.setTitle("ShowBorderPane"); // Set the stage title
25     primaryStage.setScene(scene); // Place the scene in the stage
26     primaryStage.show(); // Display the stage
27   }
28 }
29
30 // Define a custom pane to hold a label in the center of the pane
31 class CustomPane extends StackPane {
32   public CustomPane(String title) {
33     getChildren().add(new Label(title));
34     setStyle("-fx-border-color: red");
35     setPadding(new Insets(11.5, 12.5, 13.5, 14.5));
36   }
37 }
```



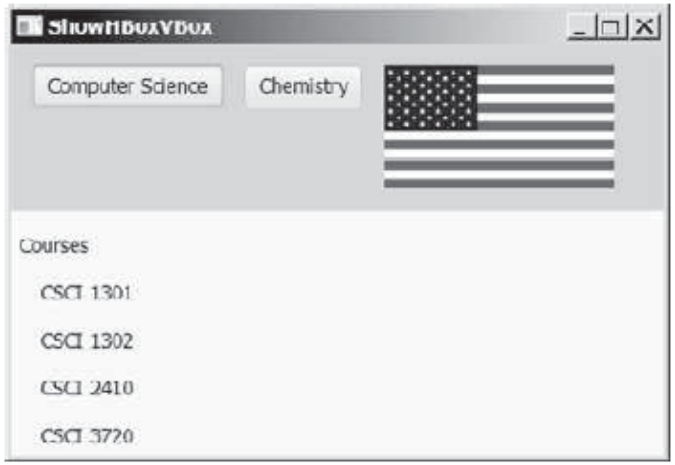
# HBox


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.

# VBox


javafx.scene.layout.VBox
-alignment: ObjectProperty<Pos>
-fillWidth: BooleanProperty
-spacing: DoubleProperty
+VBox()
+VBox(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 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.

# VBox




```
1  import javafx.application.Application;
2  import javafx.geometry.Insets;
3  import javafx.scene.Scene;
4  import javafx.scene.control.Button;
5  import javafx.scene.control.Label;
6  import javafx.scene.layout.BorderPane;
7  import javafx.scene.layout.HBox;
8  import javafx.scene.layout.VBox;
9  import javafx.stage.Stage;
10 import javafx.scene.image.Image;
11 import javafx.scene.image.ImageView;
12
13                                  ds Application {
14   @Override // Override the start method in the Application class
15   public void start(Stage primaryStage) {
16     // Create a border pane
17     BorderPane pane = new BorderPane();
18
19     // Place nodes in the pane
20     pane.setTop(getHBox());
21     pane.setLeft(getVBox());
22
23     // Create a scene and place it in the stage
24     Scene scene = new Scene(pane);
25     primaryStage.setTitle("ShowHBoxVBox"); // Set the stage title
26     primaryStage.setScene(scene); // Place the scene in the stage
27     primaryStage.show(); // Display the stage
28   }
29
```

# VBox

```
30    private HBox getHBox() {
31      HBox hBox = new HBox(15);
32      hBox.setPadding(new Insets(15, 15, 15, 15));
33      hBox.setStyle("-fx-background-color: gold");
34      hBox.getChildren().add(new Button("Computer Science"));
35      hBox.getChildren().add(new Button("Chemistry"));
36      ImageView imageView = new ImageView(new Image("image/us.gif"));
37      hBox.getChildren().add(imageView);
38      return hBox;
39    }
40
41    private VBox getVBox() {
42      VBox vBox = new VBox(15);
43      vBox.setPadding(new Insets(15, 5, 5, 5));
44      vBox.getChildren().add(new Label("Courses"));
45
46      Label[] courses = {new Label("CSCI 1301"), new Label("CSCI 1302"),
47          new Label("CSCI 2410"), new Label("CSCI 3720")};
48
49      for (Label course: courses) {
50        VBox.setMargin(course, new Insets(0, 0, 0, 15));
51        vBox.getChildren().add(course);
52      }
53
54      return vBox;
55    }
56  }
```