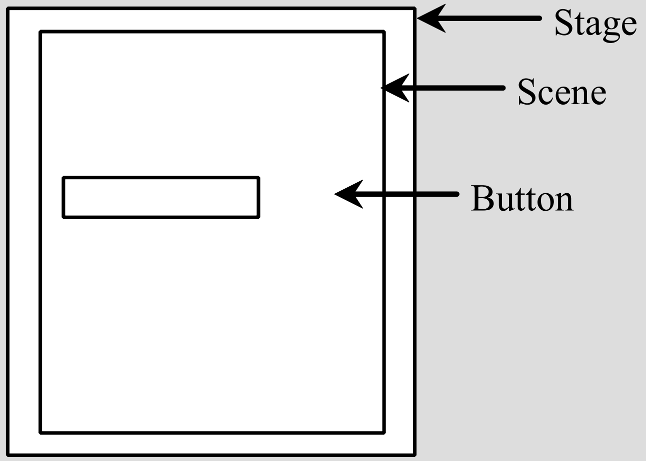
COIS 2240 Lecture 10

JavaFX for Eclipse Users

If you are using Eclipse, you have to install e(fx)clipse. Follow the instructions here: https://www.eclipse.org/efxclipse/install.html

Basic Structure of JavaFX

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



MyJavaFX Run

MultipleStageDemo Run

Example

```
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
  }

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

Example-Multiple Stages

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

public class MultipleStageDemo 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
    Scene scene = new Scene(new Button("OK"), 200, 250);
    primaryStage.setTitle("MyJavaFX"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage

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

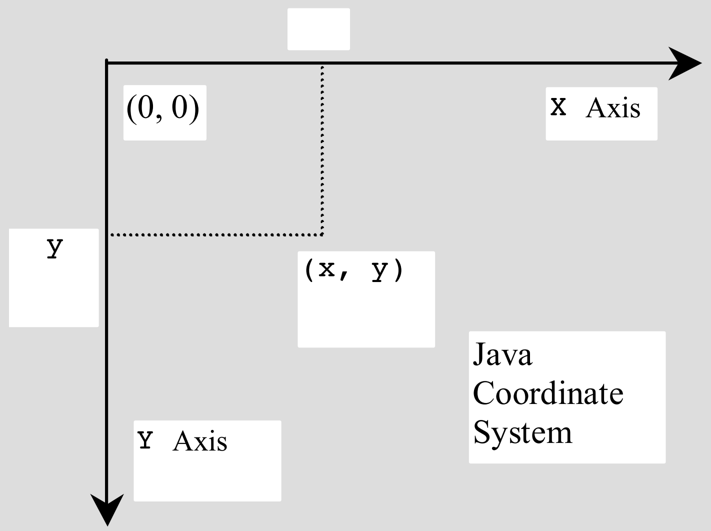
Panes

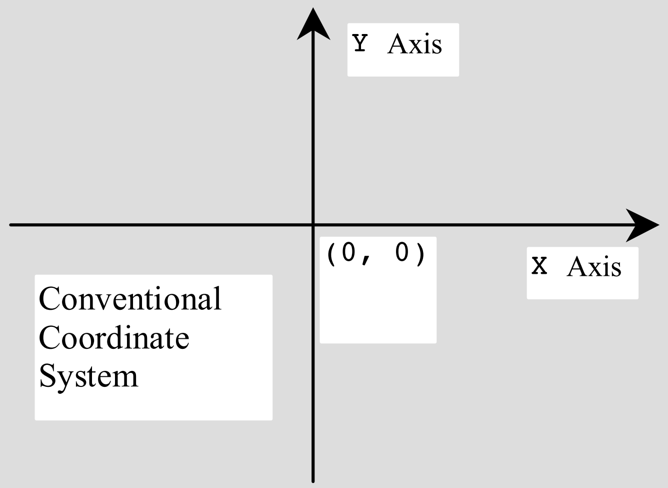
```
package JavaFXExamples;

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"));
    Scene scene = new Scene(pane, 200, 50);
    primaryStage.setTitle("Button in a pane"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }

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

ButtonInPane

Display a Shape

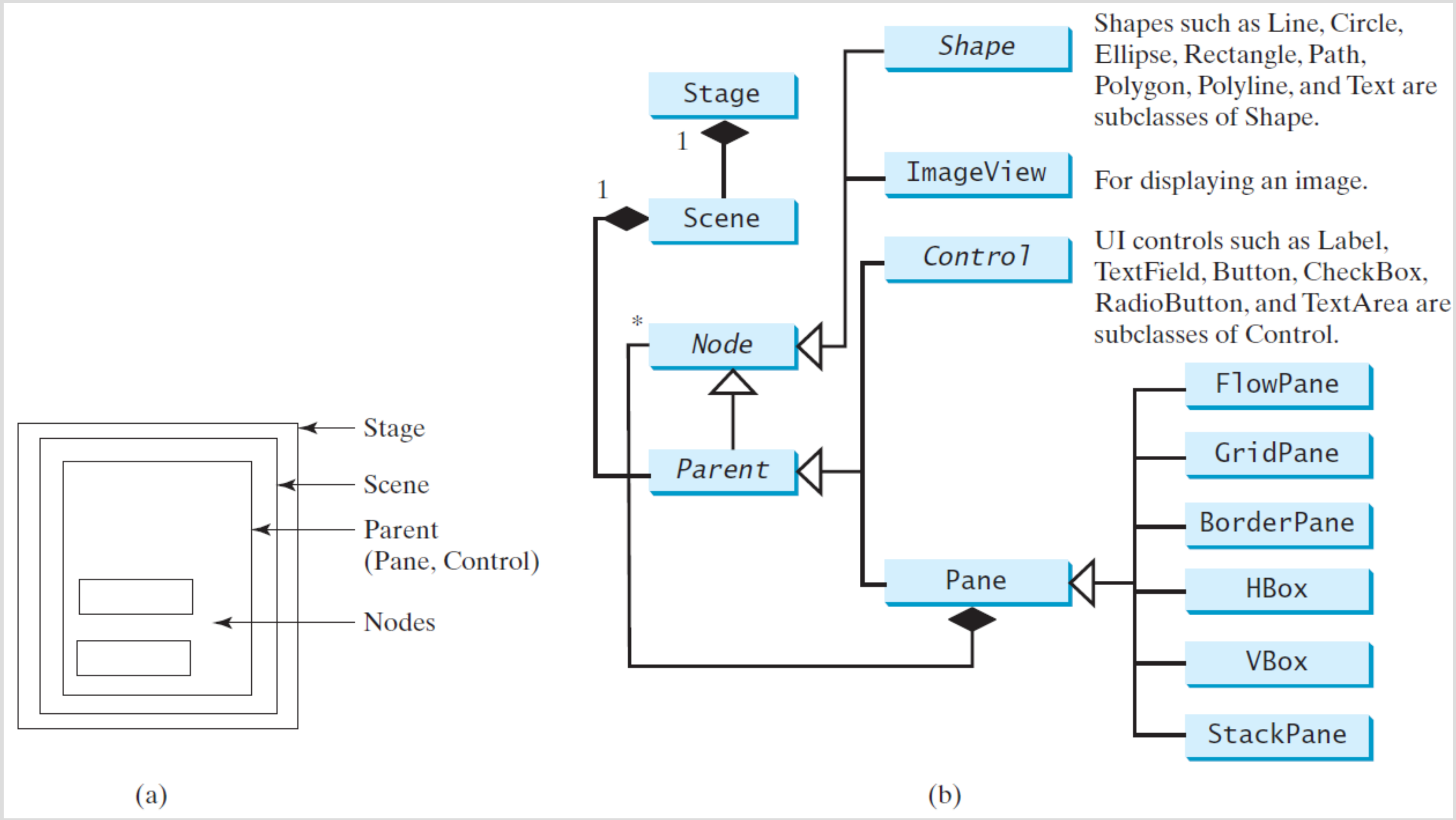
This example displays a circle in the center of the pane.





ShowCircle Run

Panes, UI Controls, and Shapes


Stage
Scene
Parent
(Pane, Control)
Nodes
(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)

Circle in a Pane

```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(100);
    circle.setCenterY(100);
    circle.setRadius(50);
    circle.setStroke(Color.BLACK);
    circle.setFill(Color.WHITE);

    // 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
  }

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

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*.

ShowCircleCentered Run

Bind Circle with Pane

```
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 ShowCircleCentered 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 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

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

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

Binding Property: getter, setter, and property getter

```java
public class SomeClassName {

  private PropertyType x;

  /** Value getter method */
  public propertyValueType getX() { ... }

  /** Value setter method */
  public void setX(propertyValueType value) { ... }

  /** Property getter method */
  public PropertyType
    xProperty() { ... }
}
```

(a) x is a binding property

```java
public class Circle {

  private DoubleProperty centerX;

  /** Value getter method */
  public double getCenterX() { ... }

  /** Value setter method */
  public void setCenterX(double value) { ... }

  /** Property getter method */
  public DoubleProperty centerXProperty() { ... }
}
```

(b) centerX is binding property

Common Properties and Methods for Nodes

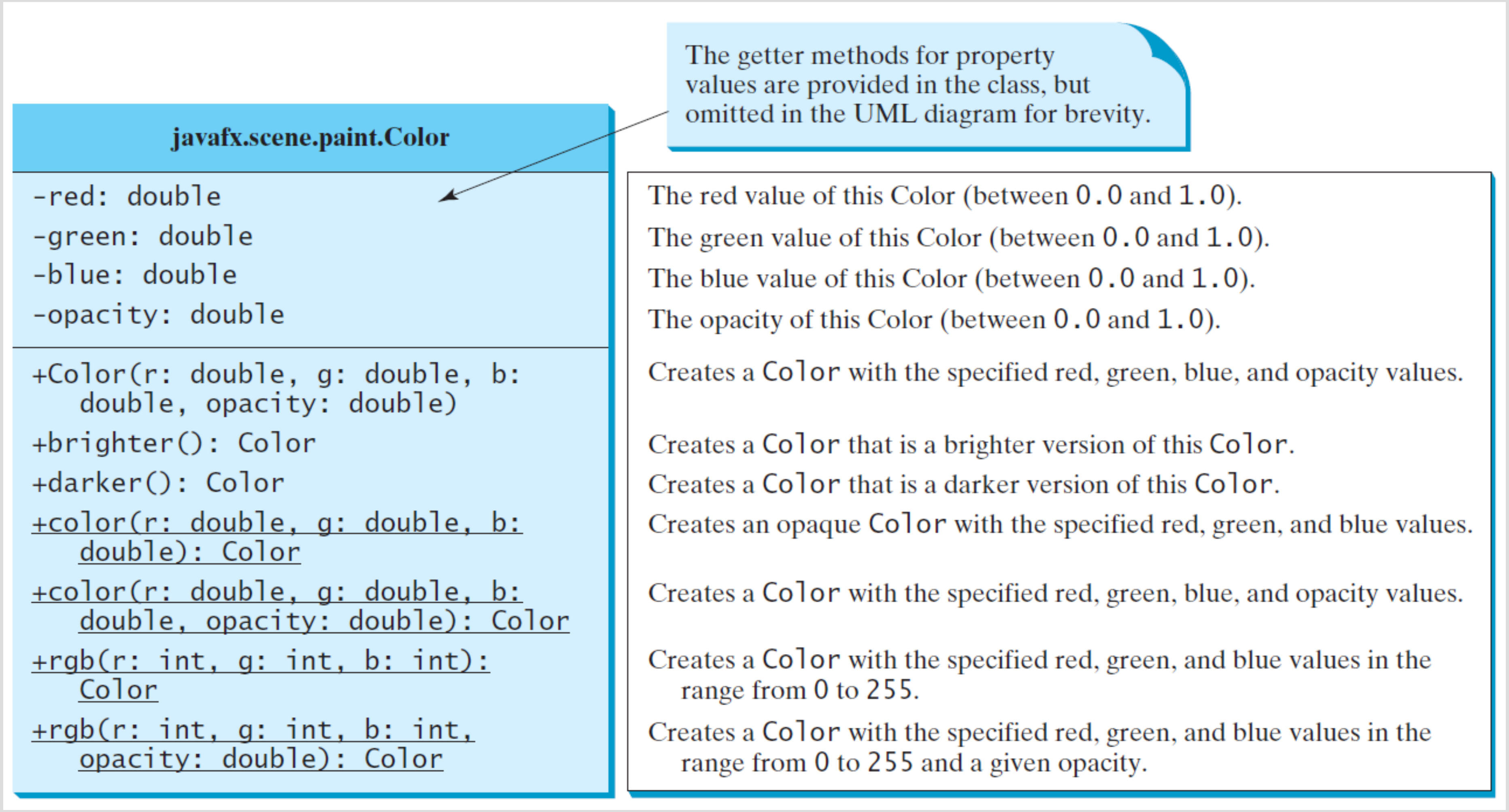
- style: set a JavaFX CSS style
- rotate: Rotate a node
- You can find them in: http://docs.oracle.com/javafx/2/api/javafx/scene/doc-files/

NodeStyleRotateDemo Run

Properties Example

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

public class NodeStyleRotateDemo 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();
    Button btOK = new Button("OK");
    btOK.setStyle("-fx-border-color: blue;");
    pane.getChildren().add(btOK);

    pane.setRotate(45);
    pane.setStyle(
      "-fx-border-color: red; -fx-background-color: lightgray;");

    Scene scene = new Scene(pane, 200, 250);
    primaryStage.setTitle("NodeStyleRotateDemo"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }

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

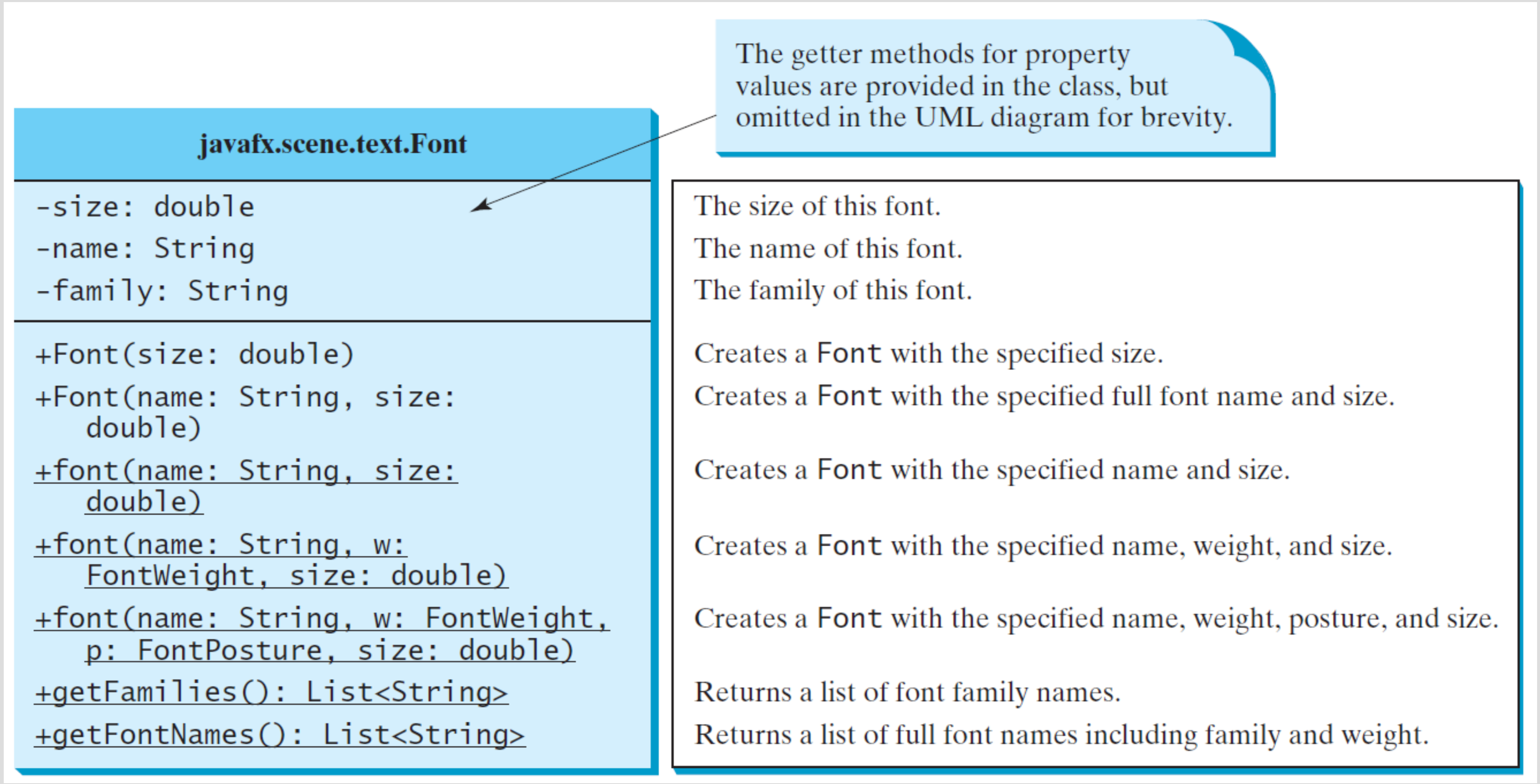
The Color Class



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.
<u>+color(r: double, g: double, b: double): Color</u>	Creates an opaque Color with the specified red, green, and blue values.
<u>+color(r: double, g: double, b: double, opacity: double): Color</u>	Creates a Color with the specified red, green, blue, and opacity values.
<u>+rgb(r: int, g: int, b: int): Color</u>	Creates a Color with the specified red, green, and blue values in the range from 0 to 255.
<u>+rgb(r: int, g: int, b: int, opacity: double): Color</u>	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

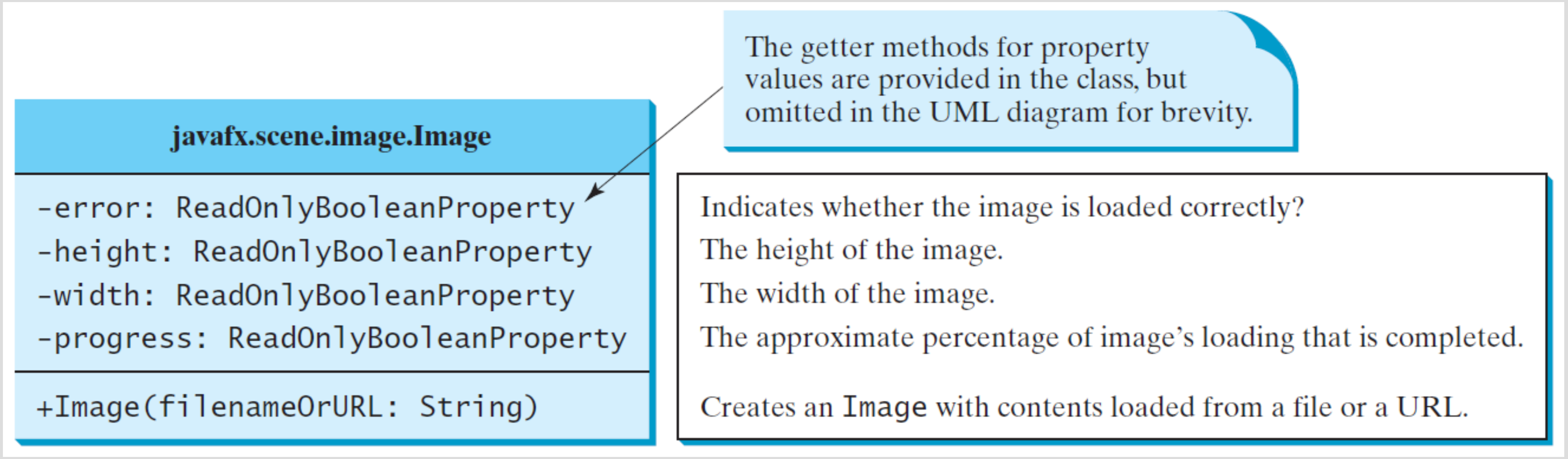
`-size: double`	The size of this font.
`-name: String`	The name of this font.
`-family: String`	The family of this font.
`+Font(size: double)`	Creates a `Font` with the specified size.
`+Font(name: String, size: double)`	Creates a `Font` with the specified full font name and size.
`+font(name: String, 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.

FontDemo Run

Font Example

```java
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
  }
}
```

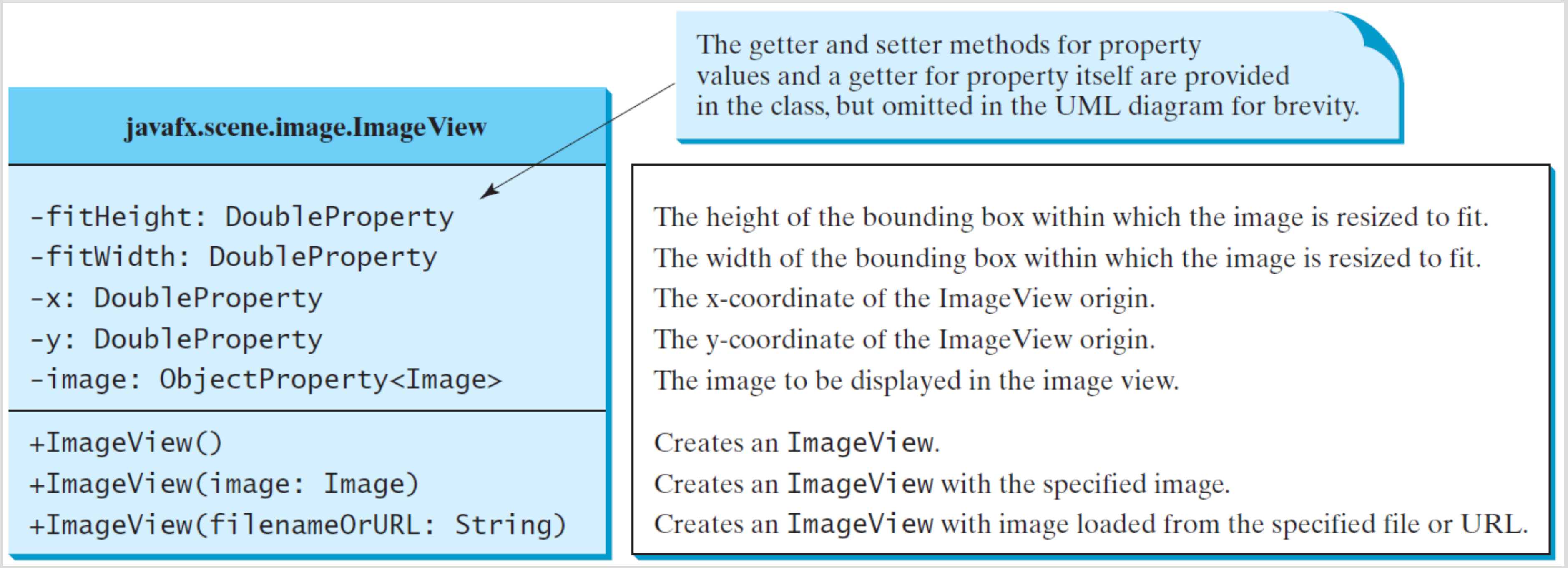
The Image Class


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 ImageView Class



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.

ShowImage

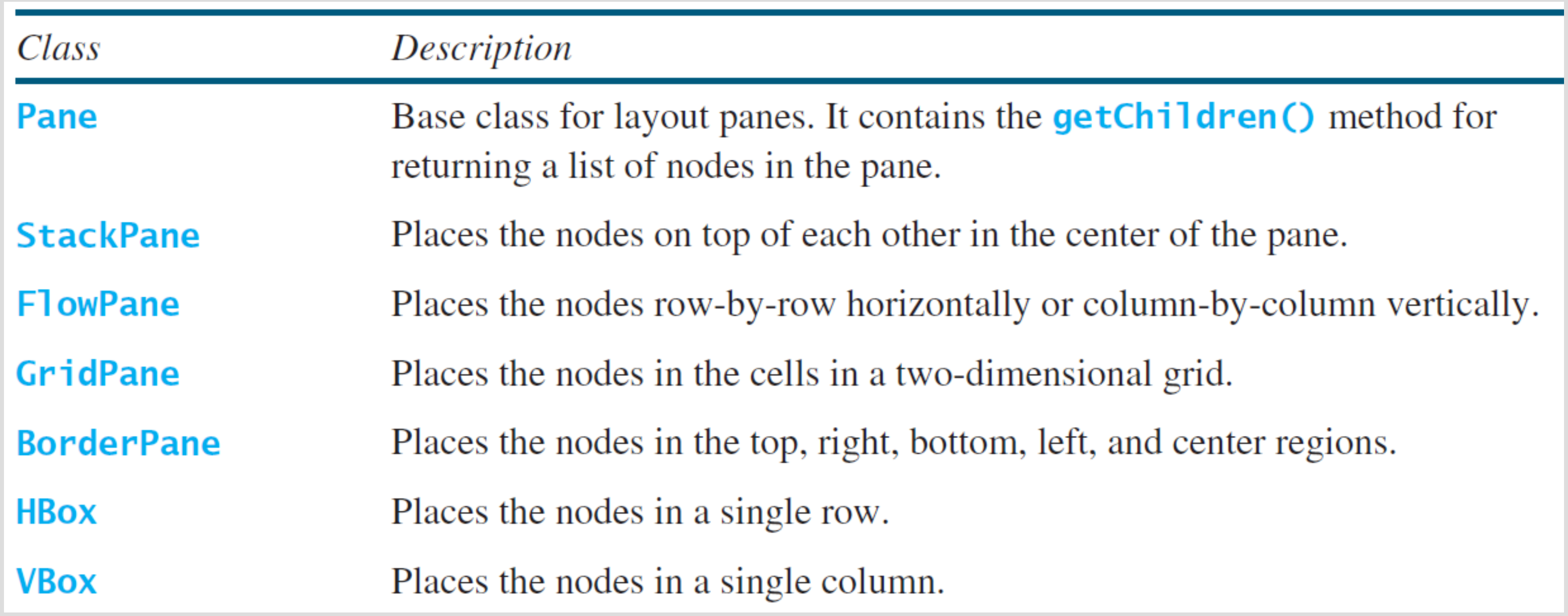
Run

Image Example

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Pane;
import javafx.geometry.Insets;
import javafx.stage.Stage;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;

public class ShowImage extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // Create a pane to hold the image views
    Pane pane = new HBox(10);
    pane.setPadding(new Insets(5, 5, 5, 5));
    Image image = new Image("image/us.gif");
    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);

    // Create a scene and place it in the stage
    Scene scene = new Scene(pane);
    primaryStage.setTitle("ShowImage"); // 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);
  }
}
```

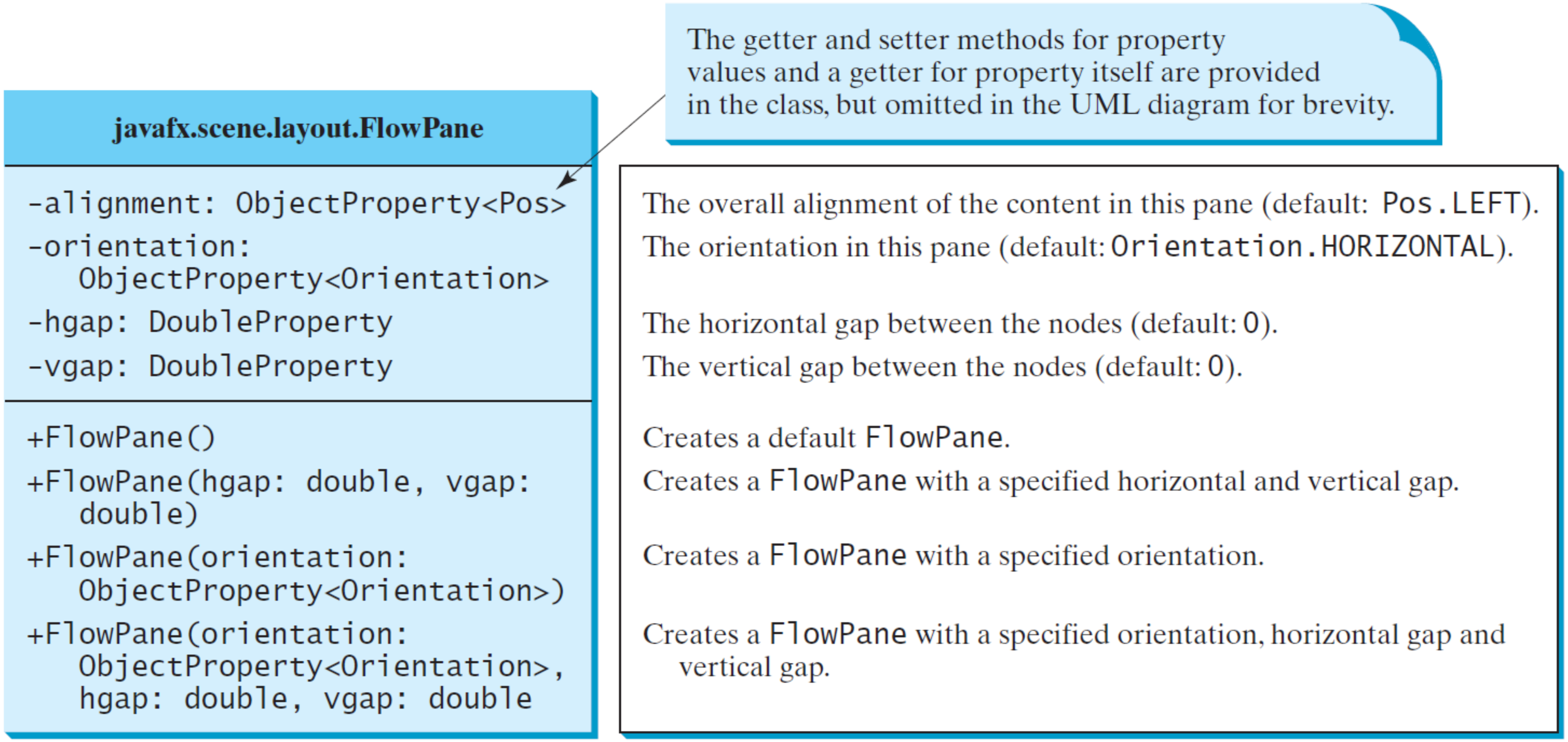
Layout Panes

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

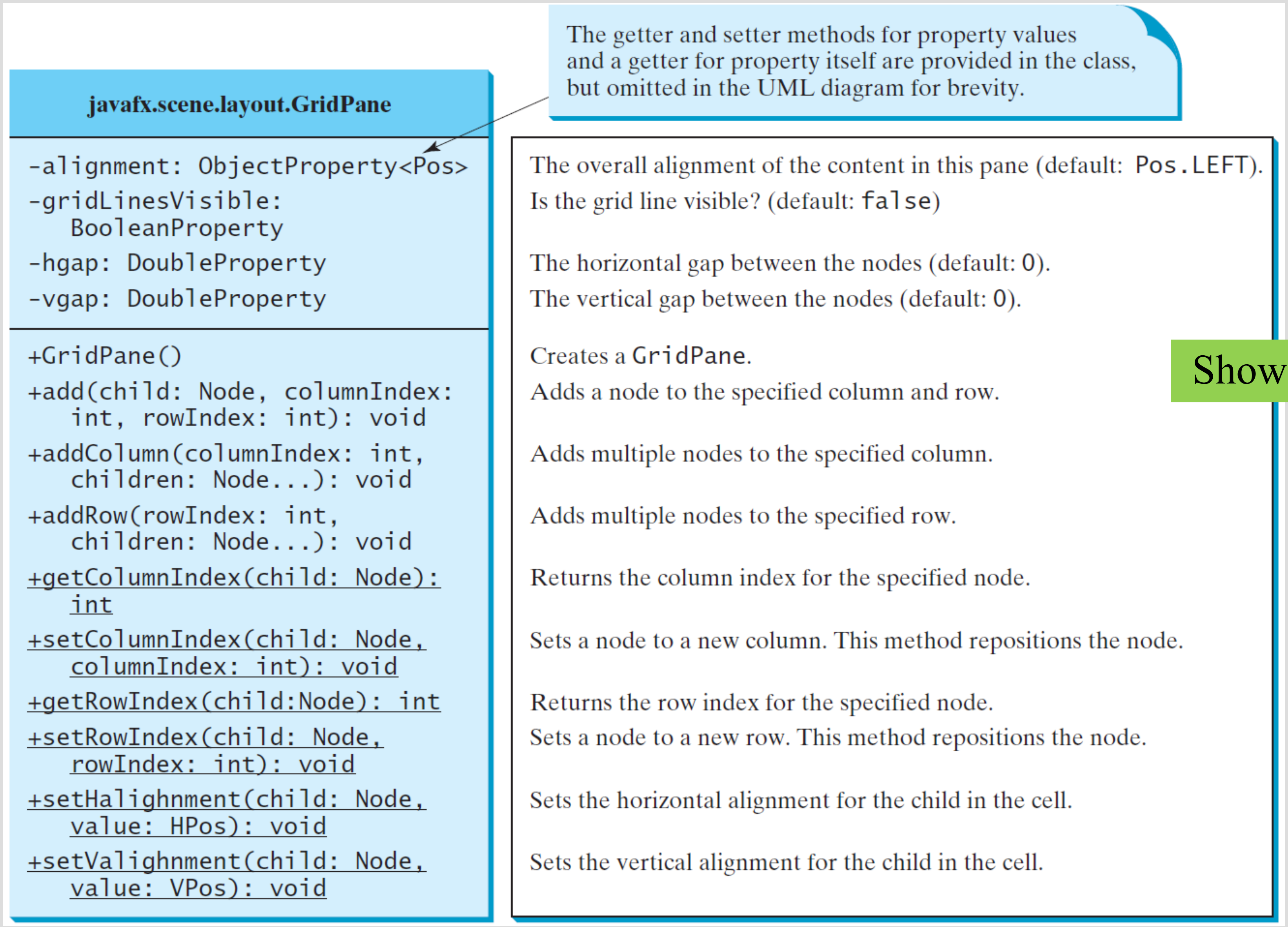
`-alignment: ObjectProperty<Pos>`	The overall alignment of the content in this pane (default: `Pos.LEFT`).
`-orientation: ObjectProperty<Orientation>`	The orientation in this pane (default: `Orientation.HORIZONTAL`).
`-hgap: DoubleProperty`	The horizontal gap between the nodes (default: `0`).
`-vgap: 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.

MultipleStageDemo Run

Flow Pane Example

```
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.FlowPane;
import javafx.stage.Stage;

public class ShowFlowPane extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // Create a pane and set its properties
    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());

    // Create a scene and place it in the stage
    Scene scene = new Scene(pane, 200, 250);
    primaryStage.setTitle("ShowFlowPane"); // 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);
  }
}
```

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.

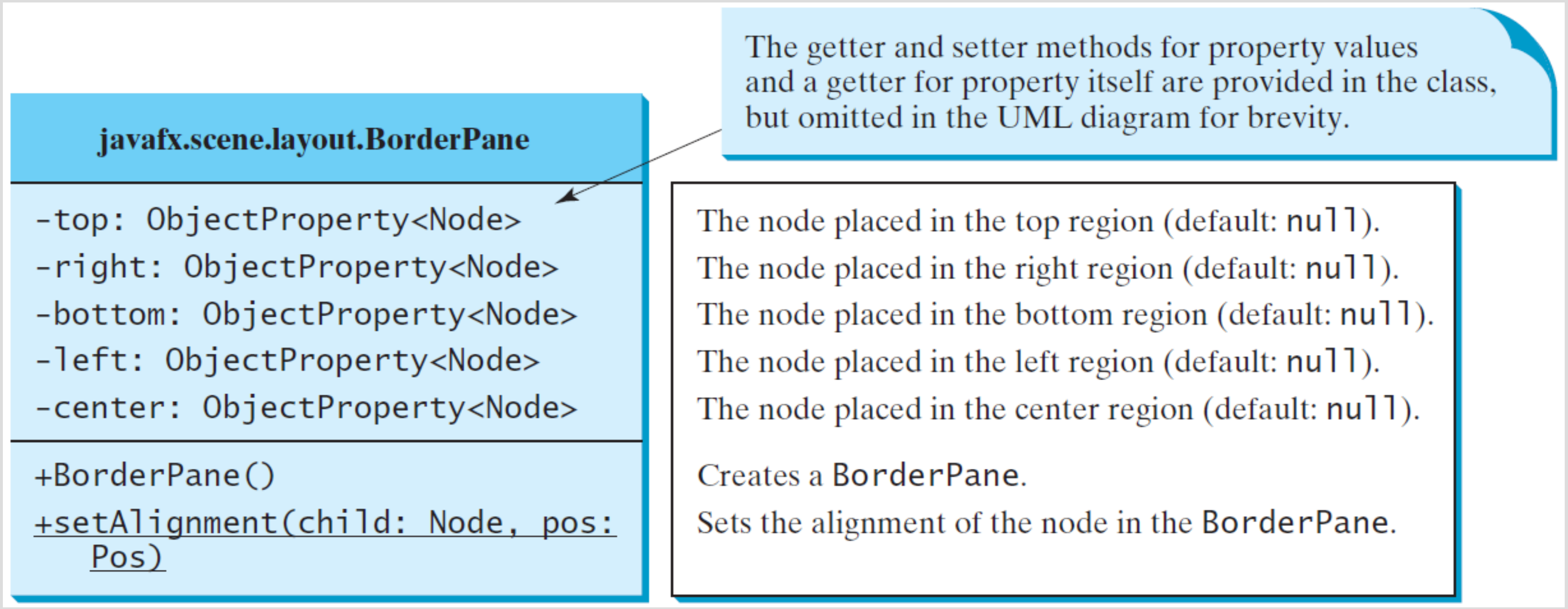
ShowGridPane

Run

Grid Pane Example

```
import javafx.application.Application;
import javafx.geometry.HPos;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;
public class ShowGridPane extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // Create a pane and set its properties
    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);
    // Create a scene and place it in the stage
    Scene scene = new Scene(pane);
    primaryStage.setTitle("ShowGridPane"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }
}
```

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`.

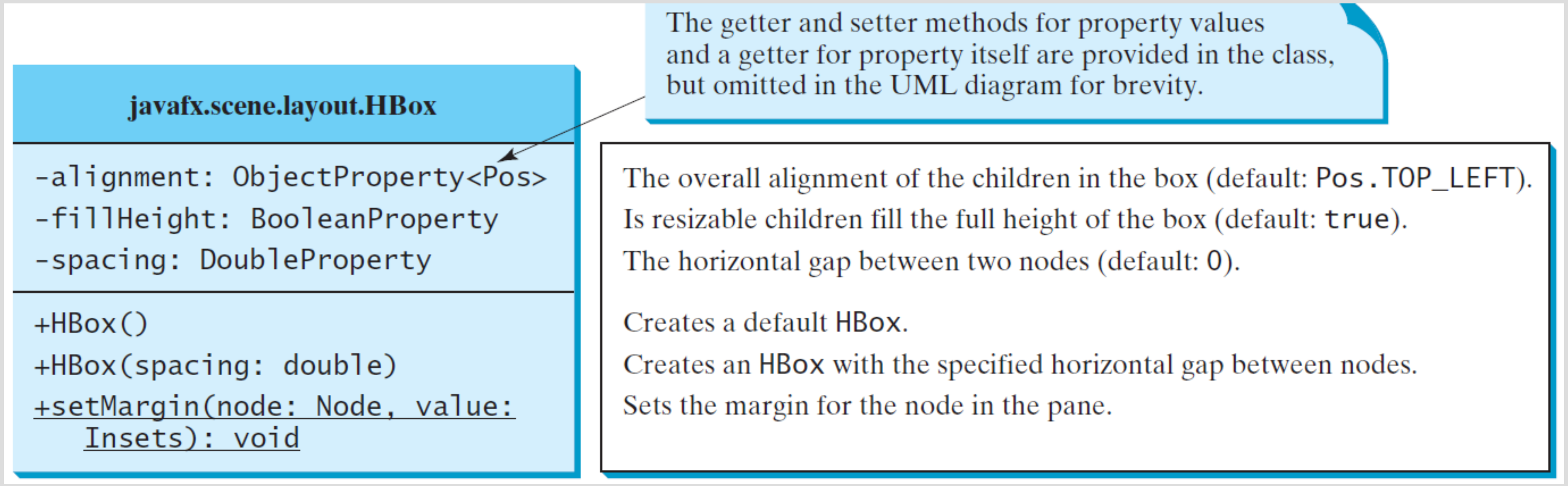
ShowBorderPane

Run

Border Pane Example

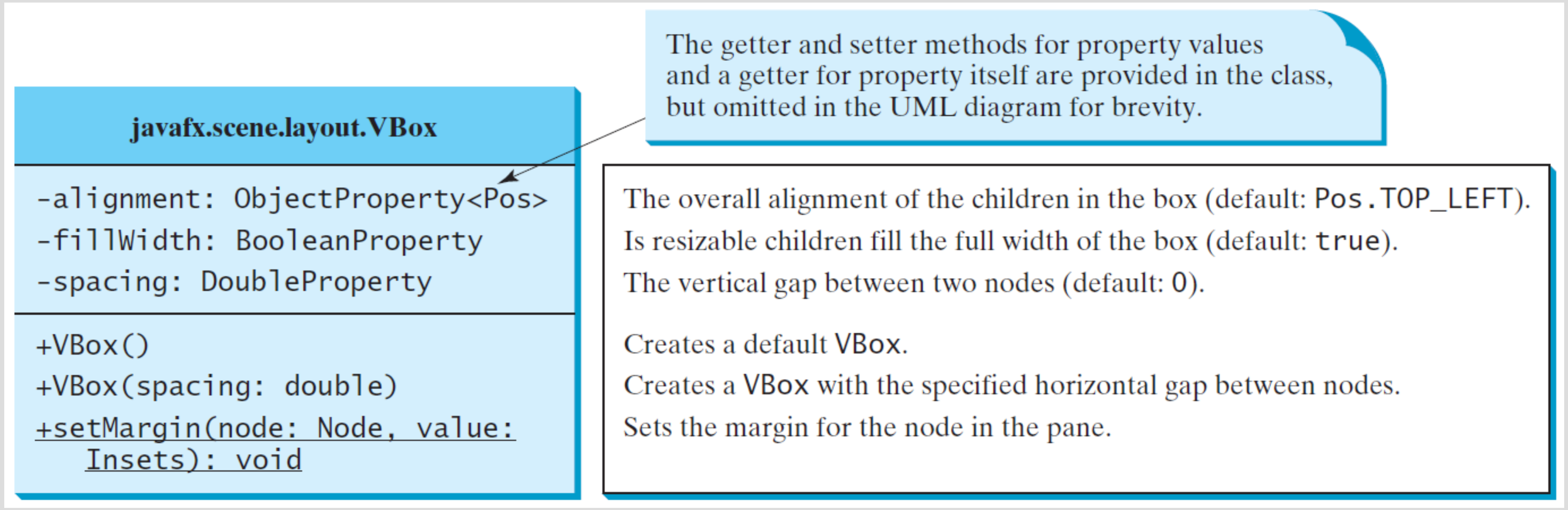
```java
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;
public class ShowBorderPane extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // 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"));
    // Create a scene and place it in the stage
    Scene scene = new Scene(pane);
    primaryStage.setTitle("ShowBorderPane"); // 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);
        }
}
// Define a custom pane to hold a label in the center of the pane
class CustomPane extends StackPane {
  public CustomPane(String title) {
    getChildren().add(new Label(title));
    setStyle("-fx-border-color: red");
    setPadding(new Insets(11.5, 12.5, 13.5, 14.5));
  }
}
```

HBox


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>`	The overall alignment of the children in the box (default: `Pos.TOP_LEFT`).
`-fillHeight: BooleanProperty`	Is resizable children fill the full height of the box (default: `true`).
`-spacing: DoubleProperty`	The horizontal gap between two nodes (default: `0`).
`+HBox()`	Creates a default `HBox`.
`+HBox(spacing: double)`	Creates an `HBox` with the specified horizontal gap between nodes.
`+setMargin(node: Node, value: Insets): void`	Sets the margin for the node in the pane.

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.

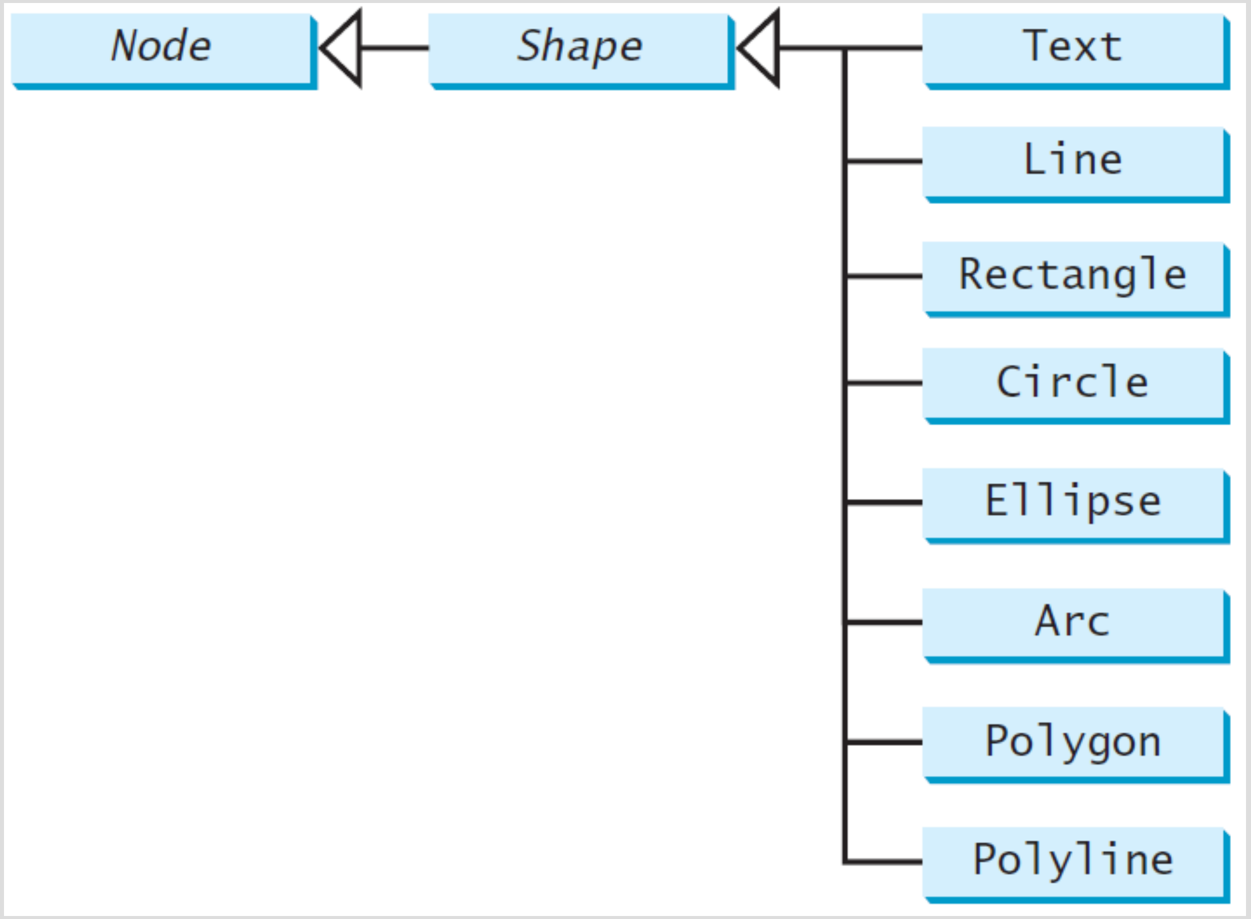
javafx.scene.layout.VBox	
-alignment: ObjectProperty<Pos>	The overall alignment of the children in the box (default: Pos.TOP_LEFT).
-fillWidth: BooleanProperty	Is resizable children fill the full width of the box (default: true).
-spacing: DoubleProperty	The vertical gap between two nodes (default: 0).
+VBox()	Creates a default VBox.
+VBox(spacing: double)	Creates a VBox with the specified horizontal gap between nodes.
+setMargin(node: Node, value: Insets): void	Sets the margin for the node in the pane.

See the code in: http://www.cs.armstrong.edu/liang/intro11e/html/ShowHBoxVBox.html

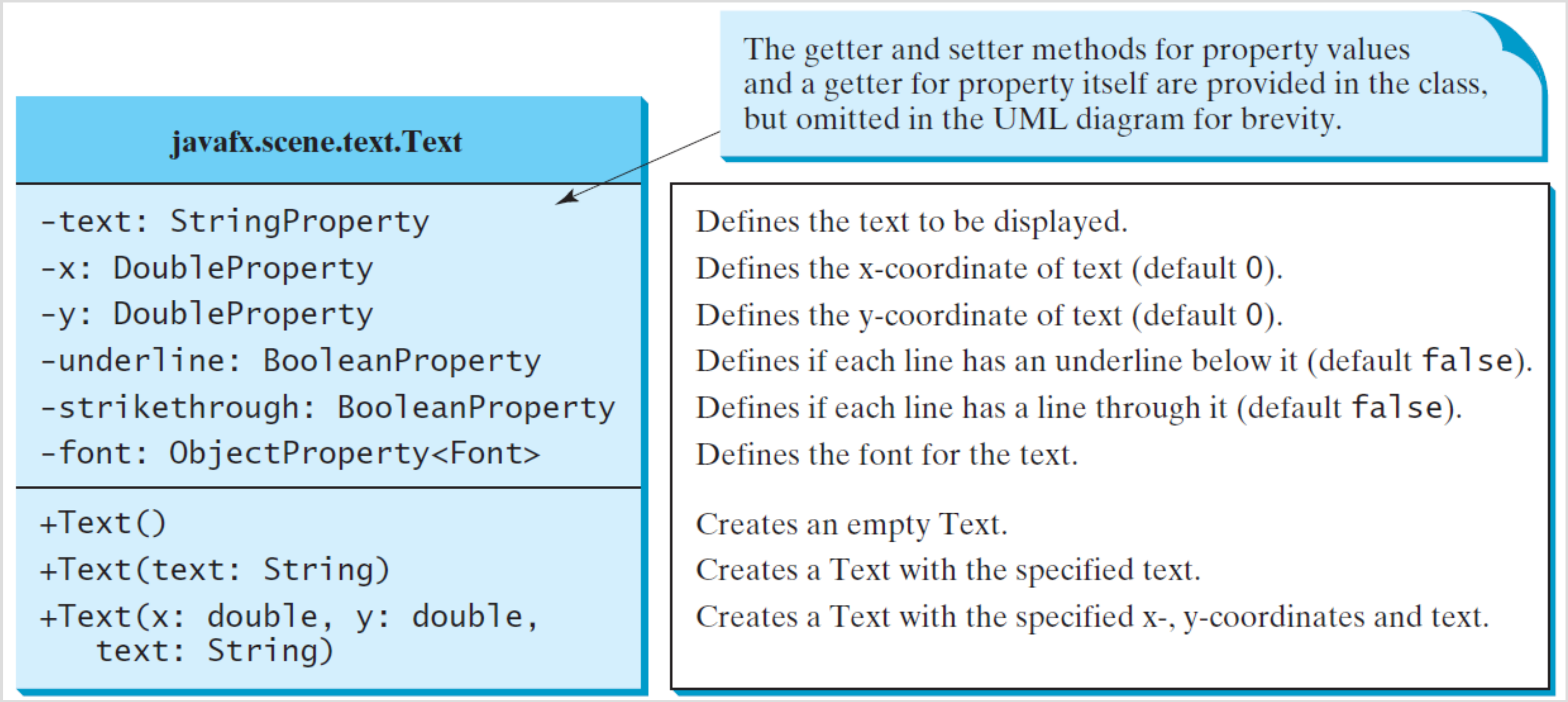
ShowHBoxVBox Run

Shapes

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



Text



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.

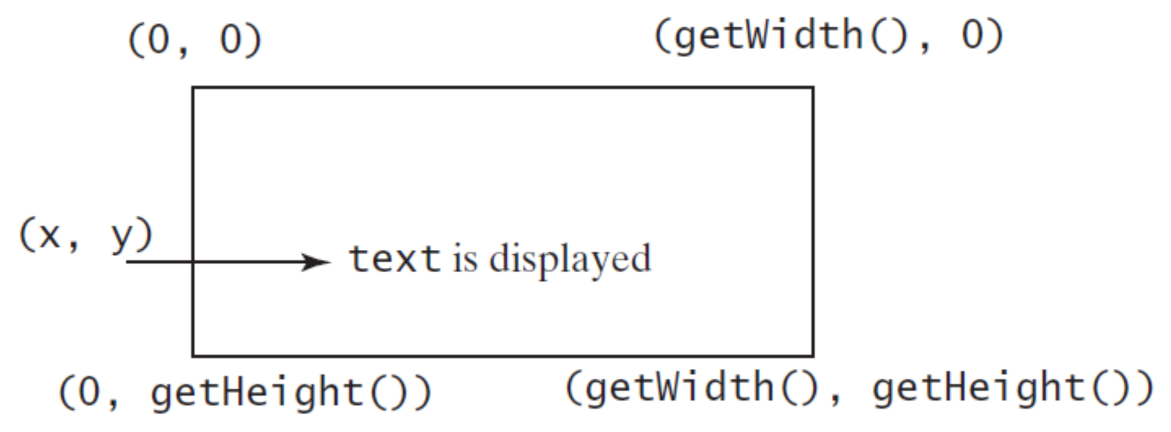
javafx.scene.text.Text

`-text: StringProperty`	Defines the text to be displayed.
`-x: DoubleProperty`	Defines the x-coordinate of text (default `0`).
`-y: DoubleProperty`	Defines the y-coordinate of text (default `0`).
`-underline: BooleanProperty`	Defines if each line has an underline below it (default `false`).
`-strikethrough: BooleanProperty`	Defines if each line has a line through it (default `false`).
`-font: ObjectProperty<Font>`	Defines the font for the text.
`+Text()`	Creates an empty Text.
`+Text(text: String)`	Creates a Text with the specified text.
`+Text(x: double, y: double, text: String)`	Creates a Text with the specified x-, y-coordinates and text.

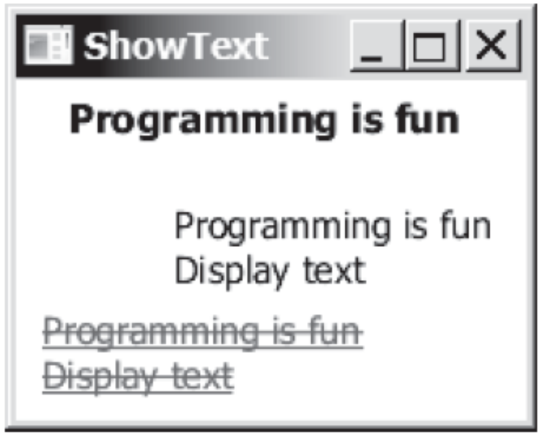
Text Example

```java
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
  }
 }
```

Text Example



(a) Text(x, y, text)



Programming is fun

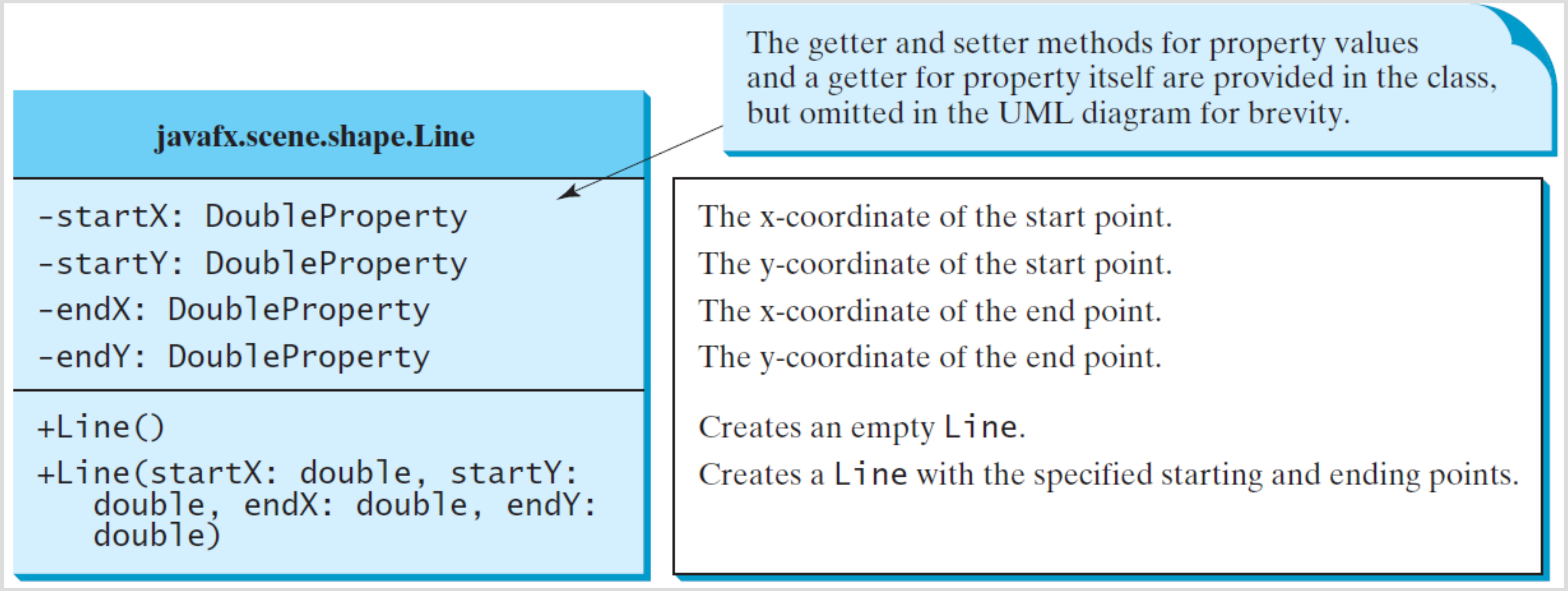
Programming is fun
Display text

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

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

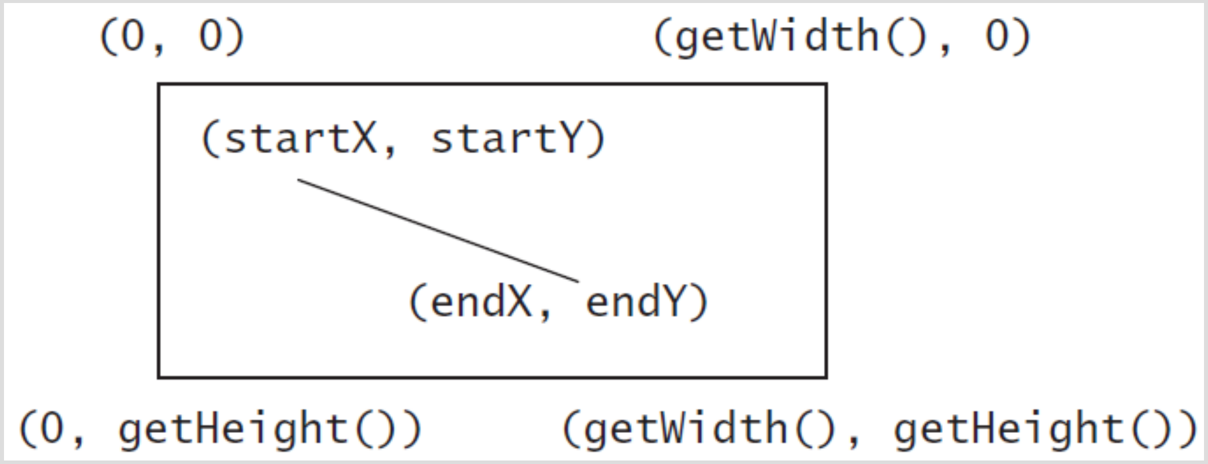
ShowText Run

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.



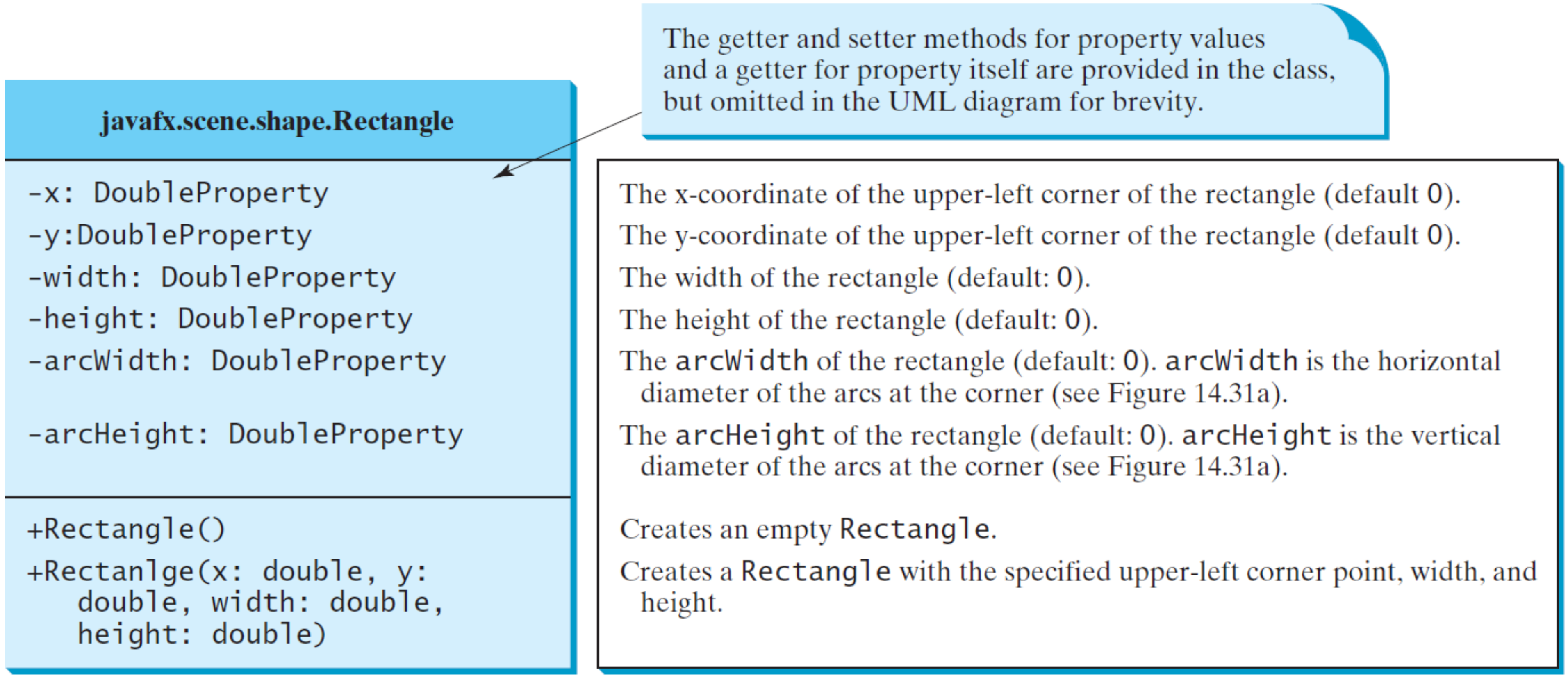
ShowLine

Run

Line Example

```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
  }
  public static void main(String[] args) {
        launch(args);
      }
}
class LinePane extends Pane {
  public LinePane() {
    Line line1 = new Line(10, 10, 10, 10);
    line1.endXProperty().bind(widthProperty().subtract(10));
    line1.endYProperty().bind(heightProperty().subtract(10));
    line1.setStrokeWidth(5);
    line1.setStroke(Color.GREEN);
    getChildren().add(line1);
    Line line2 = new Line(10, 10, 10, 10);
    line2.startXProperty().bind(widthProperty().subtract(10));
    line2.endYProperty().bind(heightProperty().subtract(10));
    line2.setStrokeWidth(5);
    line2.setStroke(Color.GREEN);
    getChildren().add(line2);
  }
}
```

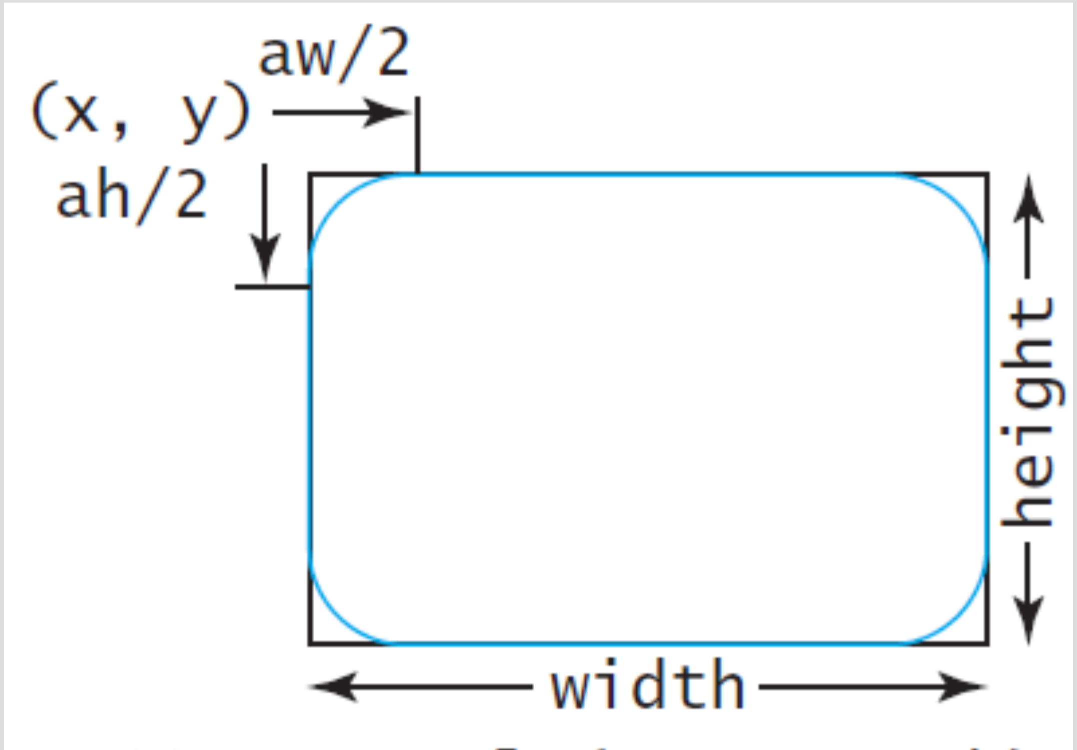
Rectangle



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.

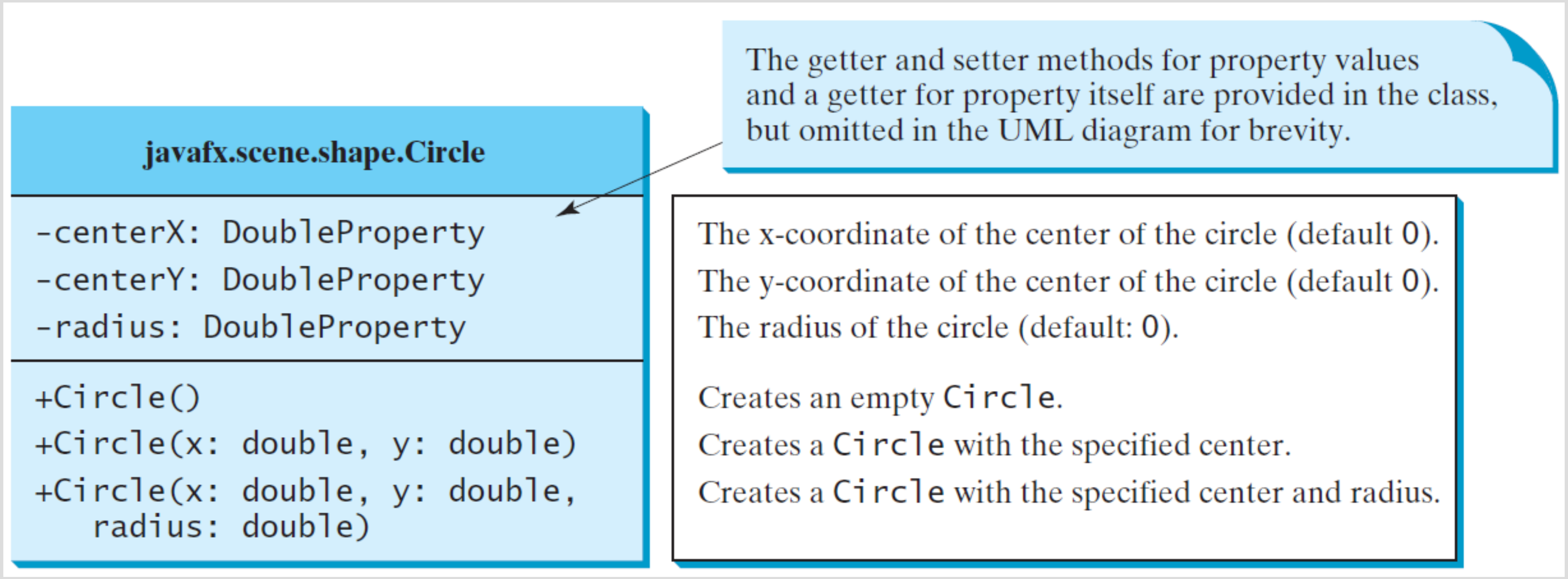
Rectangle Example



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

See example: http://www.cs.armstrong.edu/liang/intro11e/html/ShowRectangle.html

Circle



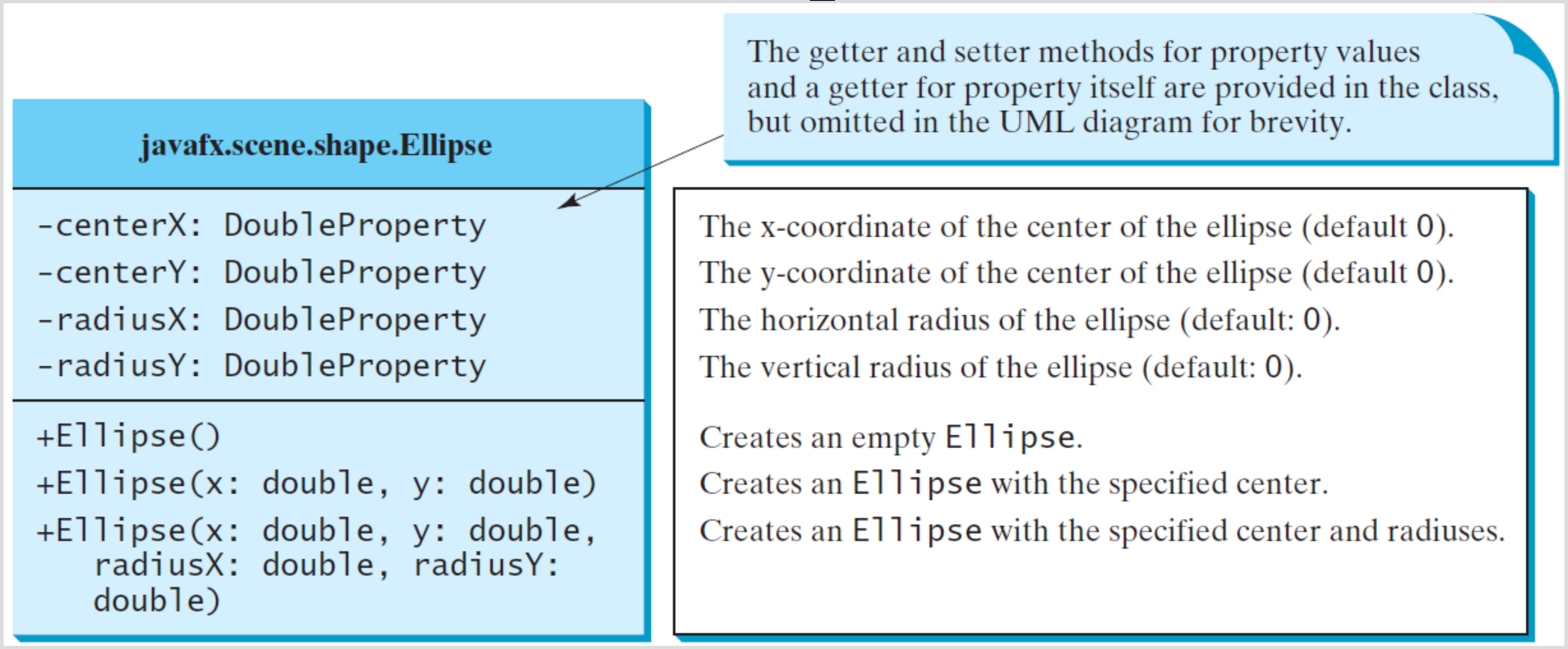
The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

```
javafx.scene.shape.Circle
-centerX: DoubleProperty
-centerY: DoubleProperty
-radius: DoubleProperty
+Circle()
+Circle(x: double, y: double)
+Circle(x: double, y: double,
   radius: double)
```

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

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

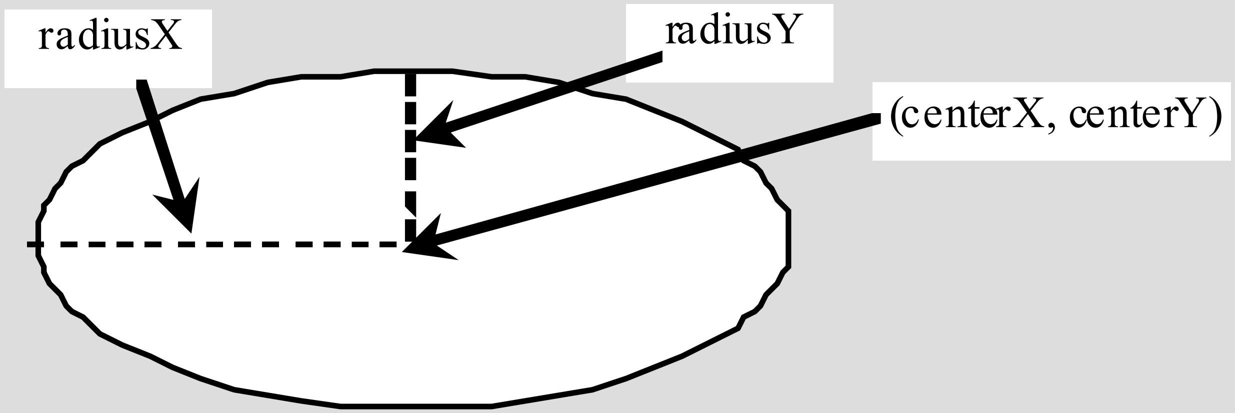
Ellipse



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.

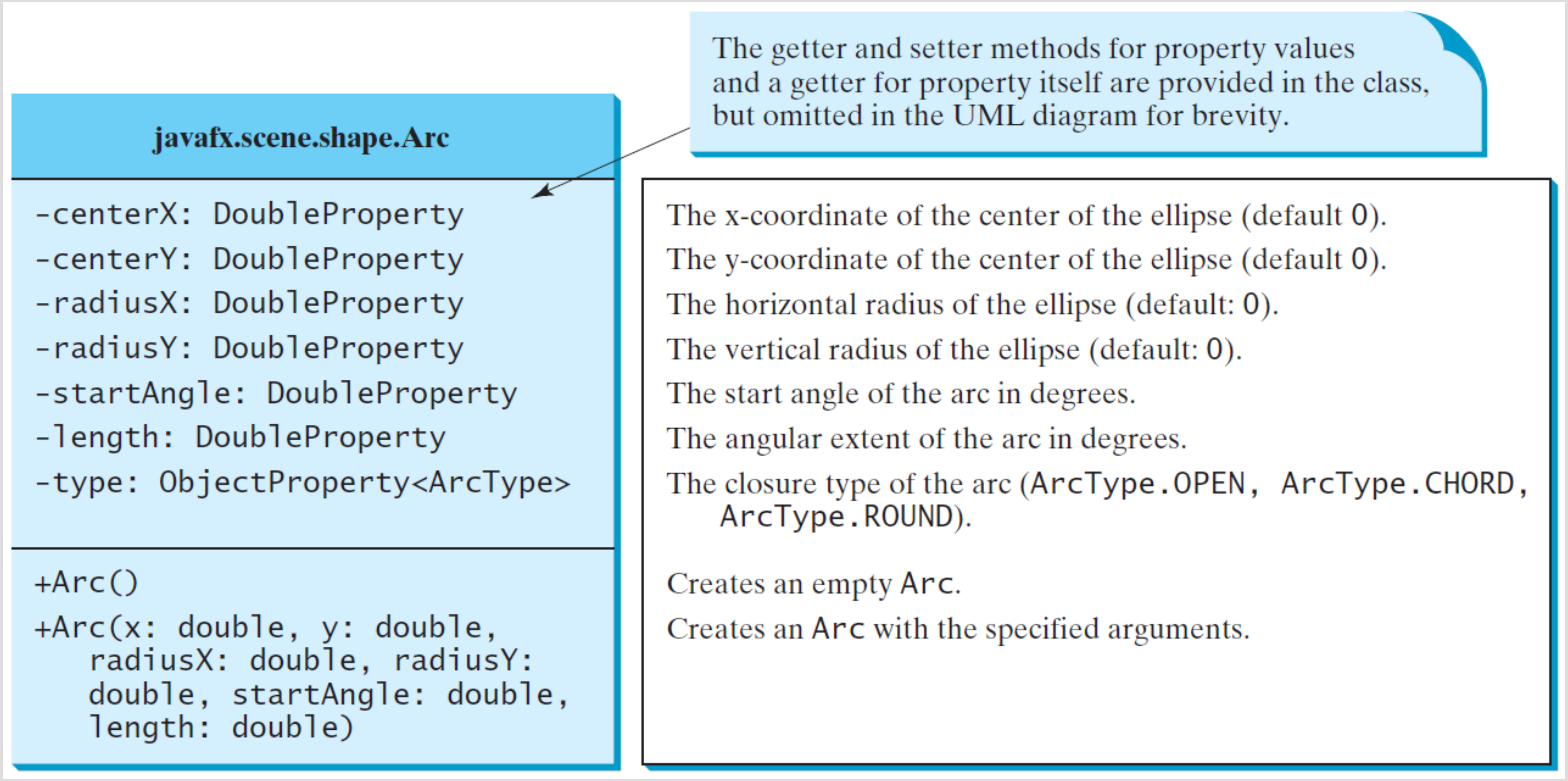
javafx.scene.shape.Ellipse

`-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.



See example: www.cs.armstrong.edu/liang/intro11e/html/ShowEllipse.html

Arc

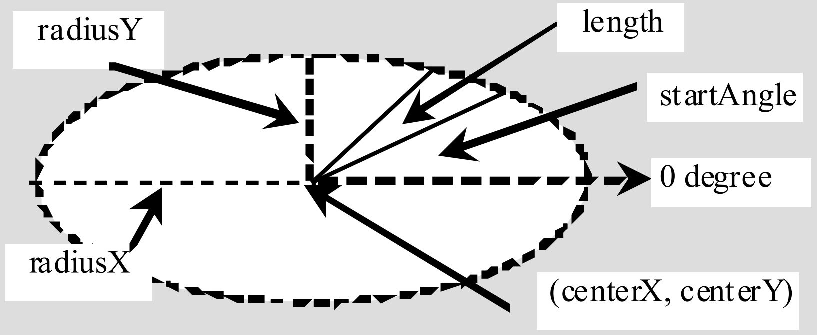


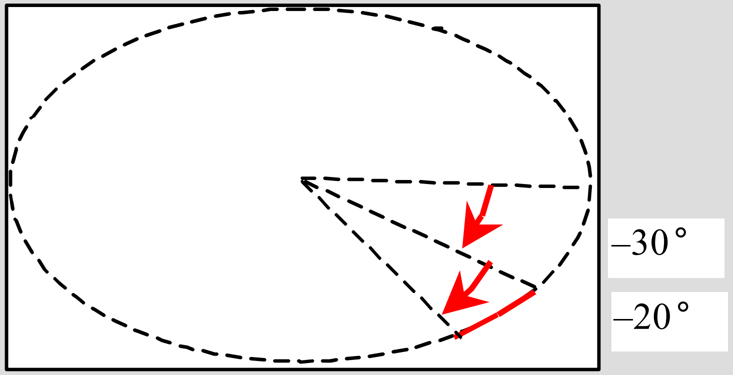
The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.shape.Arc

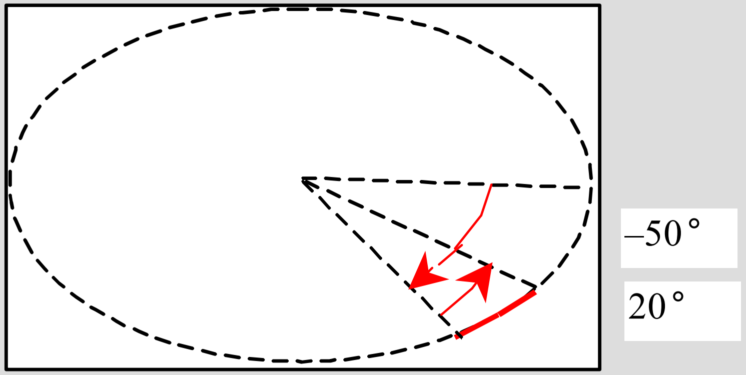
`-centerX: DoubleProperty`	The x-coordinate of the center of the ellipse (default `0`).
`-centerY: DoubleProperty`	The y-coordinate of the center of the ellipse (default `0`).
`-radiusX: DoubleProperty`	The horizontal radius of the ellipse (default: `0`).
`-radiusY: DoubleProperty`	The vertical radius of the ellipse (default: `0`).
`-startAngle: DoubleProperty`	The start angle of the arc in degrees.
`-length: DoubleProperty`	The angular extent of the arc in degrees.
`-type: ObjectProperty<ArcType>`	The closure type of the arc (`ArcType.OPEN`, `ArcType.CHORD`, `ArcType.ROUND`).
`+Arc()`	Creates an empty `Arc`.
`+Arc(x: double, y: double, radiusX: double, radiusY: double, startAngle: double, length: double)`	Creates an `Arc` with the specified arguments.

Arc Examples





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



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

See example: www.cs.armstrong.edu/liang/intro11e/html/ShowArc.html