Chapter 11 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.

For more information on JavaFX with Java SE 8, please refer to the JavaFX Documentation[https://docs.oracle.com/javase/8/javase-clienttechnologies.htm]. For JDK 11 and later releases, Oracle has open sourced JavaFX. You can find more information at OpenJFX project [https://openjfx.io/].

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.

JavaFX Application

The *entry point* for JavaFX applications is the Application class from which JavaFX applications extend

```
public abstract class Application {
    public static void launch(String... args) {}
    public void init() throws Exception {}
    public abstract void start(Stage primaryStage) throws Exception;
    public void stop() throws Exception {}
}
```

JavaFX Application Life-cycle

The JavaFX runtime does the following, in order, whenever an application is launched:

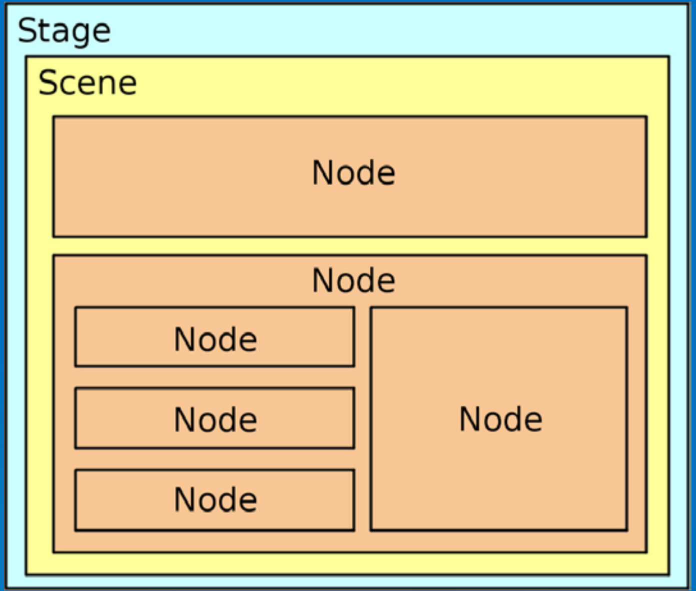
1. Constructs an instance of the specified Application class
2. Calls the `init()` method
3. Calls the `start(javafx.stage.Stage)` method
4. Waits for the application to finish, which happens when either of the following occur:
   - •the application calls `Platform.exit()`
   - •the last window has been closed and the implicitExit attribute on Platform is true
5. Calls the `stop()` method

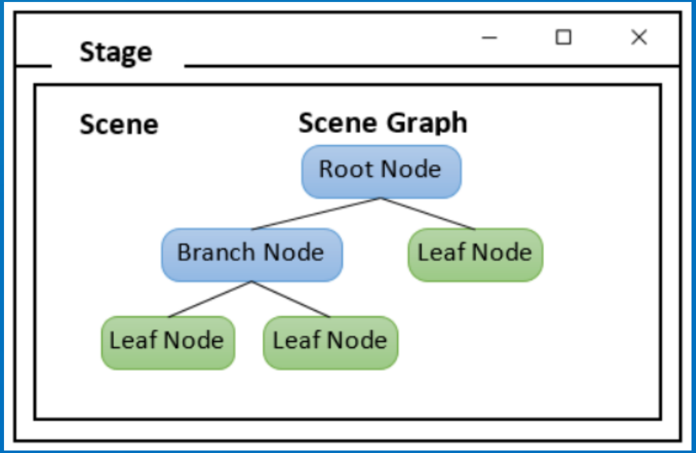
Life-cycle Demonstration

```
public class MyApp extends Application{
    public void start(Stage primaryStage) throws Exception {
        System.out.println("Call the start() method!");
        primaryStage.show();
    }
    public void init() throws Exception {
        System.out.println("Call the init() method!");
    }
    public void stop() throws Exception {
        System.out.println("Call the stop() method!");
    }
    public static void main(String[] args) {
        launch(args);
    }
}
```

```
Call the init() method!
Call the start() method!
Call the stop() method!
```

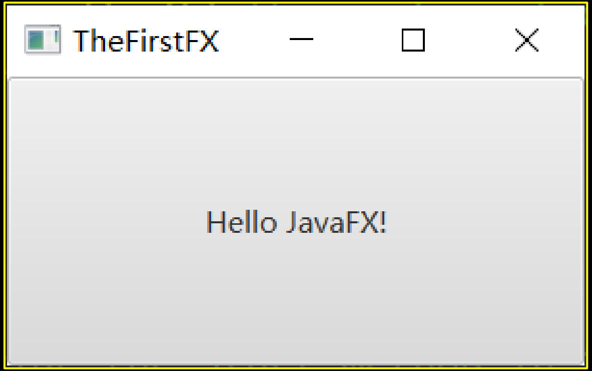
Element Organizing in an Application



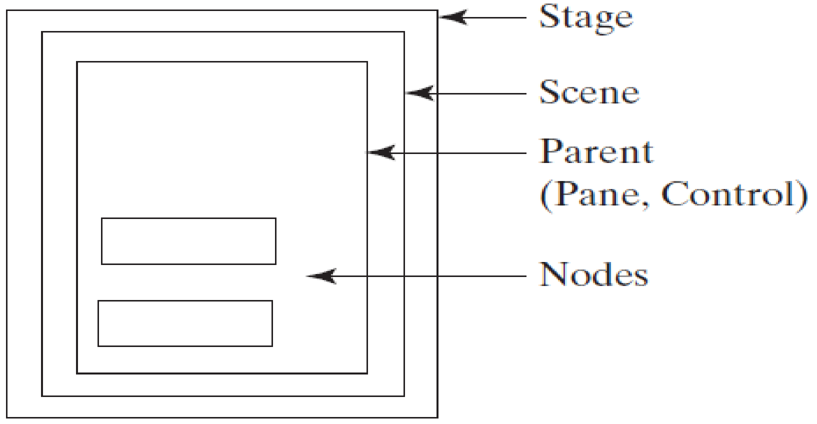


The First JavaFX Program

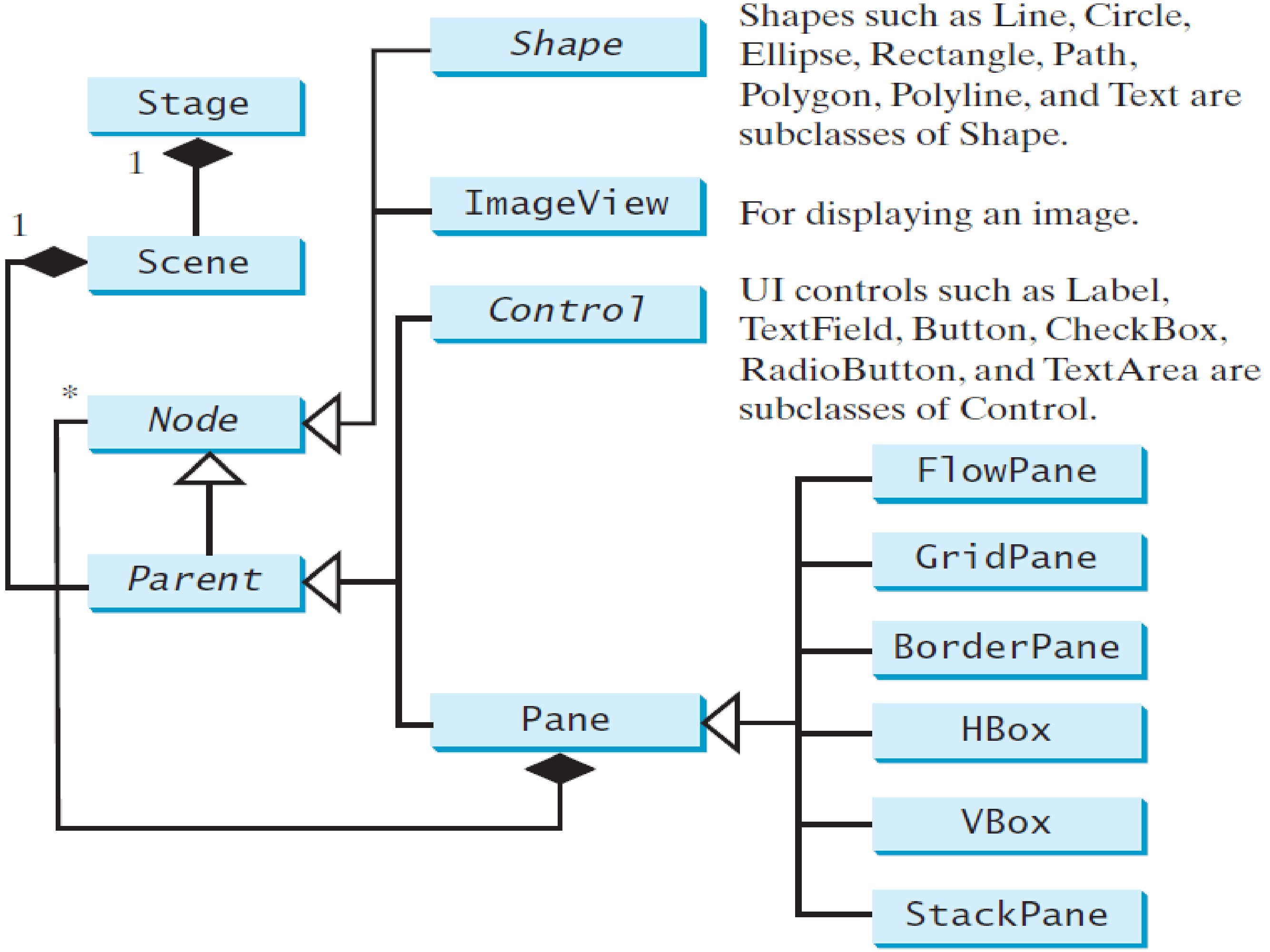
```
public class TheFirstFX extends Application {
    @Override // Override the start method in the Application class
    public void start(Stage primaryStage) {
        // Create a button and place it in the scene
        Button btOK = new Button("Hello JavaFX!");
        Scene scene = new Scene(btOK, 300, 150);
        primaryStage.setTitle("TheFirstFX"); // 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



(a)


Stage
1
1
Scene
*
Node
Parent
Shape
ImageView
Control
Pane
FlowPane
GridPane
BorderPane
HBox
VBox
StackPane
Shapes such as Line, Circle, Ellipse, Rectangle, Path, Polygon, Polyline, and Text are subclasses of Shape.
For displaying an image.
UI controls such as Label, TextField, Button, CheckBox, RadioButton, and TextArea are subclasses of Control.

(b)

Stage

```
public class Stage extends Window
```

The JavaFX Stage class is the *top level* JavaFX *container*. *The primary Stage is constructed by the platform*. *Additional Stage* objects may be constructed by the application. Stage objects must be constructed and modified on the JavaFX Application Thread. Many of the Stage properties are read only because they can be changed externally by the underlying platform and therefore must not be bindable. [The primary stage will be embedded in the browser if the application was launched as an applet.]

Scene

```
public class Scene extends Object implements EventTarget
```

The JavaFX *Scene* class is the *container* for all content in a scene graph. The *background* of the scene is fil led as specified by the *fill property*. The application must specify the *root Node* for the scene graph by setting the *root property*. If a Group is used as the root, the contents of the scene graph will be *clipped* by the scene's width and height and changes to the scene's size (if user resizes the stage) will not alter the layout of the scene graph. If a *resizable node* (layout Region or Control is set as the root, then the root's size will *track* the scene's size, causing the contents to be relayed out as necessary.

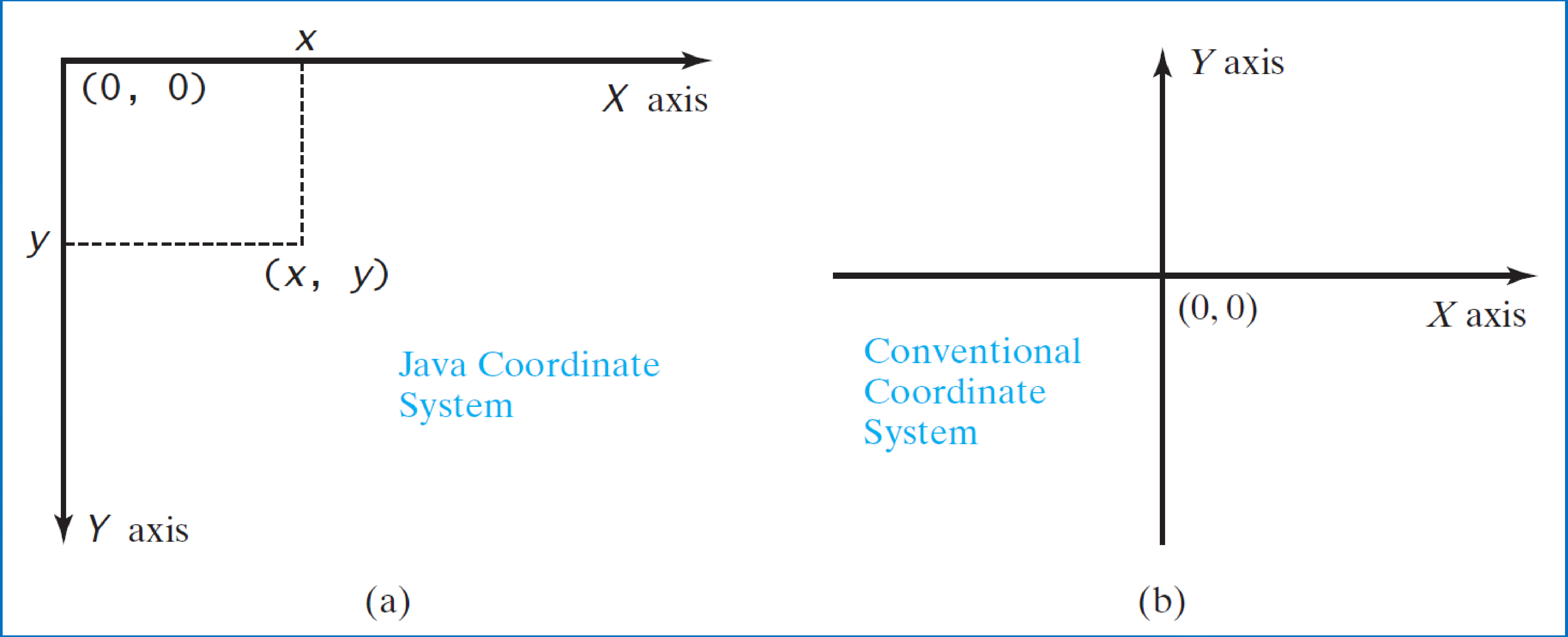
Node

```
public abstract class Node implements EventTarget, Styleable {}
```

A *scene* graph is a set of tree data structures where every item has zero or one *parent*, and each item is either a *“leaf”* with zero sub-items or a *“branch”* with zero or more sub-items. Each item in the scene graph is called a *Node*. Branch nodes are of type *Parent*, whose concrete subclasses are *Group*, *Region*, and *Control*, or subclasses thereof. *Leaf* nodes are classes such as *Rectangle*, *Text*, *ImageView*, *MediaView*, or other such leaf classes which cannot have children. Only a single node within each scene graph tree will have no parent, which is referred to as the *“root”* node.

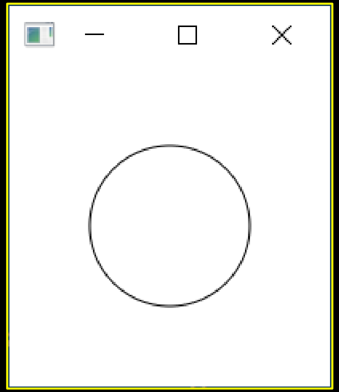
Java Coordinate System

The Java coordinate system is measured in pixels, with (0, 0) at its upper-left corner.



Display a Shape

```
// Create a circle and set its properties
Circle circle = new Circle();
circle.setCenterX(100);
circle.setCenterY(100);
circle.setRadius(50);
circle.setStroke(Color.BLACK);
circle.setFill(null);
// 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, 200, 200);
primaryStage.setTitle("ShowCircle"); // Set the stage title
primaryStage.setScene(scene); // Place the scene in the stage
primaryStage.show(); // Display the stage
```

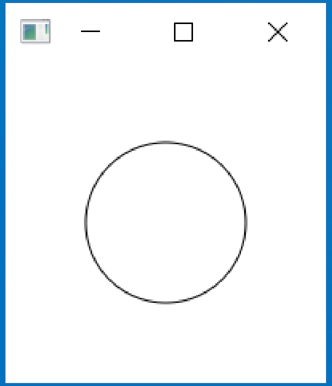


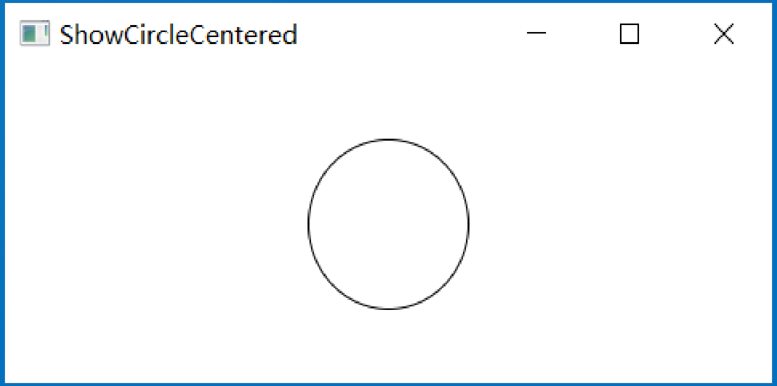
Binding Properties

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

```
// Create a pane to hold the circle
Pane pane = new Pane();
// Create a circle and set its properties
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); // Add circle to the pane
```



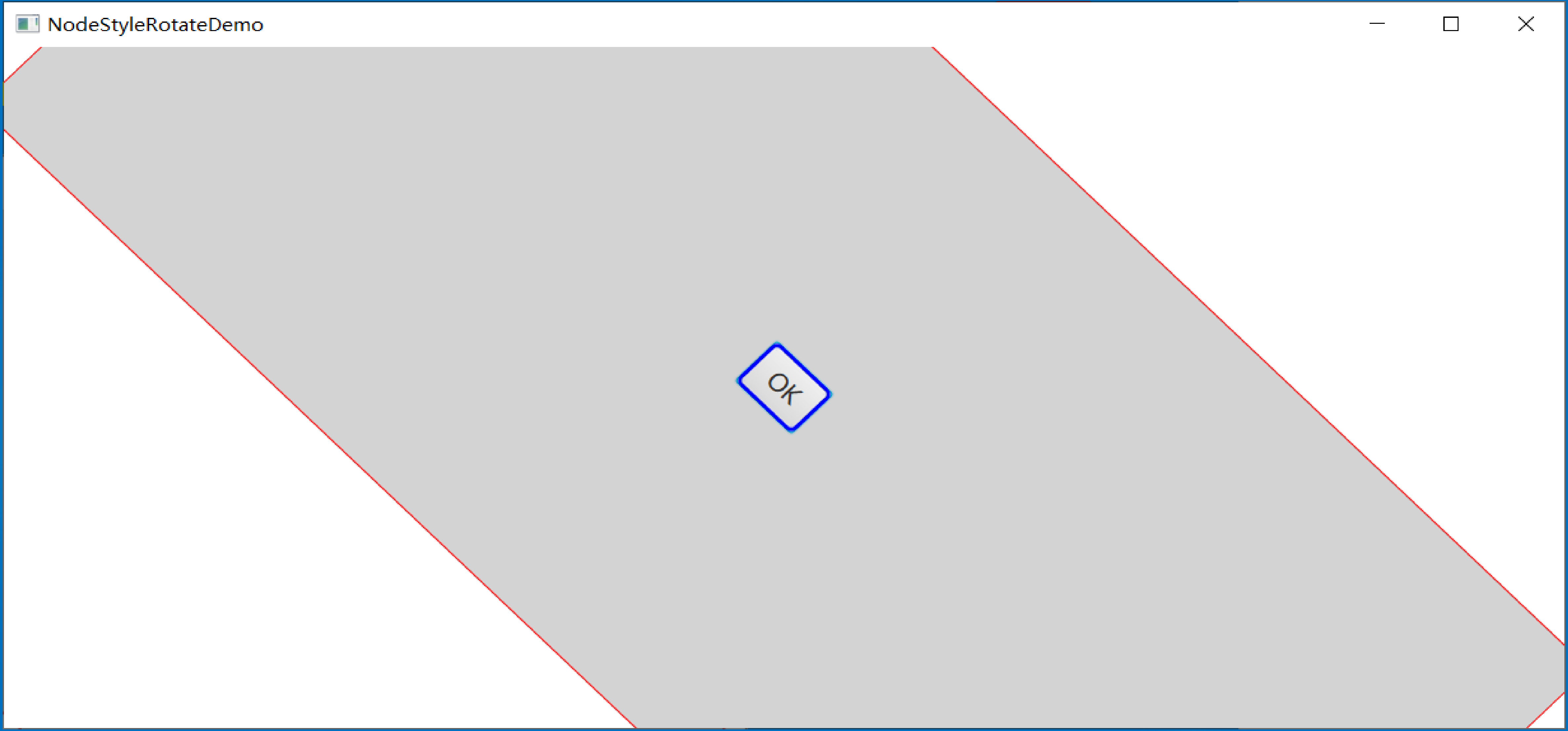


Binding Property: getter, setter, and property getter

```
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() { ... }
}
```

```
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() { ... }
}
```

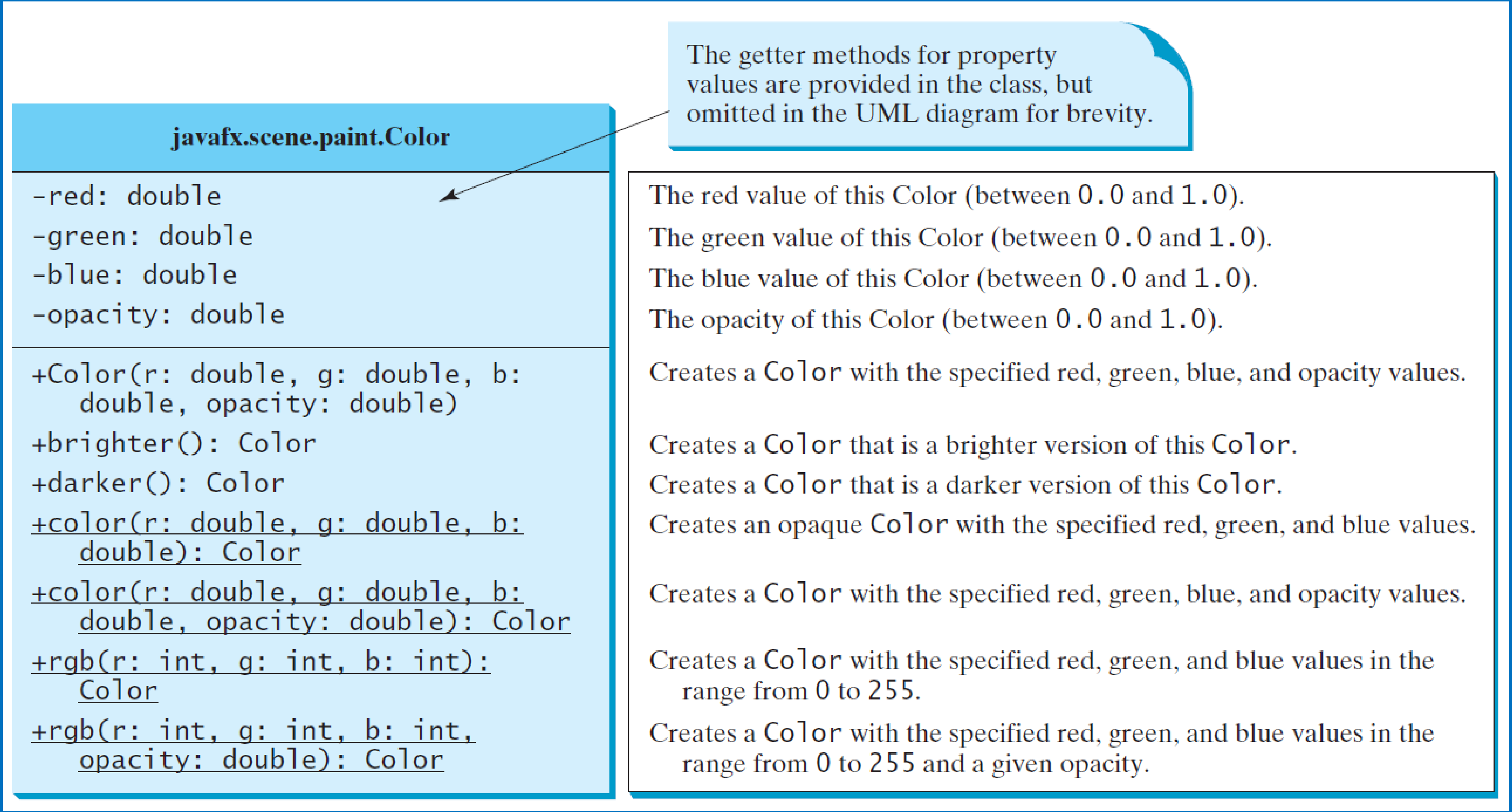
Common Properties and Methods for Nodes



Common Properties and Methods for Nodes

```
Button btOK = new Button("OK");
btOK.setFont(new Font("微软雅黑", 20));
BorderStroke borderStroke = new BorderStroke(
        Color.BLUE, BorderStrokeStyle.SOLID,
        new CornerRadii(5), new BorderWidths(3));
btOK.setBorder(new Border(borderStroke));
StackPane pane = new StackPane();
pane.getChildren().add(btOK);
borderStroke = new BorderStroke(
        Color.RED, BorderStrokeStyle.SOLID,
        new CornerRadii(0), new BorderWidths(1));
pane.setBorder(new Border(borderStroke));
pane.setBackground(new Background(
        new BackgroundFill(Color.LIGHTGRAY,
        new CornerRadii(0), Insets.EMPTY)));
pane.setRotate(45);
```

The Color Class

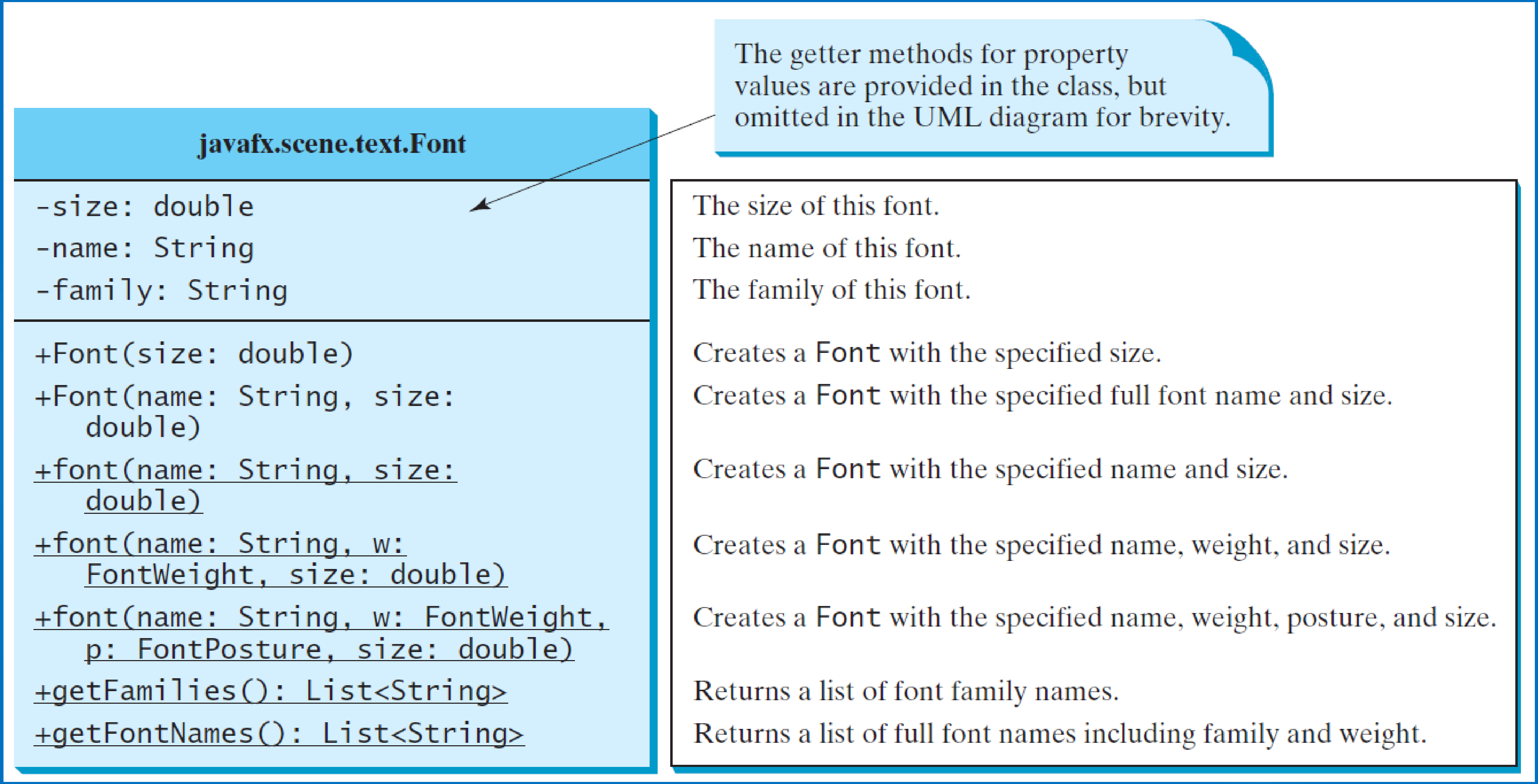


The getter methods for property values are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.paint.Color

`-red: double`	The red value of this Color (between `0.0` and `1.0`).
`-green: double`	The green value of this Color (between `0.0` and `1.0`).
`-blue: double`	The blue value of this Color (between `0.0` and `1.0`).
`-opacity: double`	The opacity of this Color (between `0.0` and `1.0`).
`+Color(r: double, g: double, b: double, opacity: double)`	Creates a `Color` with the specified red, green, blue, and opacity values.
`+brighter(): Color`	Creates a `Color` that is a brighter version of this `Color`.
`+darker(): Color`	Creates a `Color` that is a darker version of this `Color`.
`+color(r: double, g: double, b: double): Color`	Creates an opaque `Color` with the specified red, green, and blue values.
`+color(r: double, g: double, b: double, opacity: double): Color`	Creates a `Color` with the specified red, green, blue, and opacity values.
`+rgb(r: int, g: int, b: int): Color`	Creates a `Color` with the specified red, green, and blue values in the range from `0` to `255`.
`+rgb(r: int, g: int, b: int, opacity: double): Color`	Creates a `Color` with the specified red, green, and blue values in the range from `0` to `255` and a given opacity.

The Font Class


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

javafx.scene.text.Font	
-size: double	The size of this font.
-name: String	The name of this font.
-family: String	The family of this font.
+Font(size: double)	Creates a Font with the specified size.
+Font(name: String, size: double)	Creates a Font with the specified full font name and size.
+font(name: String, size: double)	Creates a Font with the specified name and size.
+font(name: String, w: FontWeight, size: double)	Creates a Font with the specified name, weight, and size.
+font(name: String, w: FontWeight, p: FontPosture, size: double)	Creates a Font with the specified name, weight, posture, and size.
+getFamilies(): List<String>	Returns a list of font family names.
+getFontNames(): List<String>	Returns a list of full font names including family and weight.

Using JavaFX CSS Style

Never has styling a Java UI been easier than with JavaFX and Cascading Style Sheets (CSS). Going from one theme to another, or customizing the look of just one control, can all be done through CSS. [https://openjfx.cn/javadoc/15/javafx.graphics/javafx/scene/doc-files/cssref.html]

```
// Create a scene and place a button in the scene
Button btOK = new Button("OK");
StackPane pane = new StackPane();
pane.getChildren().add(btOK);
btOK.setStyle("-fx-border-color:blue;"
        + "-fx-border-width:3;"
        + "-fx-border-radius:8;"
        + "-fx-font-size:20;"
        + "-fx-font-family:'Times New Roman'");
pane.setStyle("-fx-border-color:red;"
        + "-fx-background-color: lightgray;"
        + "-fx-rotate:45");
```

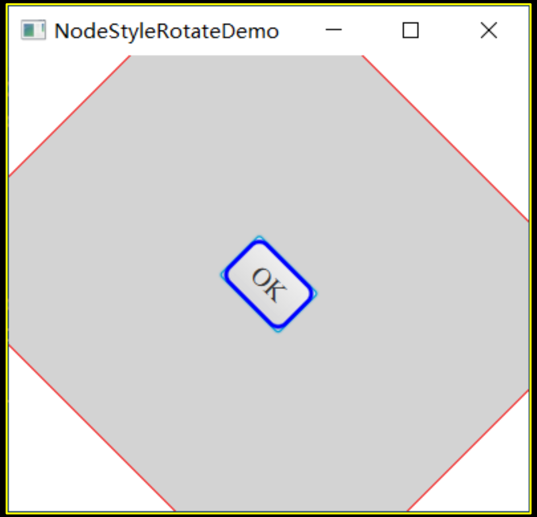
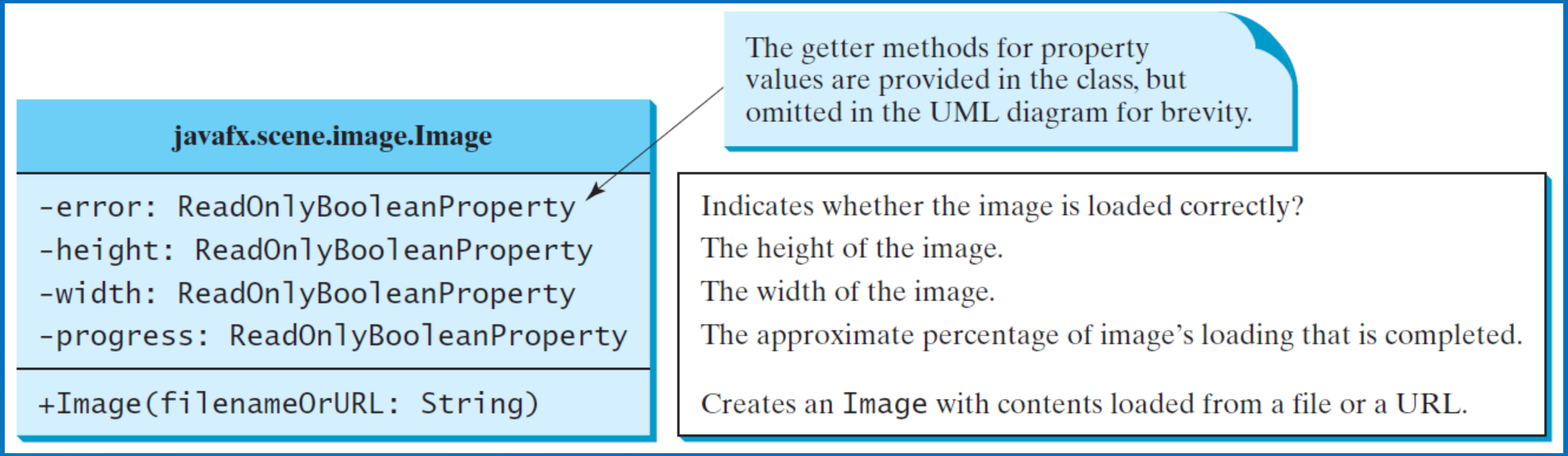
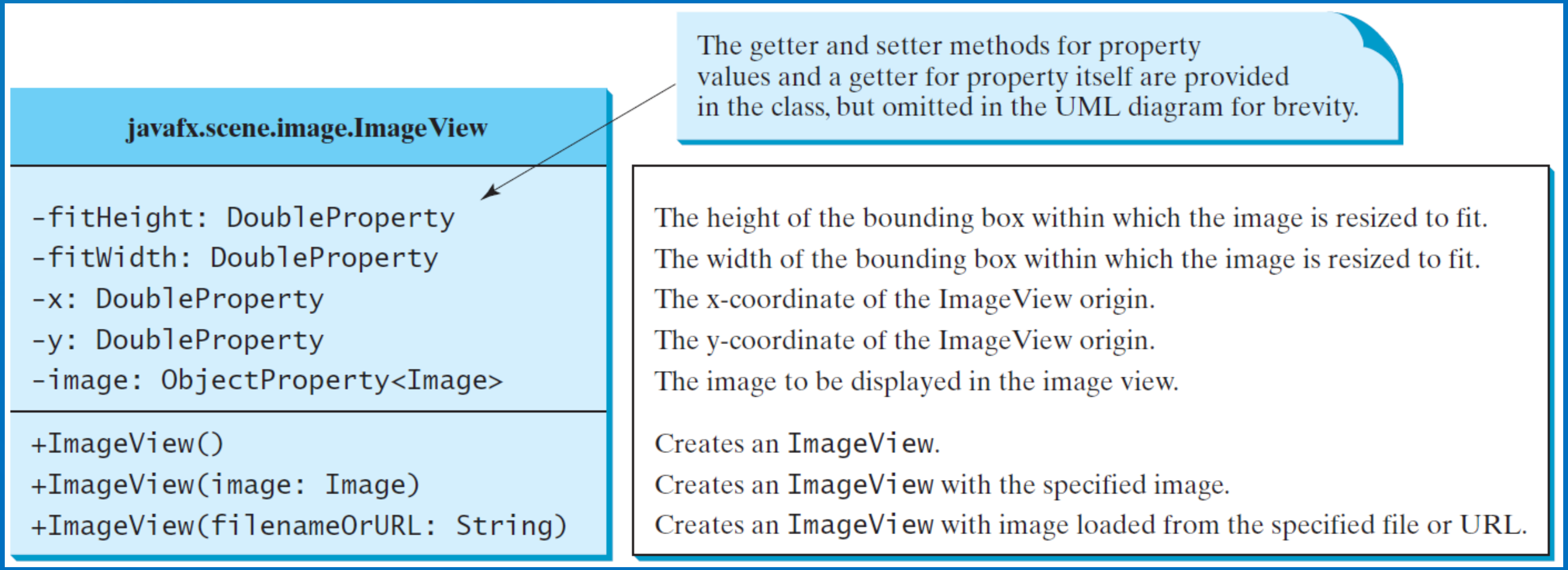


Image and ImageView Classes



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

javafx.scene.image.Image	
-error: ReadOnlyBooleanProperty	Indicates whether the image is loaded correctly?
-height: ReadOnlyBooleanProperty	The height of the image.
-width: ReadOnlyBooleanProperty	The width of the image.
-progress: ReadOnlyBooleanProperty	The approximate percentage of image's loading that is completed.
+Image(filenameOrURL: String)	Creates an Image with contents loaded from a file or a URL.

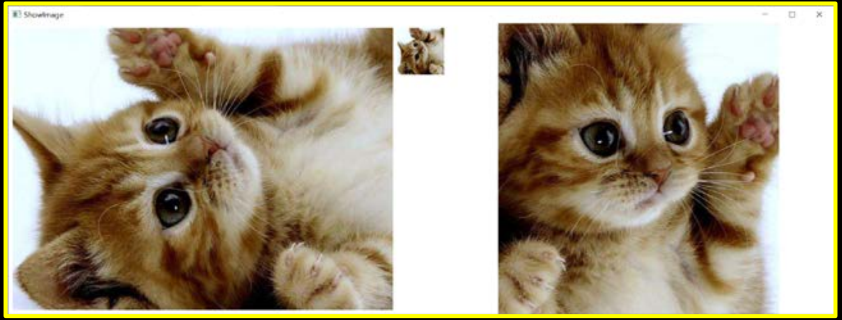


The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.image.ImageView	
-fitHeight: DoubleProperty	The height of the bounding box within which the image is resized to fit.
-fitWidth: DoubleProperty	The width of the bounding box within which the image is resized to fit.
-x: DoubleProperty	The x-coordinate of the ImageView origin.
-y: DoubleProperty	The y-coordinate of the ImageView origin.
-image: ObjectProperty<Image>	The image to be displayed in the image view.
+ImageView()	Creates an ImageView.
+ImageView(image: Image)	Creates an ImageView with the specified image.
+ImageView(filenameOrURL: String)	Creates an ImageView with image loaded from the specified file or URL.

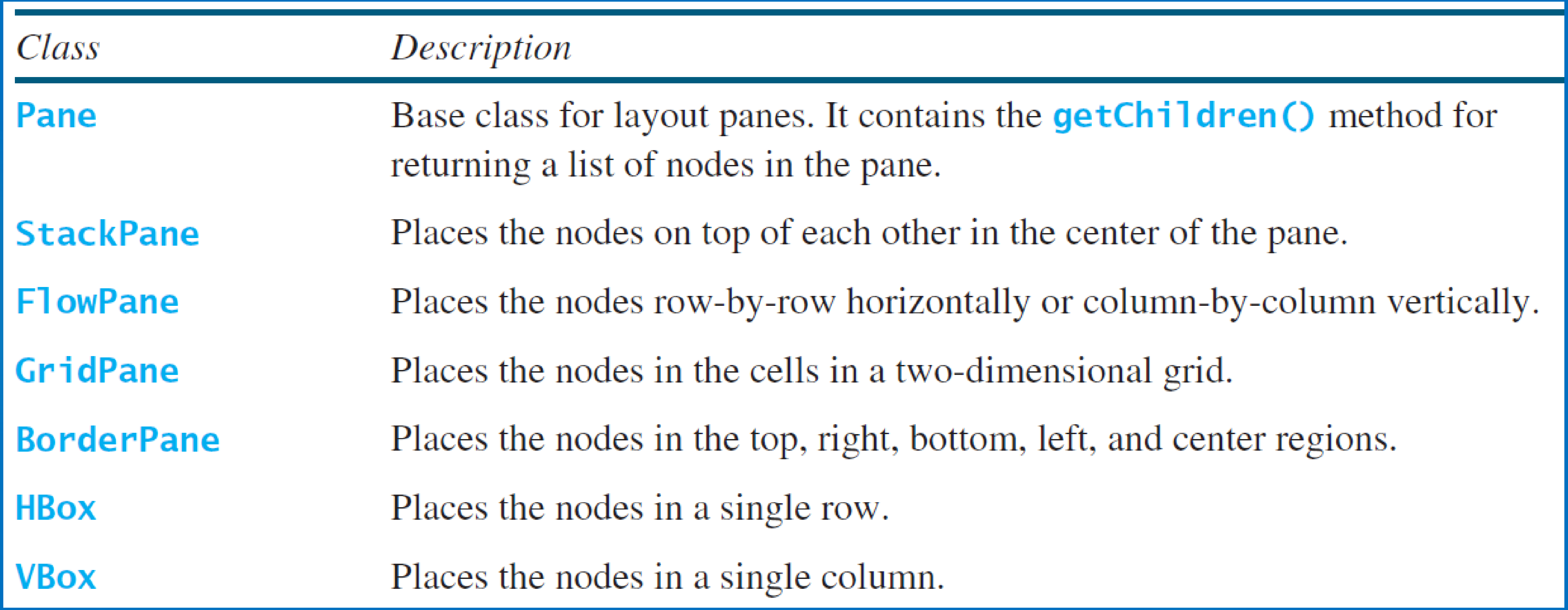
The ImageView Class

```
Pane pane = new HBox(10);
pane.setPadding(new Insets(10, 10, 10, 10));
Image image = new Image("file:1.png");
pane.getChildren().add(new ImageView(image));

ImageView imageView2 = new ImageView(image);
imageView2.setFitHeight(100);
imageView2.setFitWidth(100);
pane.getChildren().add(imageView2);

ImageView imageView3 = new ImageView(image);
imageView3.setRotate(90);
pane.getChildren().add(imageView3);
```



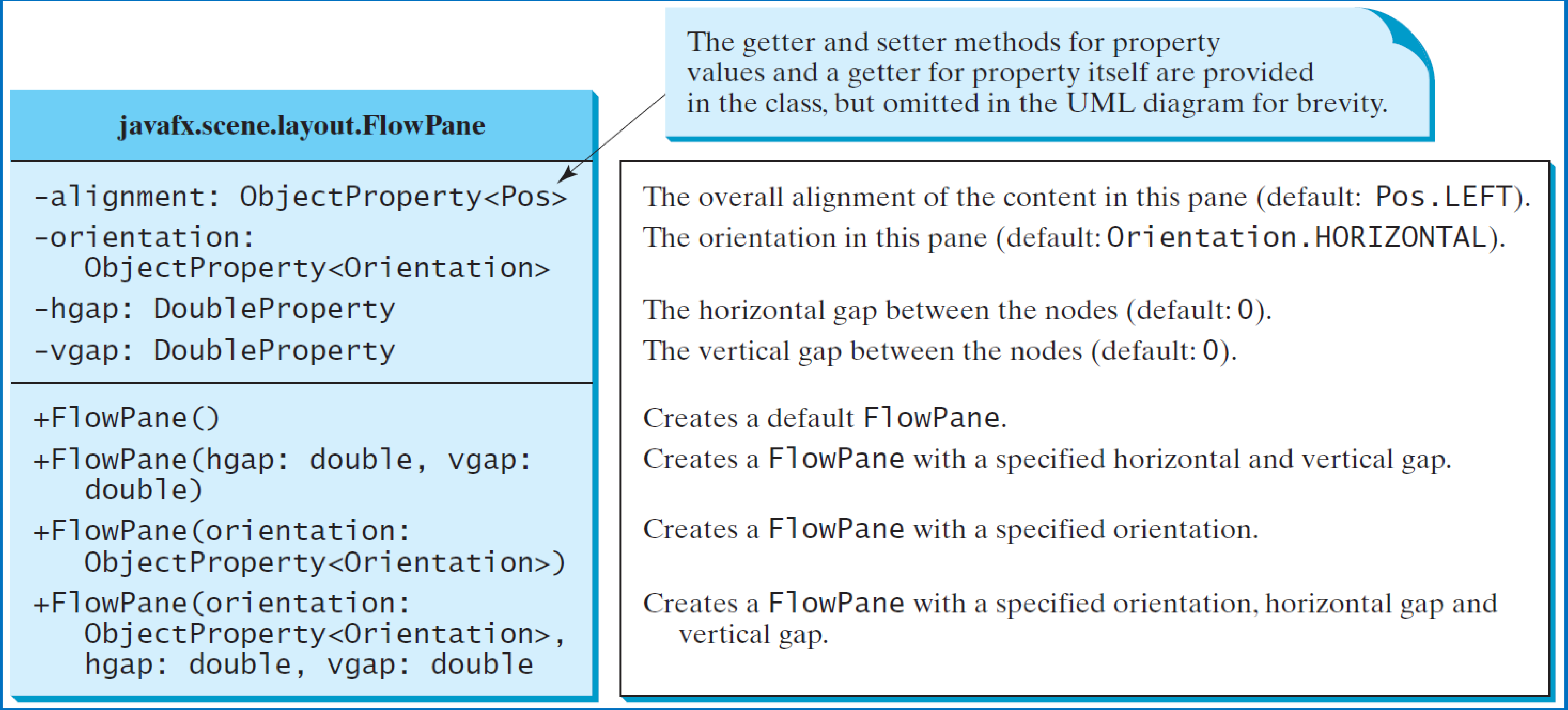
Layout Panes

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



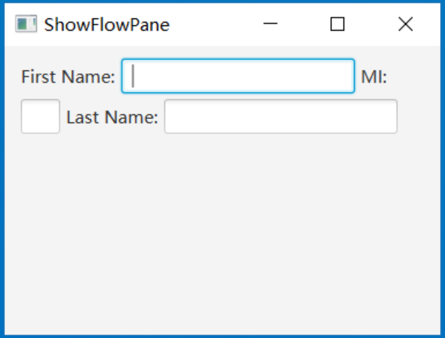
The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

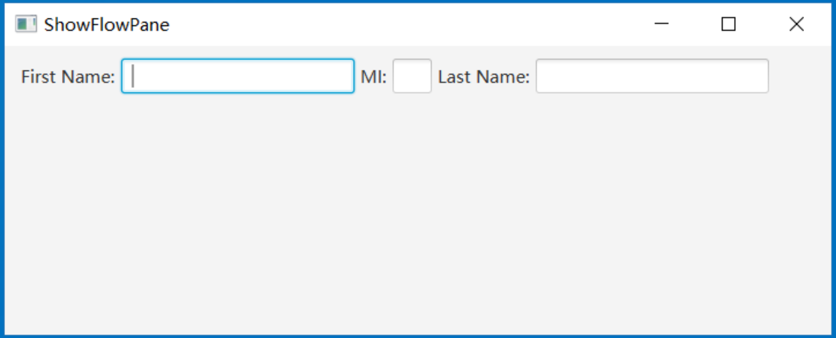
javafx.scene.layout.FlowPane

Member	Description
`-alignment: ObjectProperty<Pos>`	The overall alignment of the content in this pane (default: `Pos.LEFT`).
`-orientation: ObjectProperty<Orientation>`	The orientation in this pane (default: `Orientation.HORIZONTAL`).
`-hgap: DoubleProperty`	The horizontal gap between the nodes (default: `0`).
`-vgap: DoubleProperty`	The vertical gap between the nodes (default: `0`).
`+FlowPane()`	Creates a default `FlowPane`.
`+FlowPane(hgap: double, vgap: double)`	Creates a `FlowPane` with a specified horizontal and vertical gap.
`+FlowPane(orientation: ObjectProperty<Orientation>)`	Creates a `FlowPane` with a specified orientation.
`+FlowPane(orientation: ObjectProperty<Orientation>, hgap: double, vgap: double`	Creates a `FlowPane` with a specified orientation, horizontal gap and vertical gap.

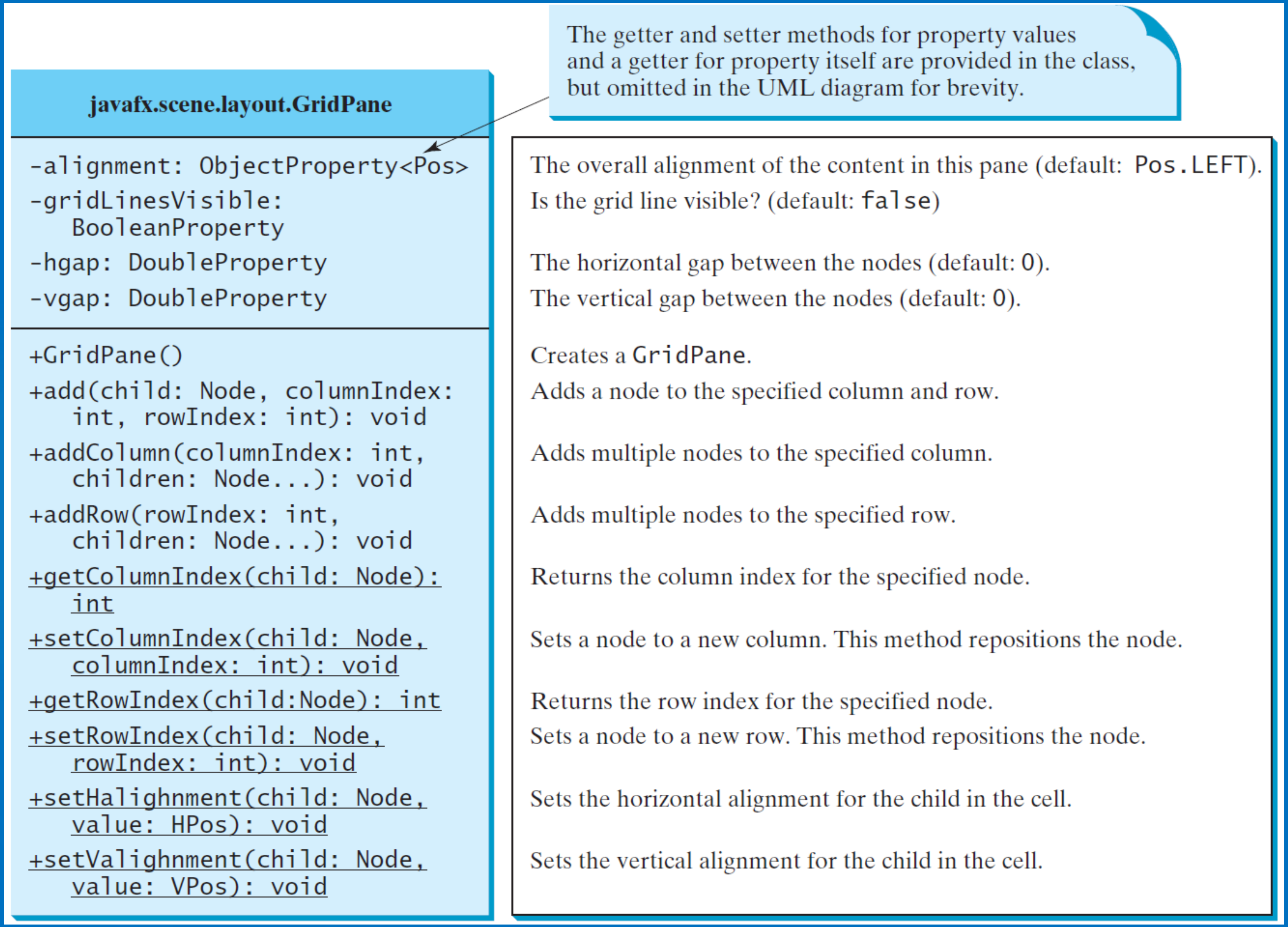
FlowPane

```
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());
```





GridPane



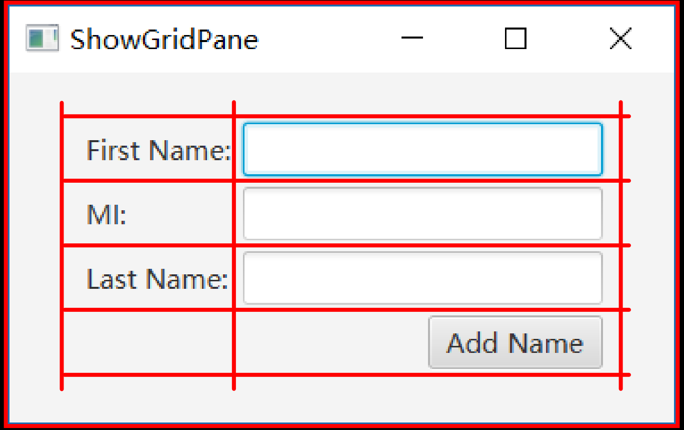
The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.layout.GridPane

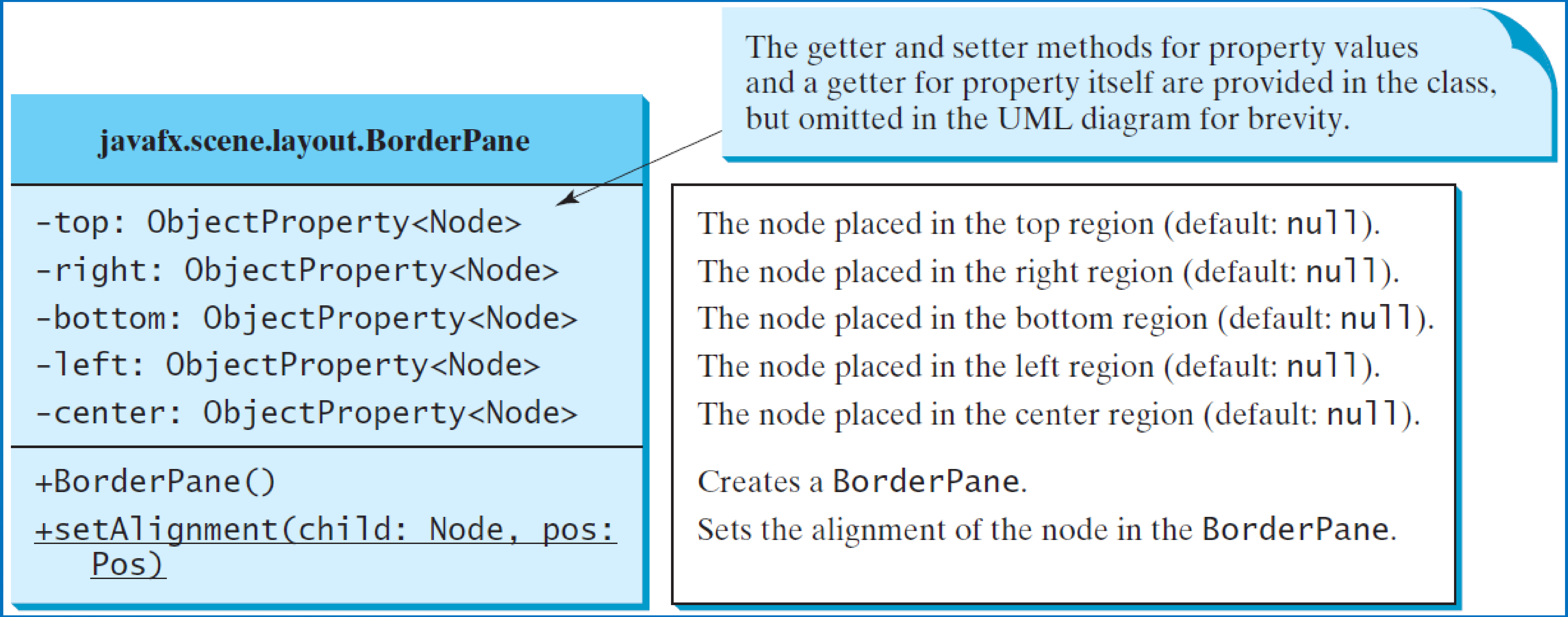
`-alignment: ObjectProperty<Pos>`	The overall alignment of the content in this pane (default: `Pos.LEFT`).
`-gridLinesVisible: BooleanProperty`	Is the grid line visible? (default: `false`)
`-hgap: DoubleProperty`	The horizontal gap between the nodes (default: `0`).
`-vgap: DoubleProperty`	The vertical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

```
GridPane pane = new GridPane();
pane.setAlignment(Pos.CENTER);
pane.setPadding(new Insets(11.5, 12.5, 13.5, 14.5));
pane.setHgap(5.5);pane.setVgap(5.5);
// Place nodes in the pane
pane.add(new Label("First Name:"), 0, 0);
pane.add(new TextField(), 1, 0);
pane.add(new Label("MI:"), 0, 1);
pane.add(new TextField(), 1, 1);
pane.add(new Label("Last Name:"), 0, 2);
pane.add(new TextField(), 1, 2);
Button btAdd = new Button("Add Name");
pane.add(btAdd, 1, 3);
GridPane.setHalignment(btAdd, HPos.RIGHT);
```



BorderPane



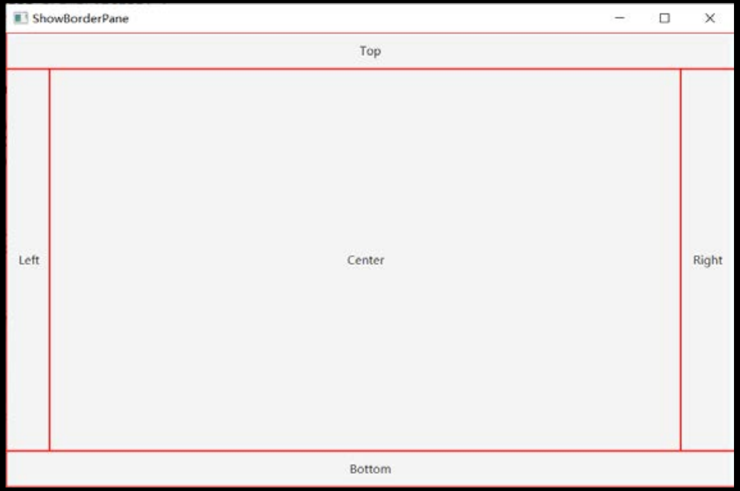
The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.layout.BorderPane

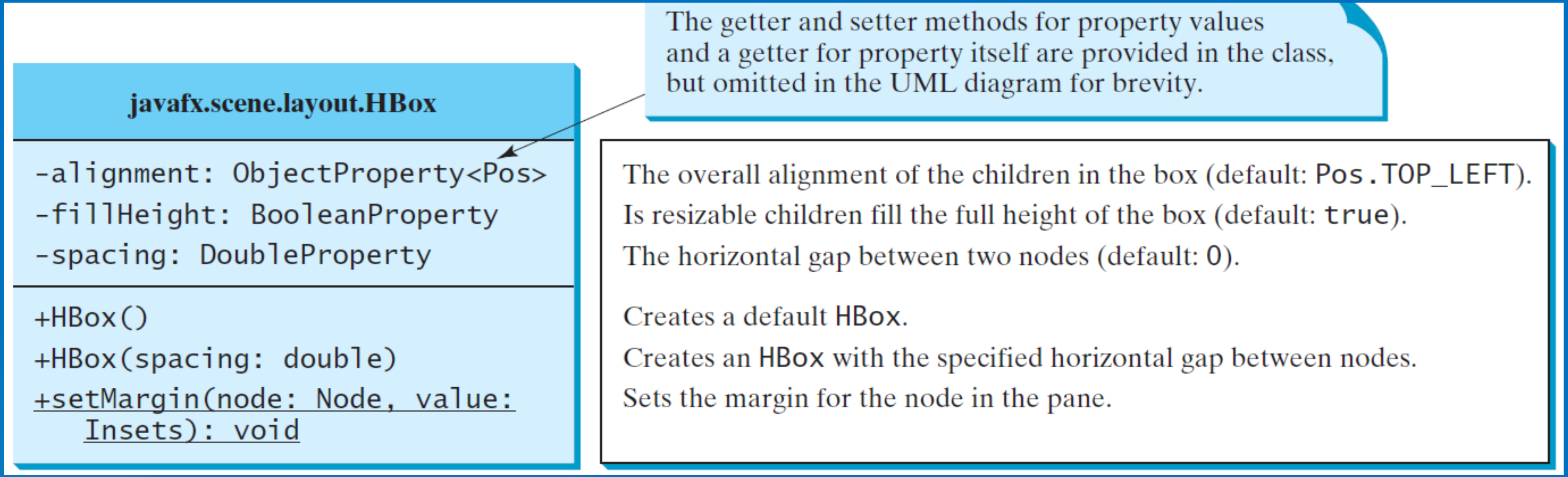
-top: ObjectProperty<Node>	The node placed in the top region (default: null).
-right: ObjectProperty<Node>	The node placed in the right region (default: null).
-bottom: ObjectProperty<Node>	The node placed in the bottom region (default: null).
-left: ObjectProperty<Node>	The node placed in the left region (default: null).
-center: ObjectProperty<Node>	The node placed in the center region (default: null).
+BorderPane()	Creates a BorderPane.
+setAlignment(child: Node, pos: Pos)	Sets the alignment of the node in the BorderPane.

BorderPane

```
// Create a border pane
BorderPane pane = new BorderPane();
// Place nodes in the pane
pane.setTop(new CustomPane("Top"));
pane.setRight(new CustomPane("Right"));
pane.setBottom(new CustomPane("Bottom"));
pane.setLeft(new CustomPane("Left"));
pane.setCenter(new CustomPane("Center"));
```



HBox and VBox



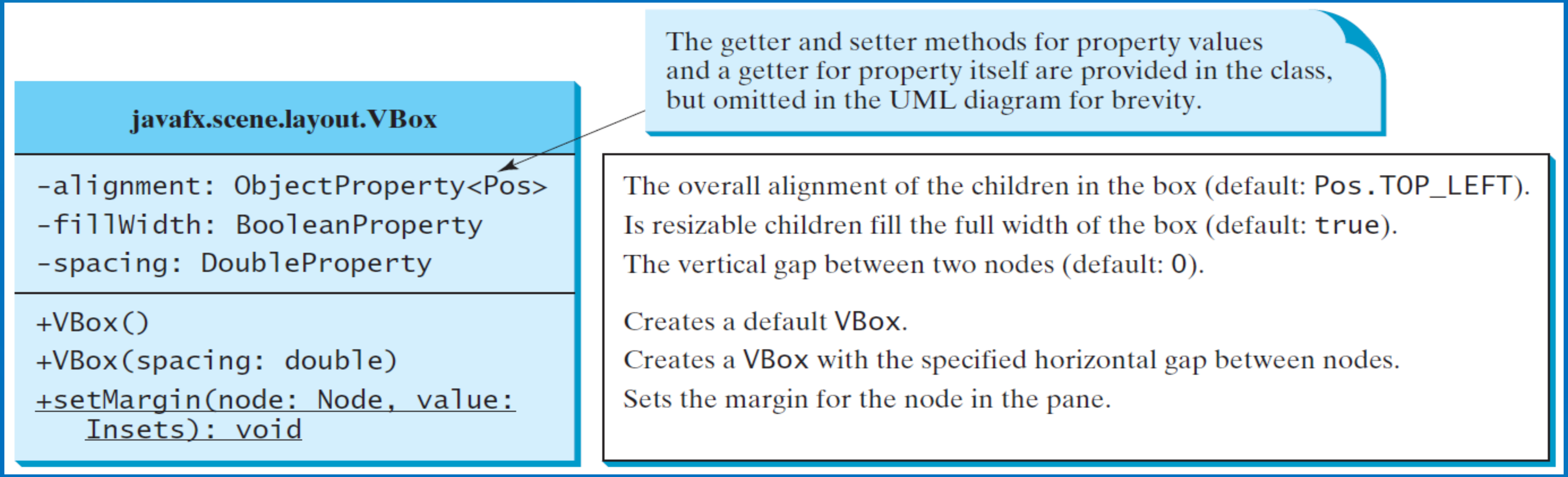
The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.layout.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.



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.

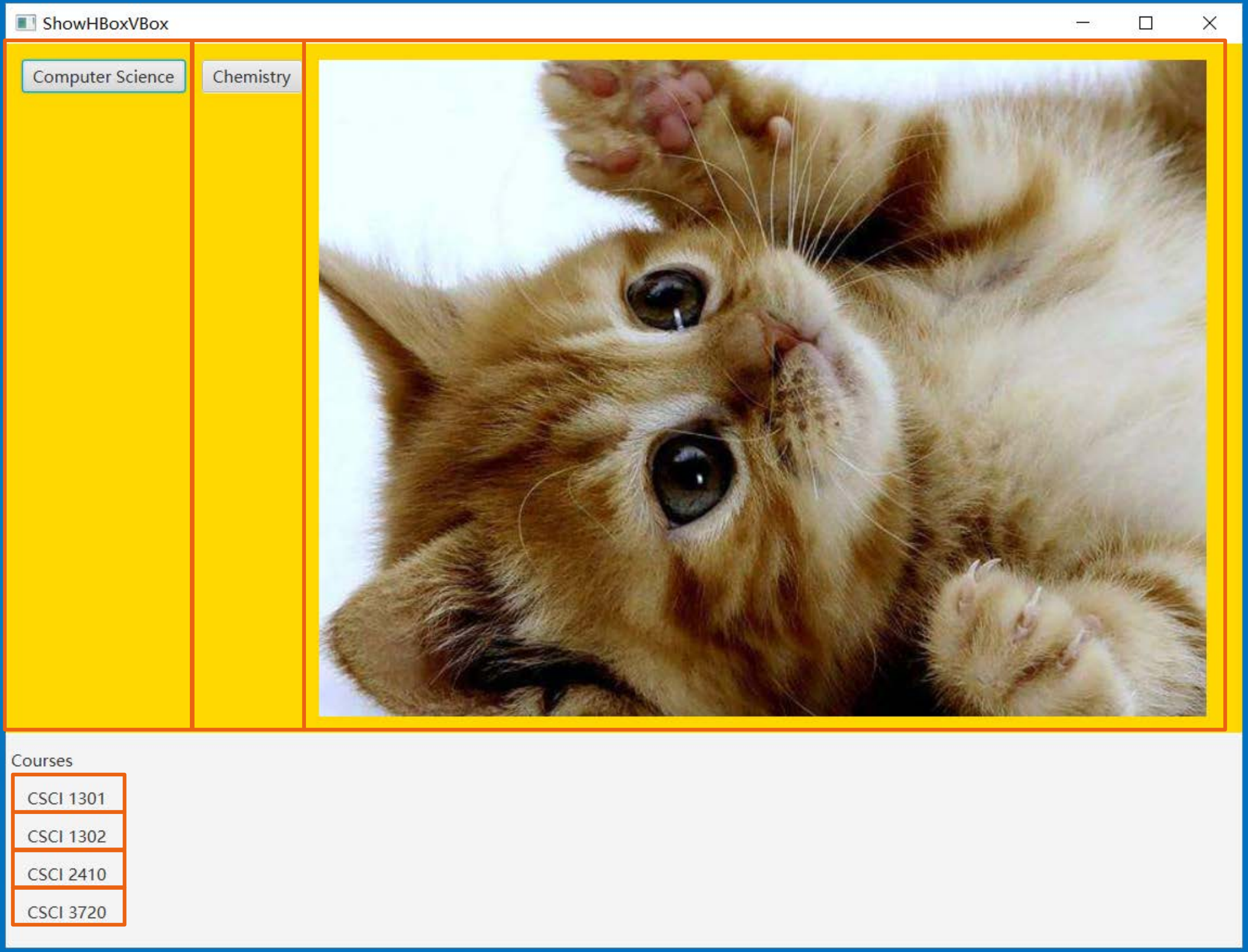
javafx.scene.layout.VBox

```
-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.

HBox and VBox

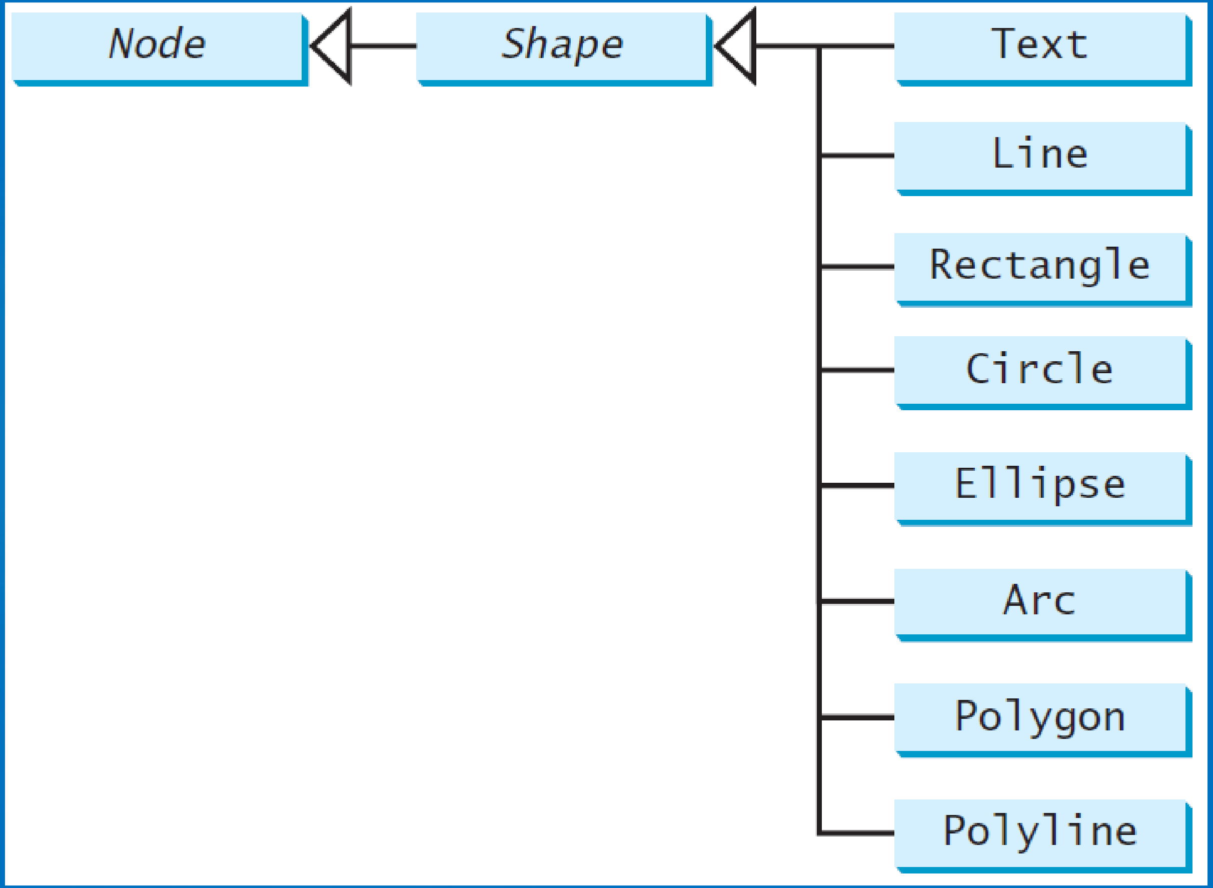


HBox

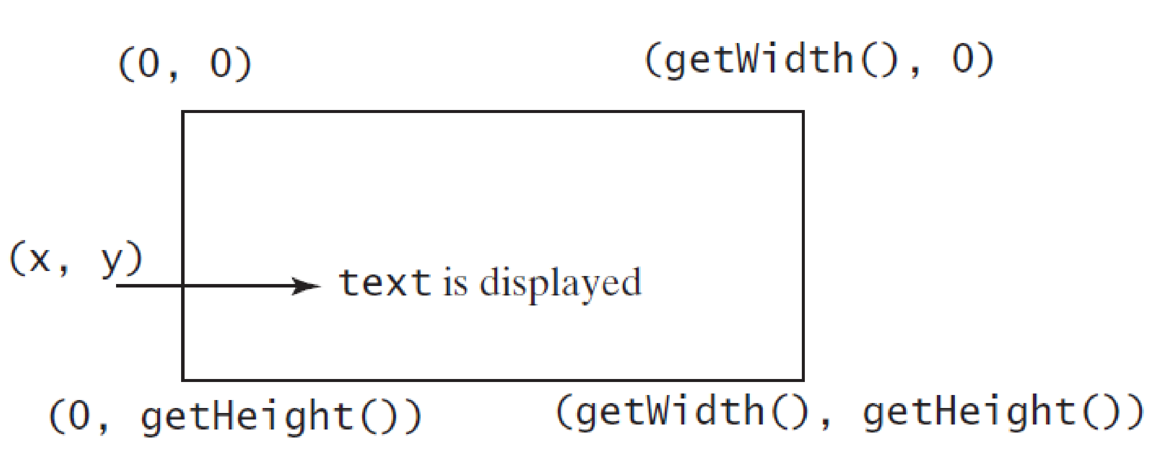
VBox

Shapes

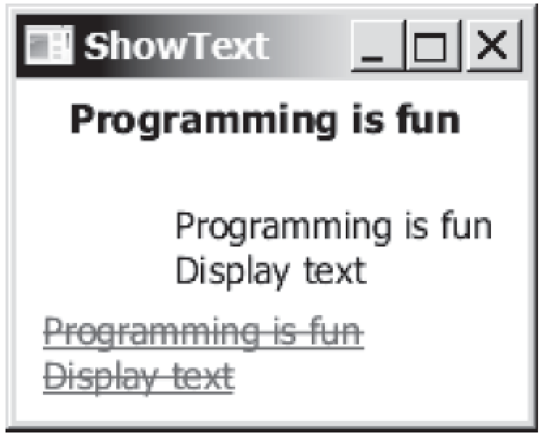
JavaFX provides many shape classes for drawing texts, lines, circles, rectangles, ellipses, arcs, polygons, and polylines.



Text Example



(a) Text(x, y, text)



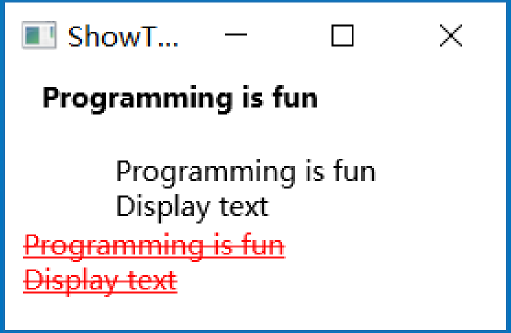
ShowText

Programming is fun

Programming is fun
Display text

~~Programming is fun~~
~~Display text~~

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



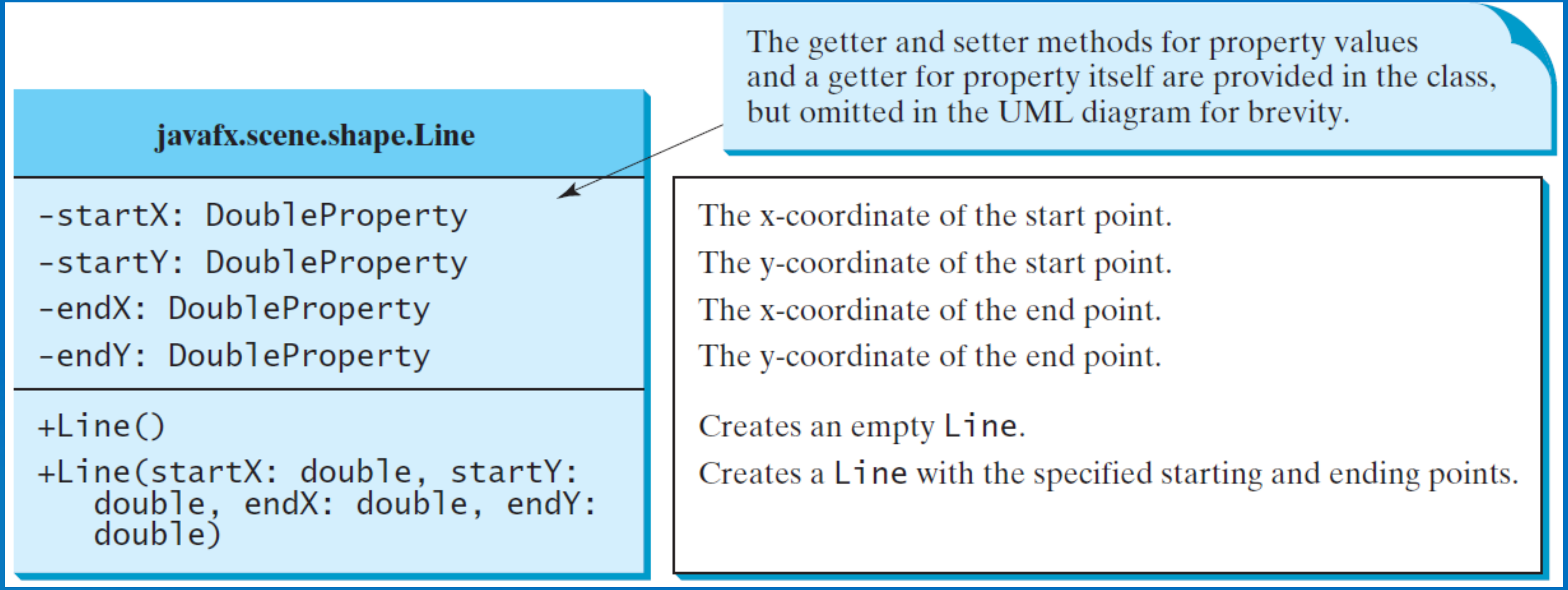
ShowT...

Programming is fun

Programming is fun
Display text

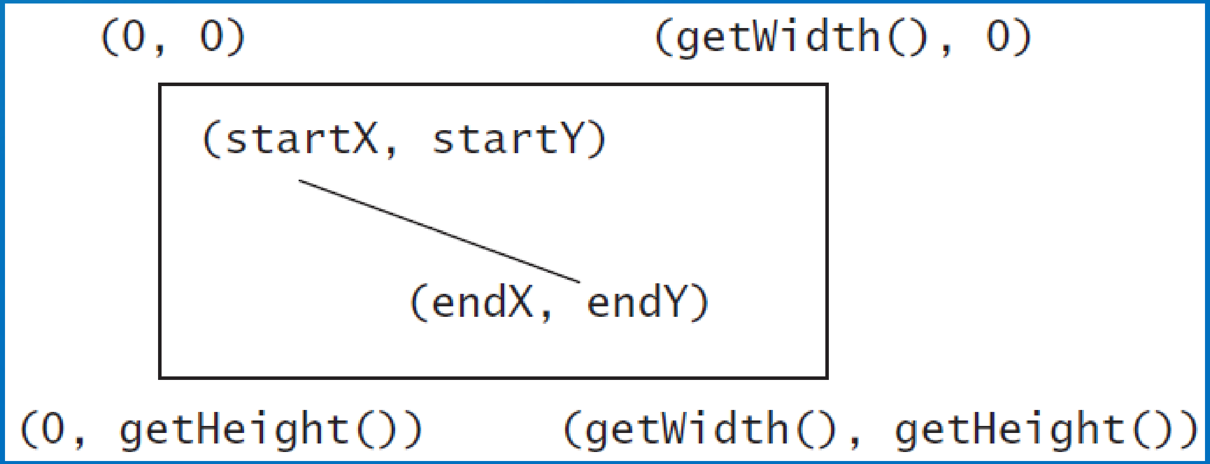
~~Programming is fun~~
~~Display text~~

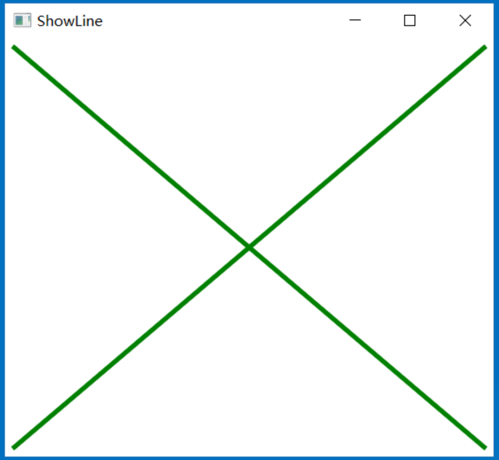
Line



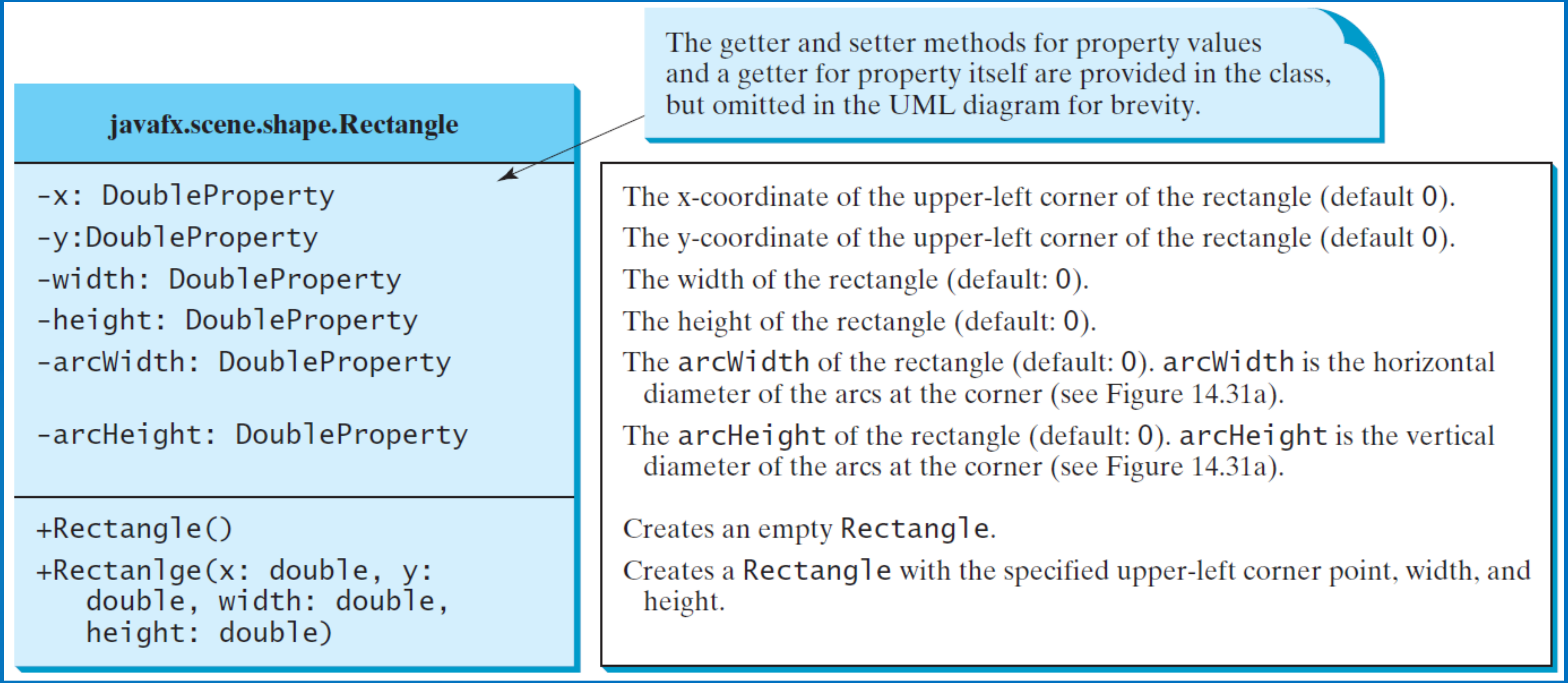
The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.shape.Line	
-startX: DoubleProperty	The x-coordinate of the start point.
-startY: DoubleProperty	The y-coordinate of the start point.
-endX: DoubleProperty	The x-coordinate of the end point.
-endY: DoubleProperty	The y-coordinate of the end point.
+Line()	Creates an empty Line.
+Line(startX: double, startY: double, endX: double, endY: double)	Creates a Line with the specified starting and ending points.





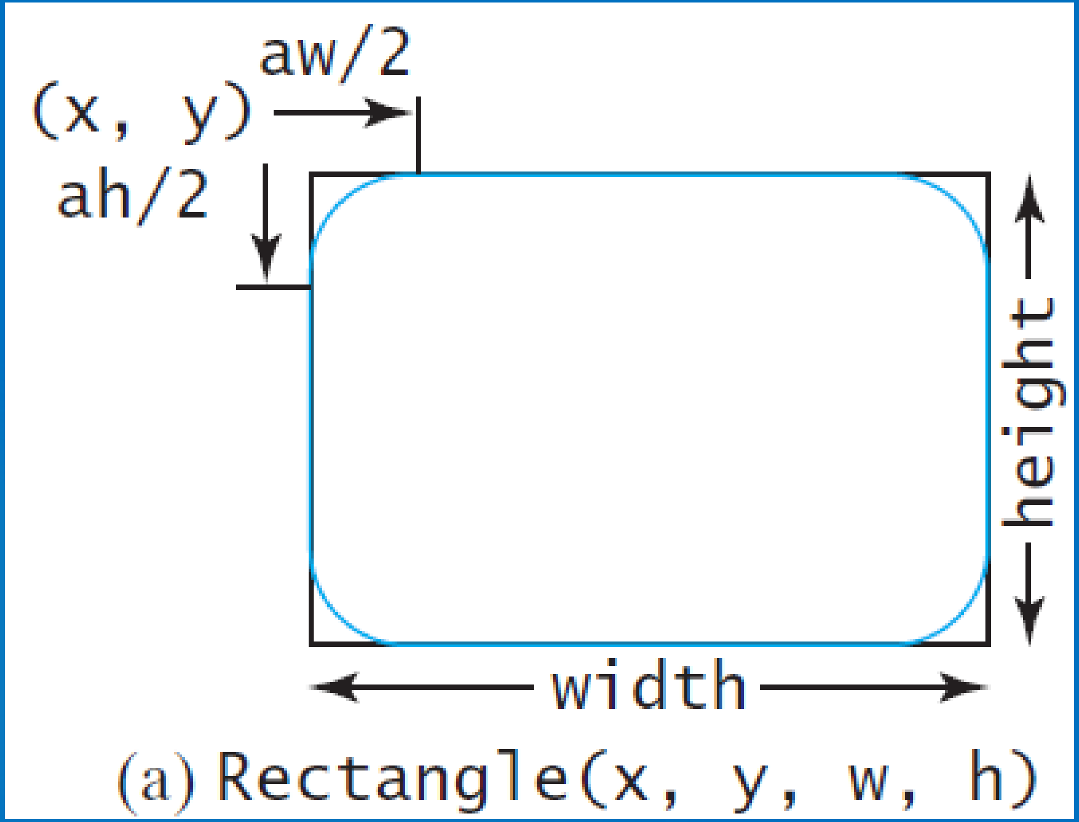
Rectangle

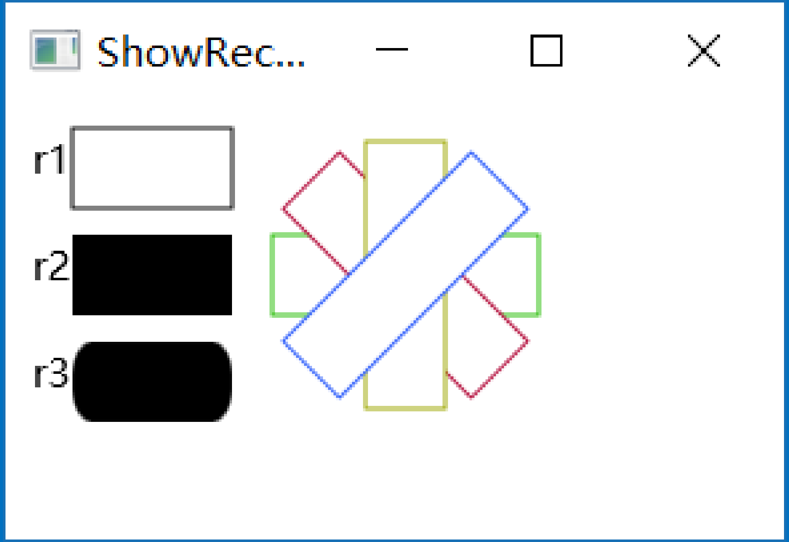


The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

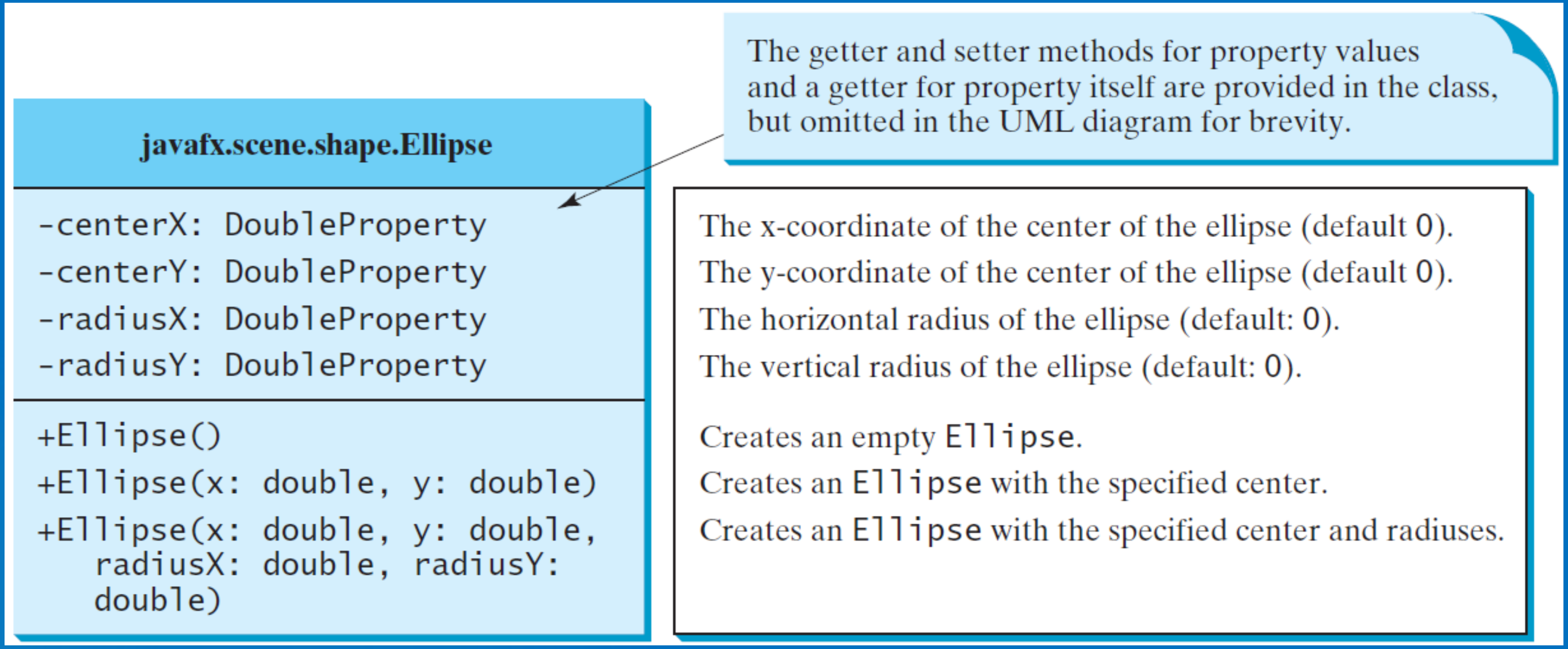
javafx.scene.shape.Rectangle	
`-x: DoubleProperty`	The x-coordinate of the upper-left corner of the rectangle (default `0`).
`-y:DoubleProperty`	The y-coordinate of the upper-left corner of the rectangle (default `0`).
`-width: DoubleProperty`	The width of the rectangle (default: `0`).
`-height: DoubleProperty`	The height of the rectangle (default: `0`).
`-arcWidth: DoubleProperty`	The `arcWidth` of the rectangle (default: `0`). `arcWidth` is the horizontal diameter of the arcs at the corner (see Figure 14.31a).
`-arcHeight: DoubleProperty`	The `arcHeight` of the rectangle (default: `0`). `arcHeight` is the vertical diameter of the arcs at the corner (see Figure 14.31a).
`+Rectangle()`	Creates an empty `Rectangle`.
`+Rectanlge(x: double, y: double, width: double, height: double)`	Creates a `Rectangle` with the specified upper-left corner point, width, and height.

Rectangle





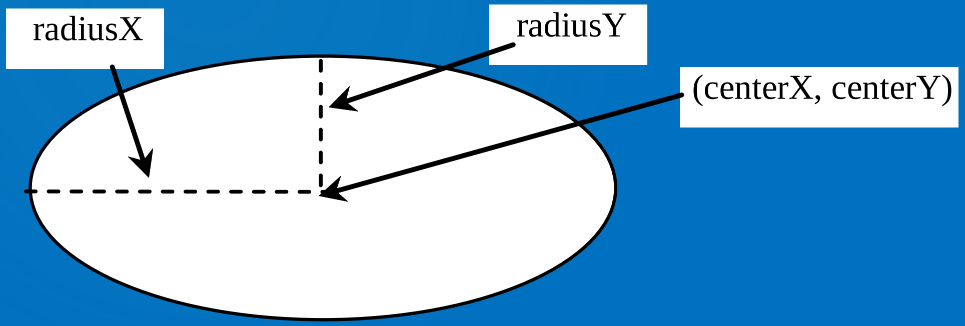
Ellipse

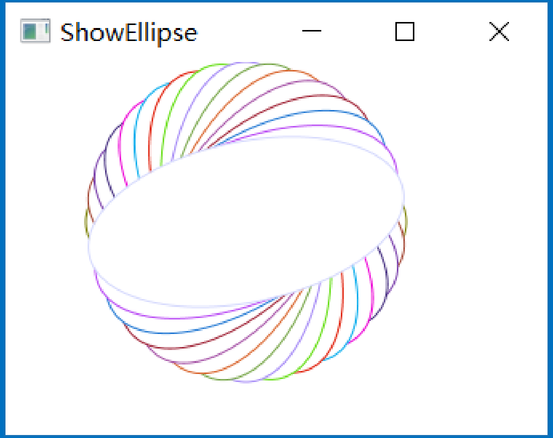


The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

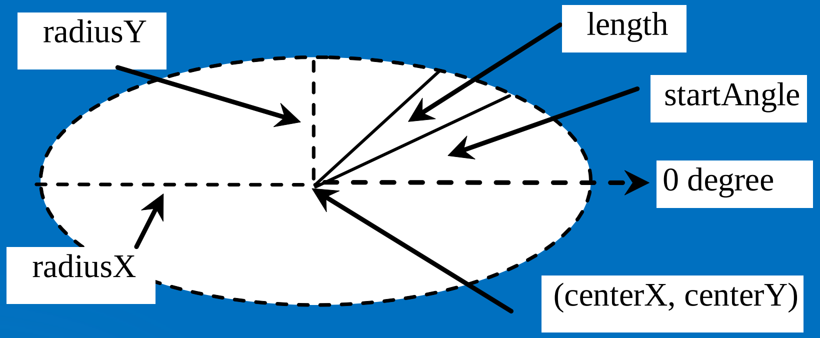
javafx.scene.shape.Ellipse

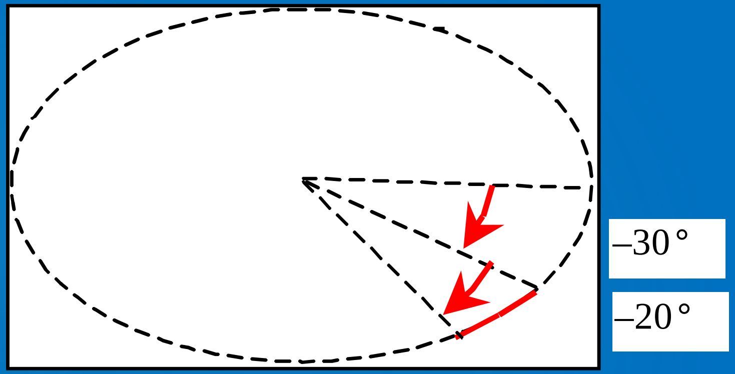
`-centerX: DoubleProperty`	The x-coordinate of the center of the ellipse (default 0).
`-centerY: DoubleProperty`	The y-coordinate of the center of the ellipse (default 0).
`-radiusX: DoubleProperty`	The horizontal radius of the ellipse (default: 0).
`-radiusY: DoubleProperty`	The vertical radius of the ellipse (default: 0).
`+Ellipse()`	Creates an empty `Ellipse`.
`+Ellipse(x: double, y: double)`	Creates an `Ellipse` with the specified center.
`+Ellipse(x: double, y: double, radiusX: double, radiusY: double)`	Creates an `Ellipse` with the specified center and radiuses.



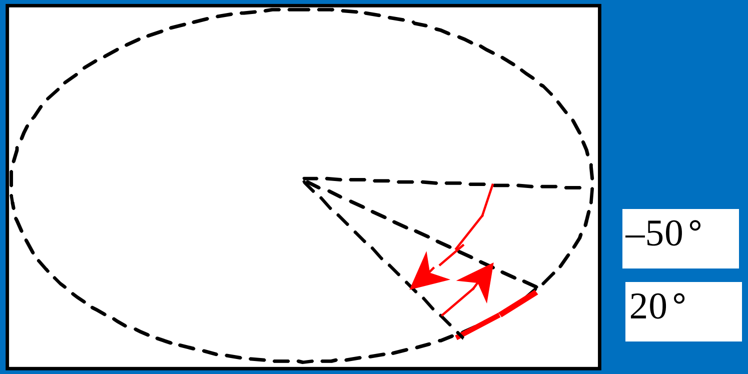


Arc

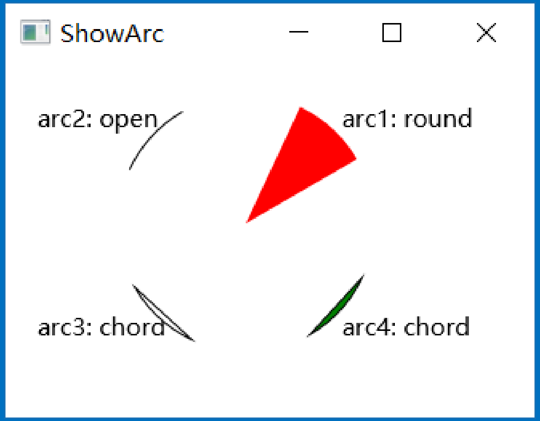




(a) Negative starting angle –30° and negative spanning angle –20°



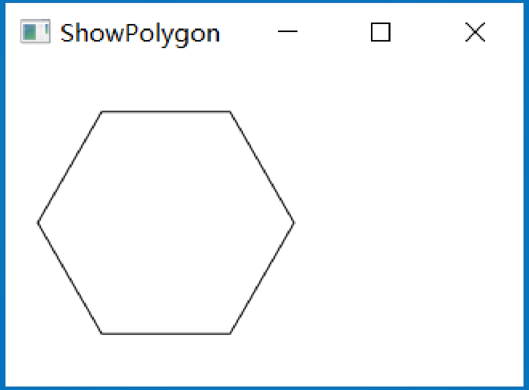
(b) Negative starting angle –50° and positive spanning angle 20°



Polygon

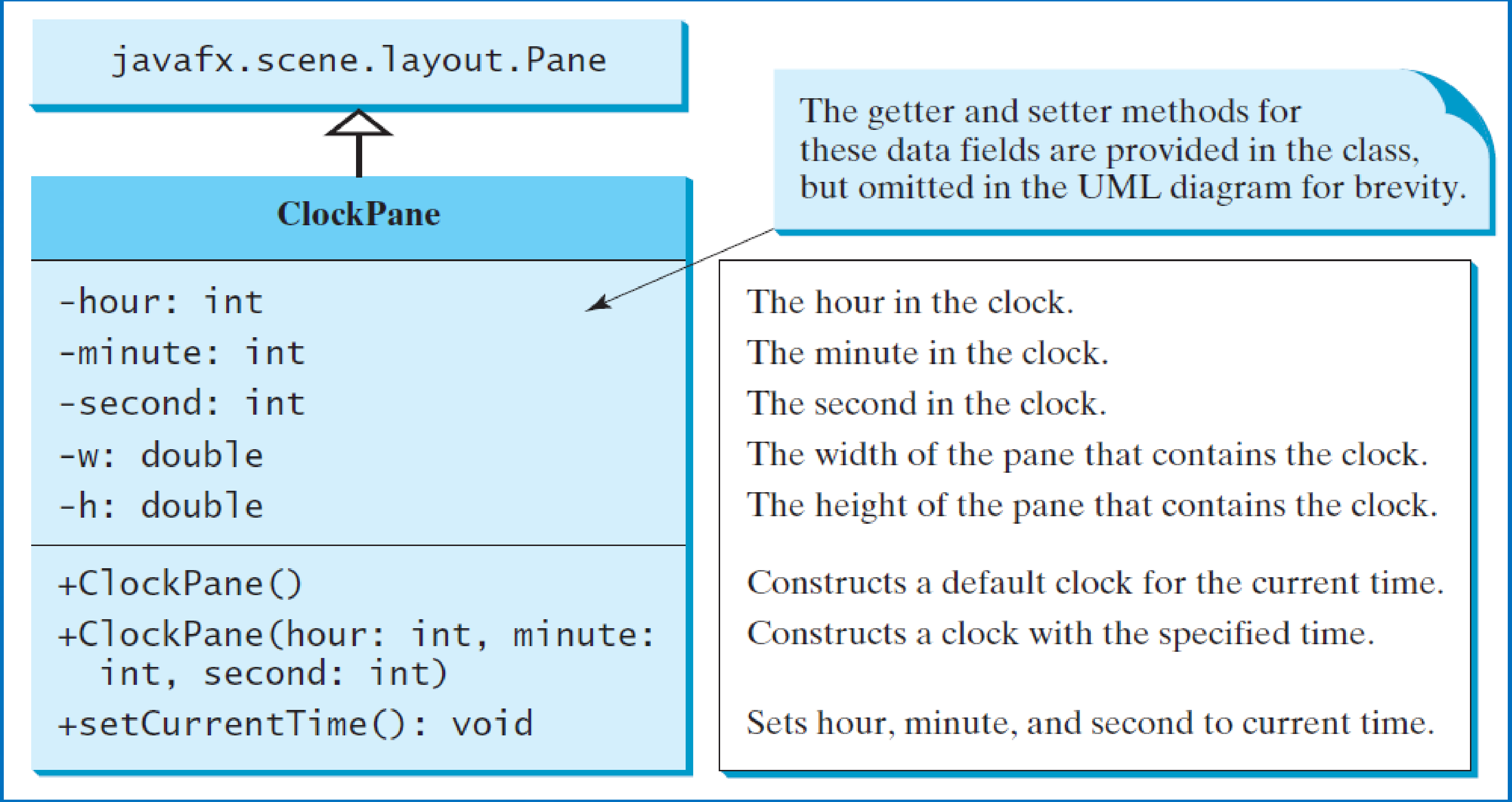
The `getter` and `setter` methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.shape.Polygon	
`+Polygon()`	Creates an empty polygon.
`+Polygon(double... points)`	Creates a polygon with the given points.
`+getPoints(): ObservableList<Double>`	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.

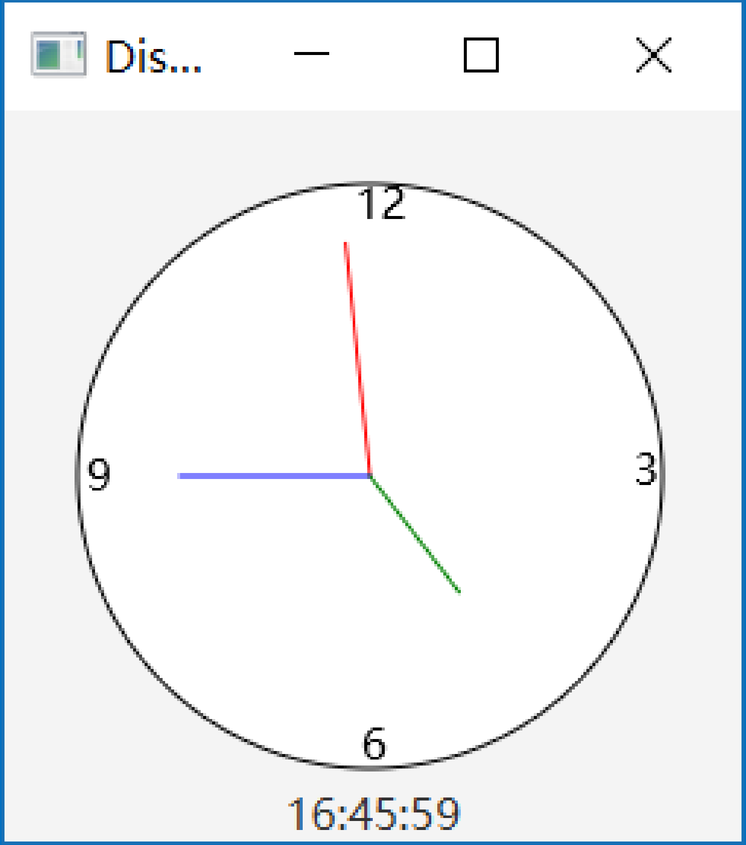


The getter and setter methods for these data fields are provided in the class, but omitted in the UML diagram for brevity.

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.


Dis...
12
3
6
9
16:45:59