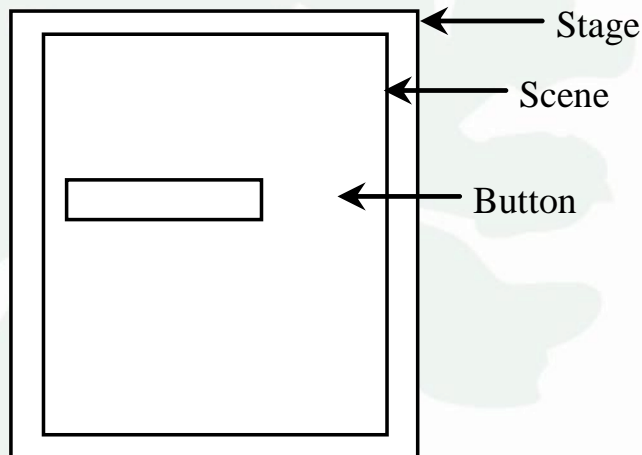


Basic Structure of JavaFX

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



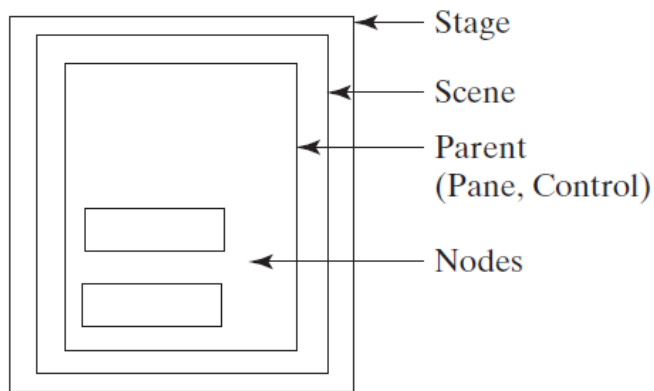
MyJavaFX

Run

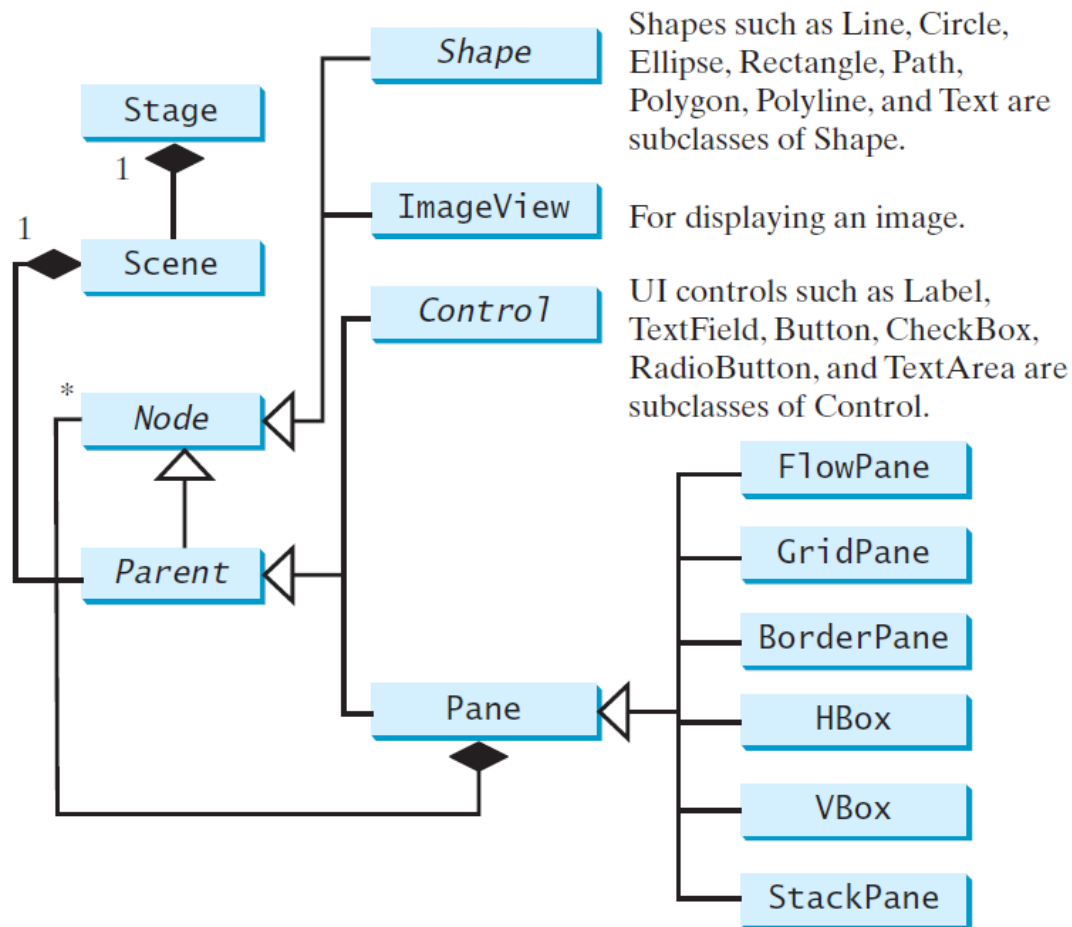
MultipleStageDemo

Run

Panes, UI Controls, and Shapes



(a)

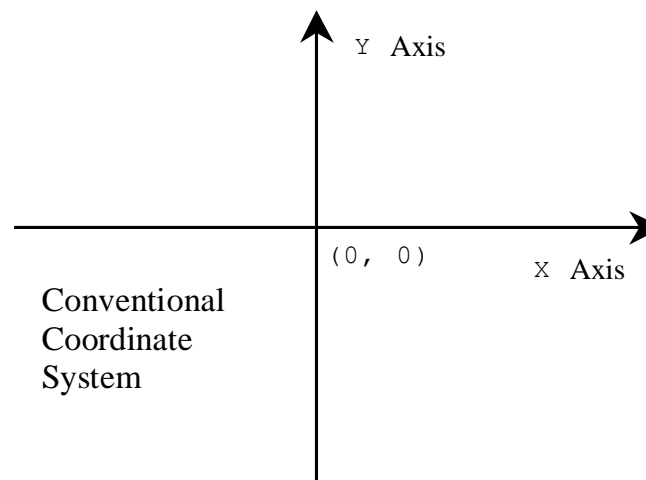
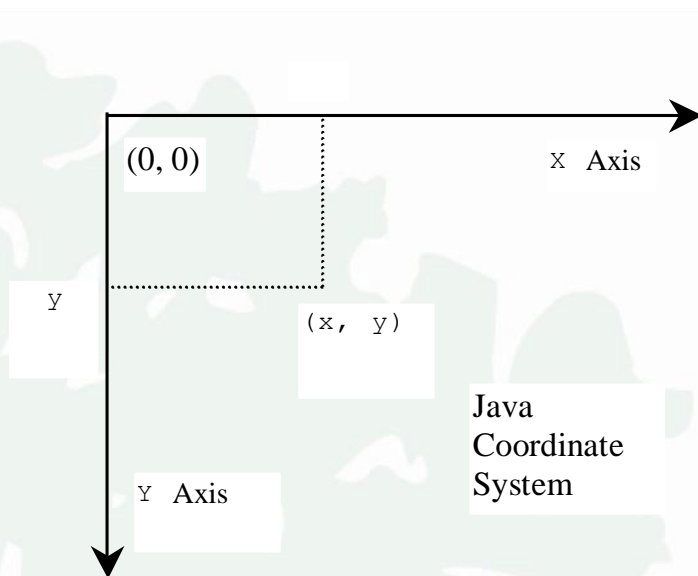


(b)

ButtonInPane Run

Display a Shape

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



ShowCircle

Run

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*.

A large, light green silhouette of a tree is positioned on the left side of the slide, extending from the bottom to the middle.

ShowCircleCentered

Run

Binding Property: getter, setter, and property getter

```
public class SomeClassName {
    private PropertyType x;

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

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

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

(a) x is a binding property

```
public class Circle {
    private DoubleProperty centerX;

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

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

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

(b) centerX is binding property

Uni/Bidirectional Binding



BindingDemo

Run

BidirectionalBindingDemo

Run

Common Properties and Methods for Nodes

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

NodeStyleRotateDemo

Run

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

+ Color(r: double, g: double, b: double, opacity: double)
+ brighter(): Color
+ darker(): Color
+ color(r: double, g: double, b: double): Color
+ color(r: double, g: double, b: double, opacity: double): Color
+ rgb(r: int, g: int, b: int): Color
+ rgb(r: int, g: int, b: int, opacity: double): Color
    
```

- The red value of this Color (between 0.0 and 1.0).
- The green value of this Color (between 0.0 and 1.0).
- The blue value of this Color (between 0.0 and 1.0).
- The opacity of this Color (between 0.0 and 1.0).
- Creates a Color with the specified red, green, blue, and opacity values.
- Creates a Color that is a brighter version of this Color.
- Creates a Color that is a darker version of this Color.
- Creates an opaque Color with the specified red, green, and blue values.
- Creates a Color with the specified red, green, blue, and opacity values.
- Creates a Color with the specified red, green, and blue values in the range from 0 to 255.
- Creates a Color with the specified red, green, and blue values in the range from 0 to 255 and a given opacity.

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
-name: String
-family: String

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

The size of this font.

The name of this font.

The family of this font.

Creates a Font with the specified size.

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

Creates a Font with the specified name and size.

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

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

Returns a list of font family names.

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

FontDemo

Run

The Image Class

javafx.scene.image.Image

-error: ReadOnlyBooleanProperty
 -height: ReadOnlyBooleanProperty
 -width: ReadOnlyBooleanProperty
 -progress: ReadOnlyBooleanProperty

+Image(filenameOrURL: String)

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

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

The ImageView Class

javafx.scene.image.ImageView

-fitHeight: DoubleProperty
 -fitWidth: DoubleProperty
 -x: DoubleProperty
 -y: DoubleProperty
 -image: ObjectProperty<Image>

+ImageView()
 +ImageView(image: Image)
 +ImageView(filenameOrURL: String)

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 height of the bounding box within which the image is resized to fit.
 The width of the bounding box within which the image is resized to fit.
 The x-coordinate of the ImageView origin.
 The y-coordinate of the ImageView origin.
 The image to be displayed in the image view.

Creates an ImageView.
 Creates an ImageView with the specified image.
 Creates an ImageView with image loaded from the specified file or URL.

ShowImage

Run

Layout Panes

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

<i>Class</i>	<i>Description</i>
Pane	Base class for layout panes. It contains the <code>getChildren()</code> 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>
 -orientation: ObjectProperty<Orientation>
 -hgap: DoubleProperty
 -vgap: DoubleProperty

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

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

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

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

Creates a default FlowPane.

Creates a FlowPane with a specified horizontal and vertical gap.

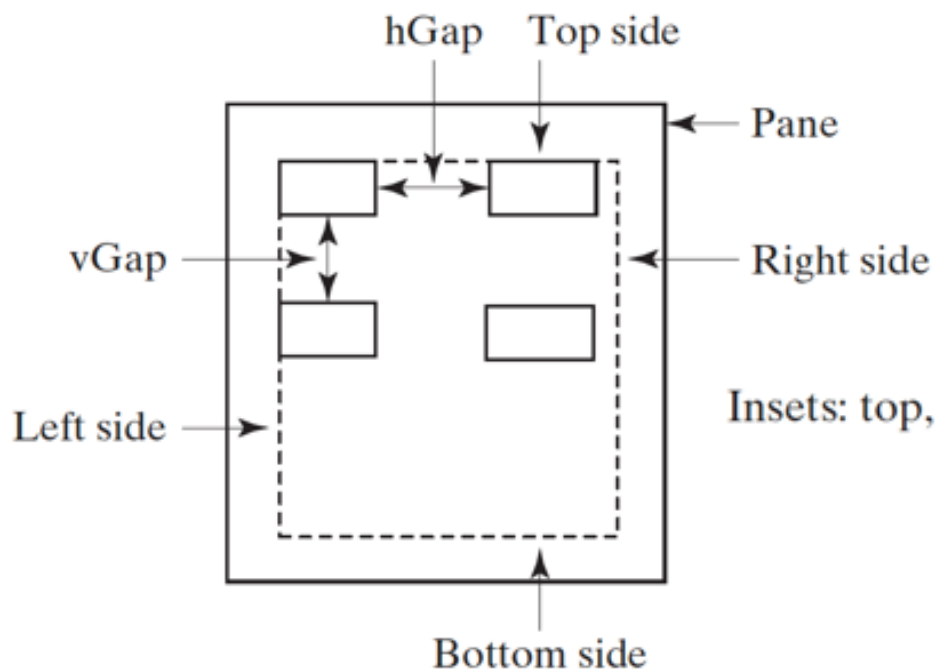
Creates a FlowPane with a specified orientation.

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

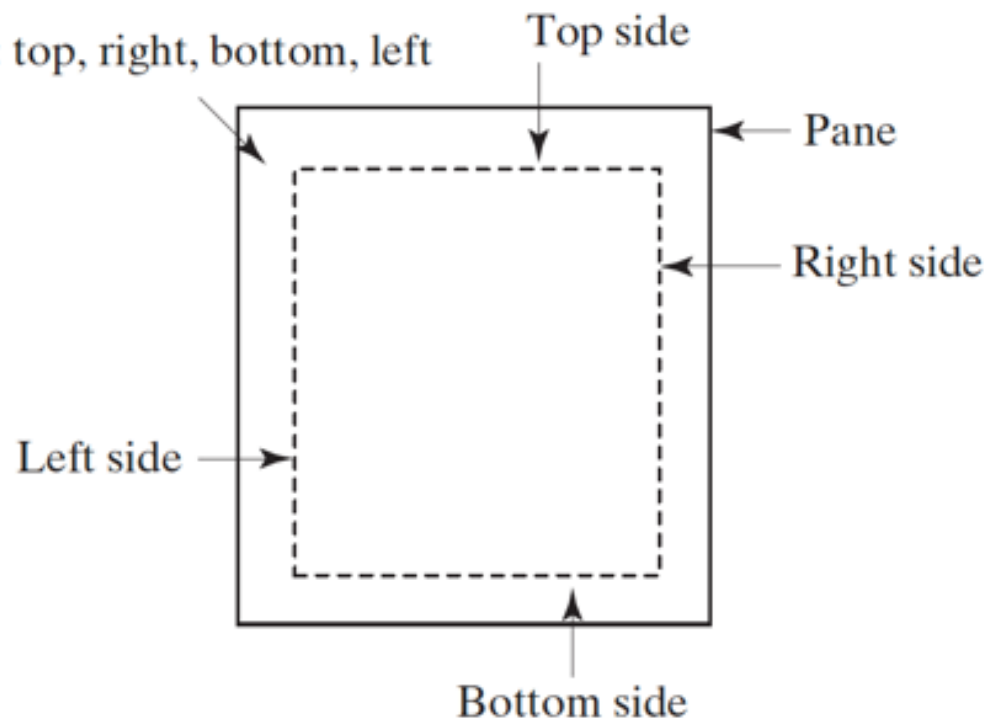
MultipleStageDemo

Run

Gaps



Insets: top, right, bottom, left



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>
 -gridLinesVisible: BooleanProperty
 -hgap: DoubleProperty
 -vgap: DoubleProperty

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

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

Is the grid line visible? (default: false)

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

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

Creates a GridPane.

Adds a node to the specified column and row.

Adds multiple nodes to the specified column.

Adds multiple nodes to the specified row.

Returns the column index for the specified node.

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

Returns the row index for the specified node.

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

Sets the horizontal alignment for the child in the cell.

Sets the vertical alignment for the child in the cell.

ShowGridPane

Run

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.

ShowBorderPane

Run

HBox

javafx.scene.layout.HBox

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

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

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

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

Creates a default HBox.

Creates an HBox with the specified horizontal gap between nodes.

Sets the margin for the node in the pane.

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>
 -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).

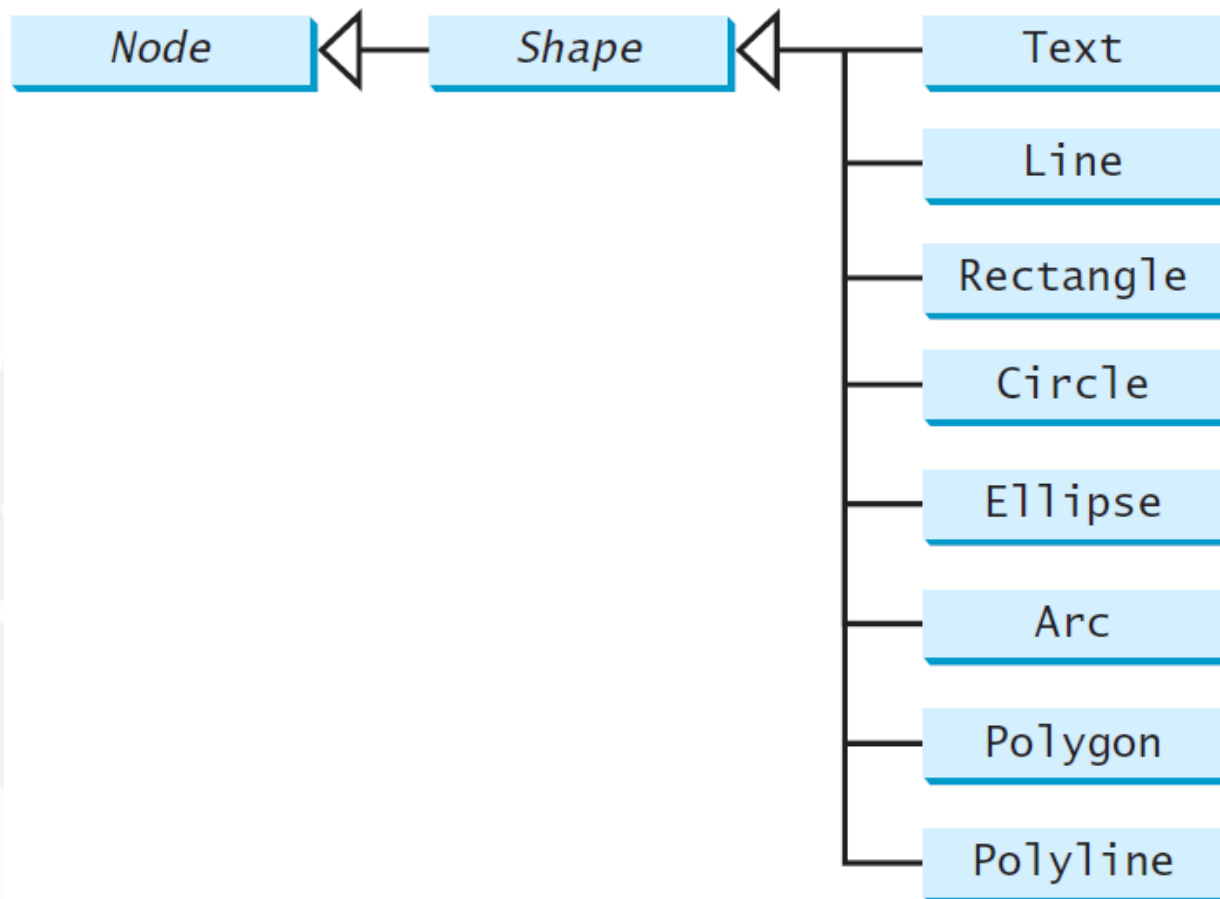
Creates a default VBox.
 Creates a VBox with the specified horizontal gap between nodes.
 Sets the margin for the node in the pane.

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
-x: DoubleProperty
-y: DoubleProperty
-underline: BooleanProperty
-strikethrough: BooleanProperty
-font: ObjectProperty<Font>

```

```

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

```

Defines the text to be displayed.

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

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

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

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

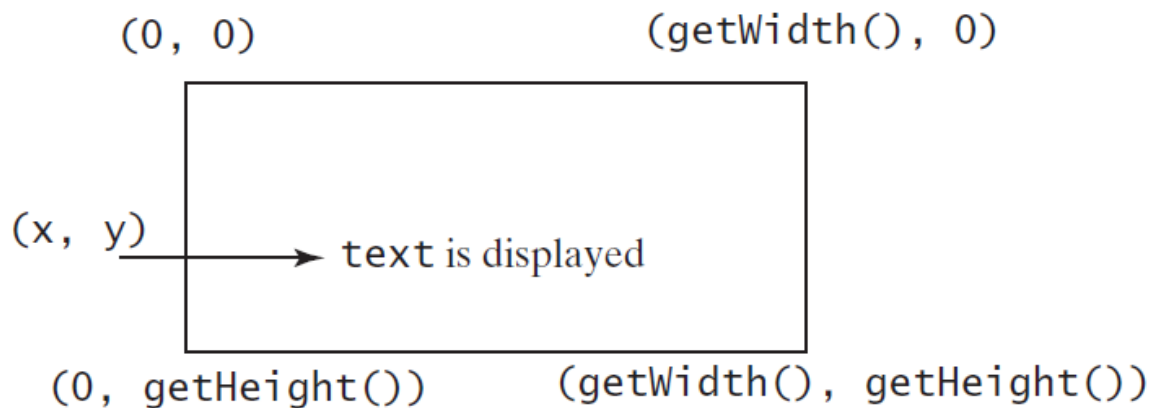
Defines the font for the text.

Creates an empty Text.

Creates a Text with the specified text.

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

Text Example



(a) `Text(x, y, text)`



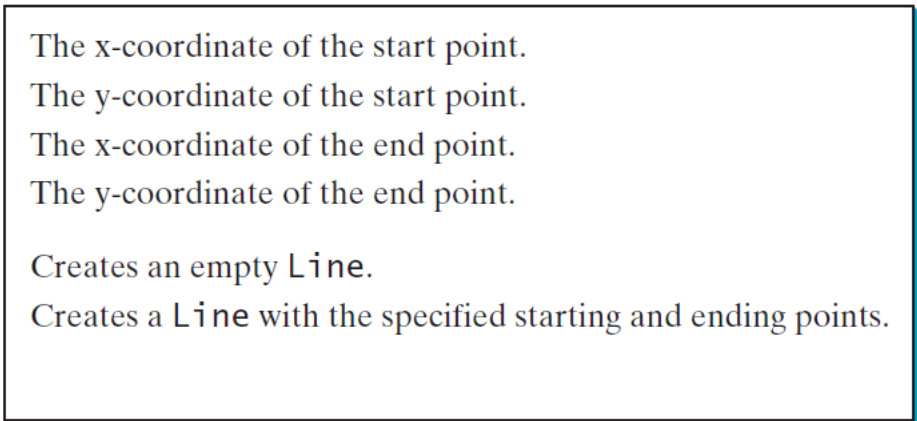
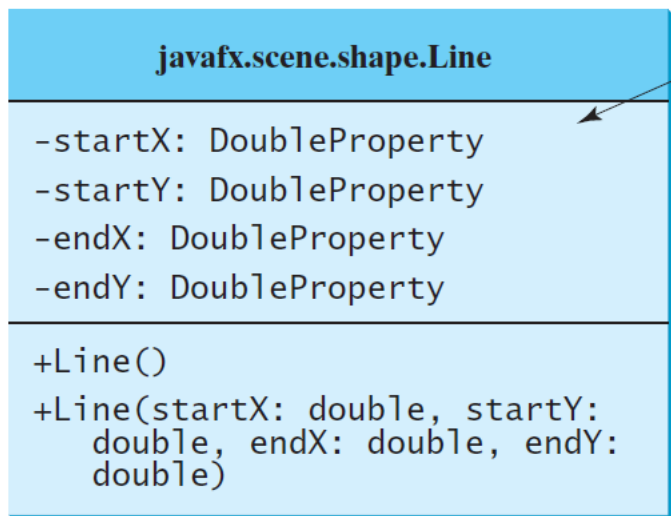
(b) *Three Text objects are displayed*

ShowText

Run

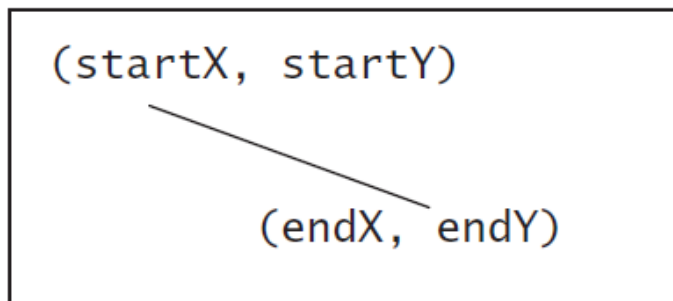
Line

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.



(0, 0)

(getWidth(), 0)



ShowLine

Run

(0, getHeight())

(getWidth(), getHeight())

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

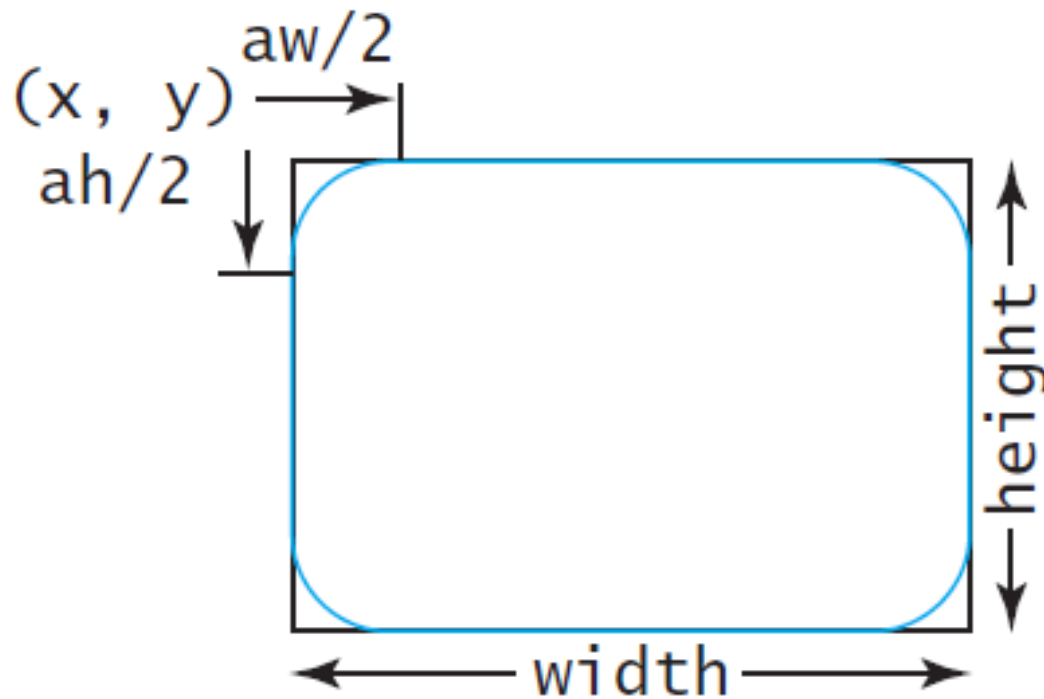
+Rectangle()
 +Rectangle(x: double, y: double, width: double, height: double)

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

Creates an empty Rectangle.

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

Rectangle Example



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

ShowRectangle

Run

Circle

javafx.scene.shape.Circle

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

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

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

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

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

Ellipse

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

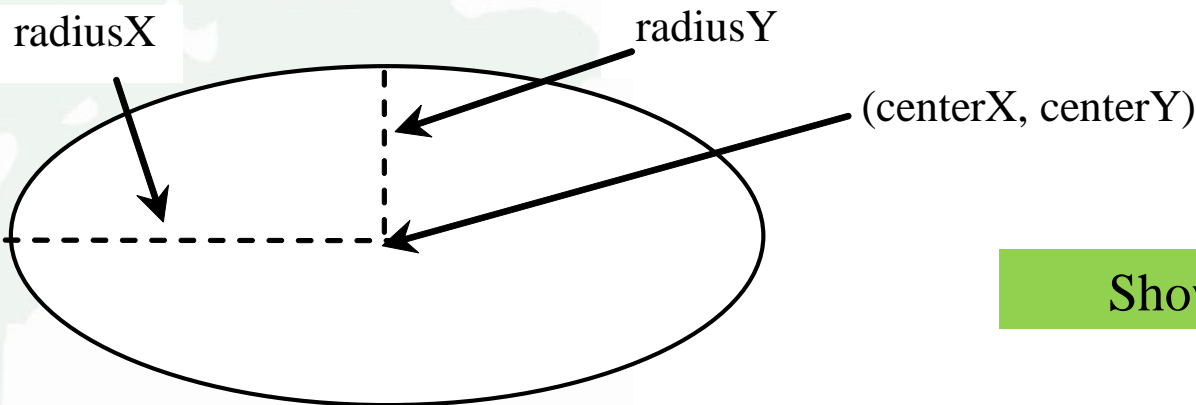
javafx.scene.shape.Ellipse

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

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

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

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



ShowEllipse

Run

Arc

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

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

The x-coordinate of the center of the ellipse (default 0).

The y-coordinate of the center of the ellipse (default 0).

The horizontal radius of the ellipse (default: 0).

The vertical radius of the ellipse (default: 0).

The start angle of the arc in degrees.

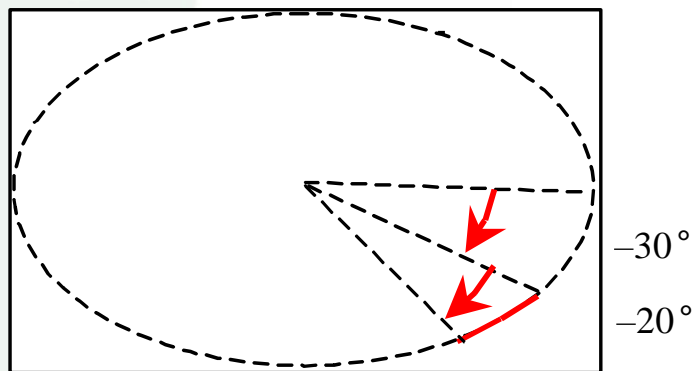
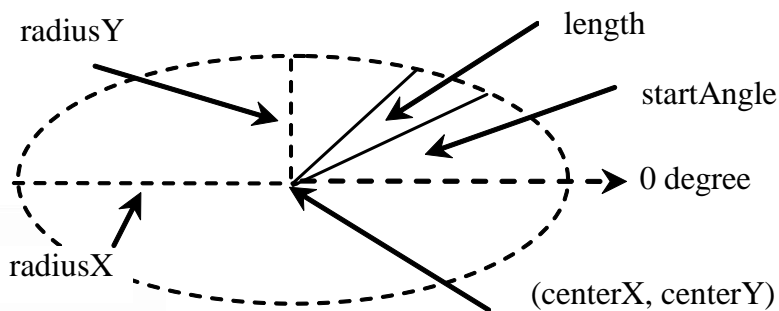
The angular extent of the arc in degrees.

The closure type of the arc (ArcType.OPEN, ArcType.CHORD, ArcType.ROUND).

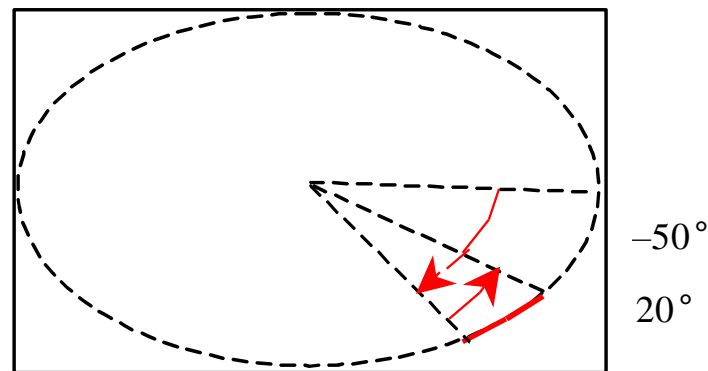
Creates an empty Arc.

Creates an Arc with the specified arguments.

Arc Examples



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

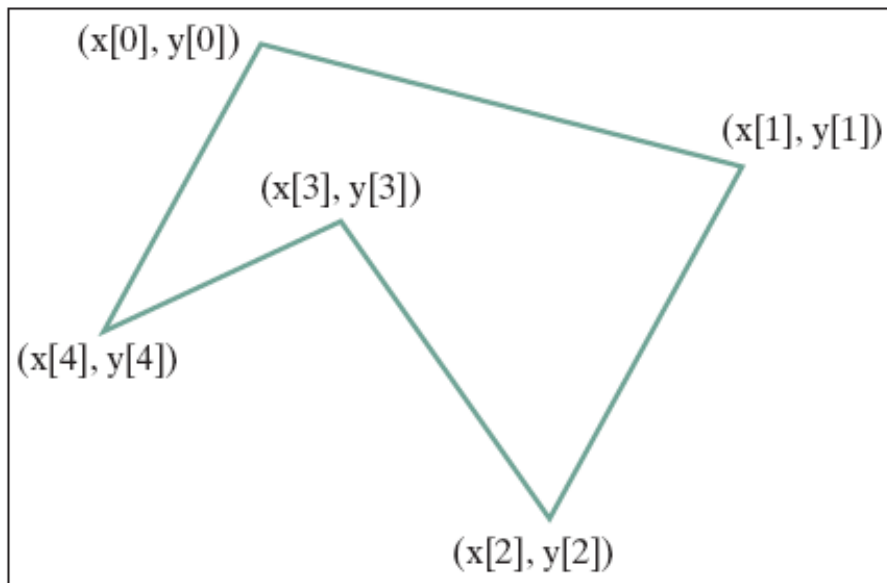


(b) Negative starting angle -50° and positive spanning angle 20°

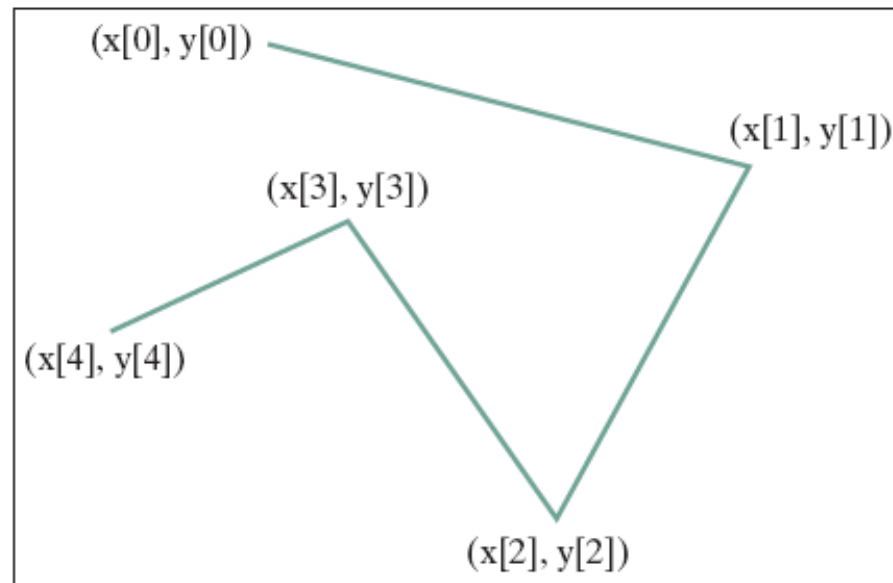
ShowArc

Run

Polygon and Polyline



(a) Polygon



(b) Polyline

Polygon

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

javafx.scene.shape.Polygon

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

Creates an empty polygon.

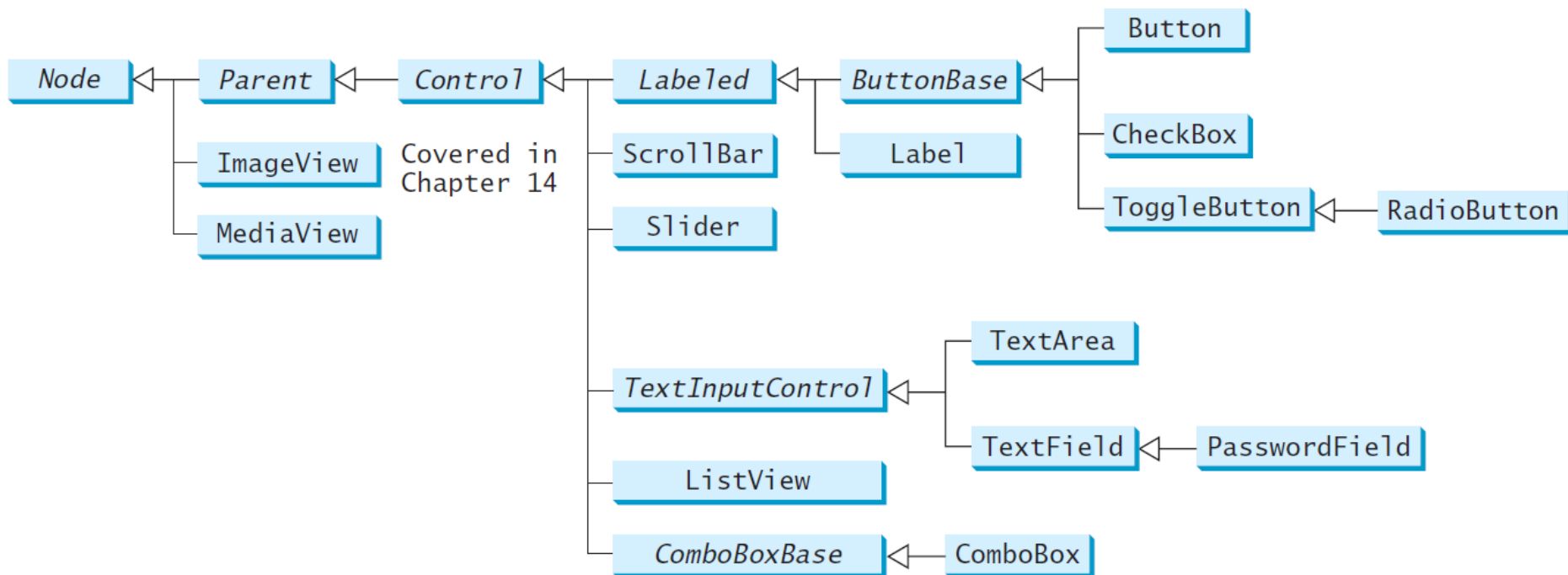
Creates a polygon with the given points.

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

ShowPolygon

Run

Frequently Used UI Controls



Throughout this book, the prefixes **lbl**, **bt**, **chk**, **rb**, **tf**, **pf**, **ta**, **cbo**, **lv**, **scb**, **sld**, and **mp** are used to name reference variables for **Label**, **Button**, **CheckBox**, **RadioButton**, **TextField**, **PasswordField**, **TextArea**, **ComboBox**, **ListView**, **ScrollBar**, **Slider**, and **MediaPlayer**.

Labeled

A *label* is a display area for a short text, a node, or both. It is often used to label other controls (usually text fields). Labels and buttons share many common properties. These common properties are defined in the **Labeled** class.

javafx.scene.control.Labeled

```

-alignment: ObjectProperty<Pos>
-contentDisplay:
    ObjectProperty<ContentDisplay>
-graphic: ObjectProperty<Node>
-graphicTextGap: DoubleProperty
-textFill: ObjectProperty<Paint>
-text: StringProperty
-underline: BooleanProperty
-wrapText: BooleanProperty
    
```

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.

Specifies the alignment of the text and node in the labeled.

Specifies the position of the node relative to the text using the constants TOP, BOTTOM, LEFT, and RIGHT defined in ContentDisplay.

A graphic for the labeled.

The gap between the graphic and the text.

The paint used to fill the text.

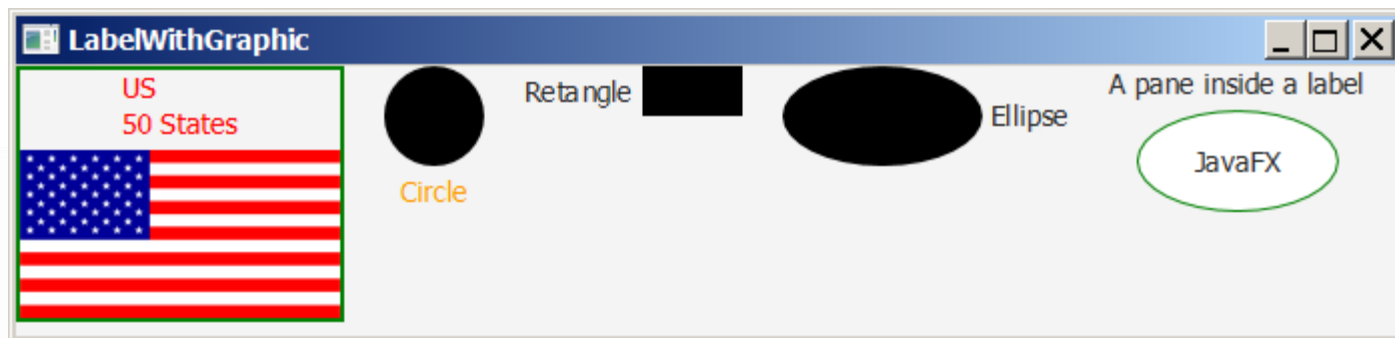
A text for the labeled.

Whether text should be underlined.

Whether text should be wrapped if the text exceeds the width.

Label

The Label class defines labels.



javafx.scene.control.Labeled



javafx.scene.control.Label

+Label()
 +Label(text: String)
 +Label(text: String, graphic: Node)

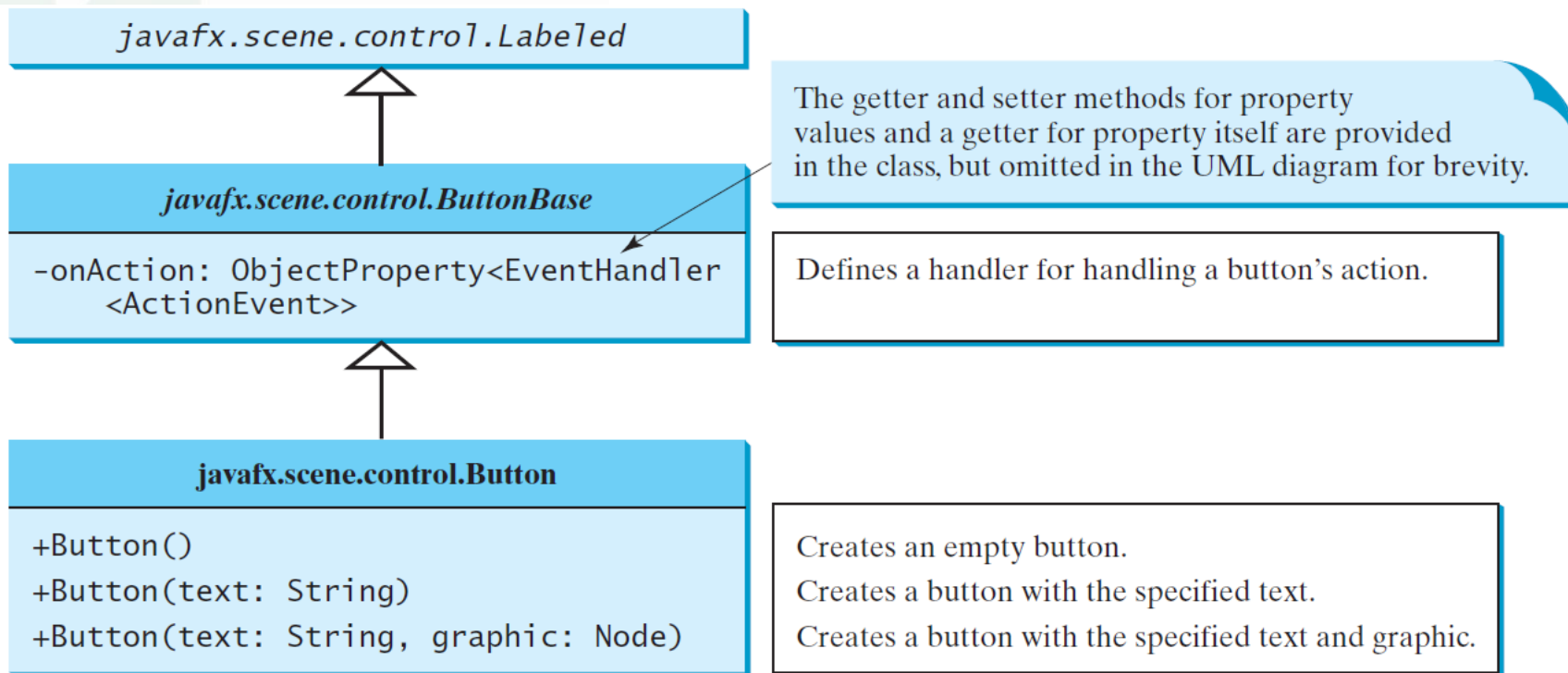
Creates an empty label.
 Creates a label with the specified text.
 Creates a label with the specified text and graphic.

LabelWithGraphic

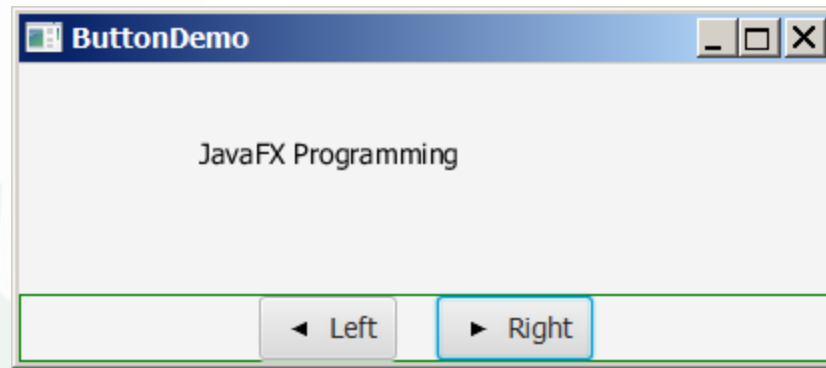
Run

ButtonBase and Button

A *button* is a control that triggers an action event when clicked. JavaFX provides regular buttons, toggle buttons, check box buttons, and radio buttons. The common features of these buttons are defined in **ButtonBase** and **Labeled** classes.



Button Example

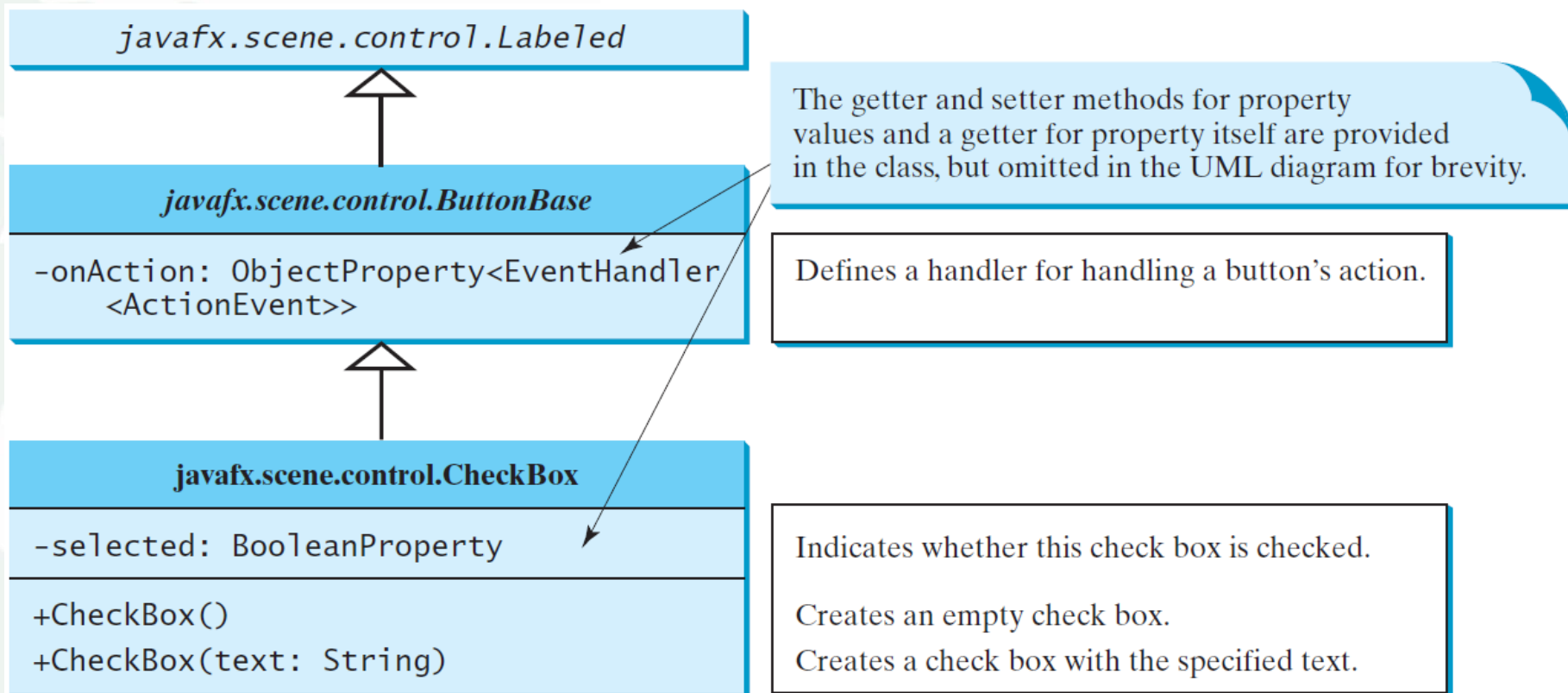


ButtonDemo

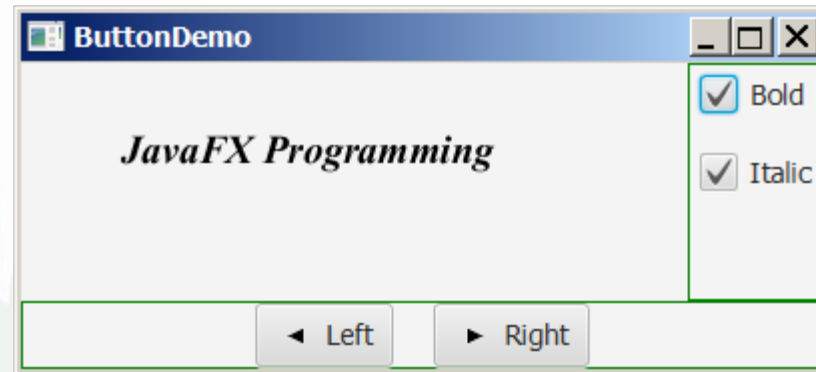
Run

CheckBox

A **CheckBox** is used for the user to make a selection. Like **Button**, **CheckBox** inherits all the properties such as **onAction**, **text**, **graphic**, **alignment**, **graphicTextGap**, **textFill**, **contentDisplay** from **ButtonBase** and **Labeled**.



CheckBox Example

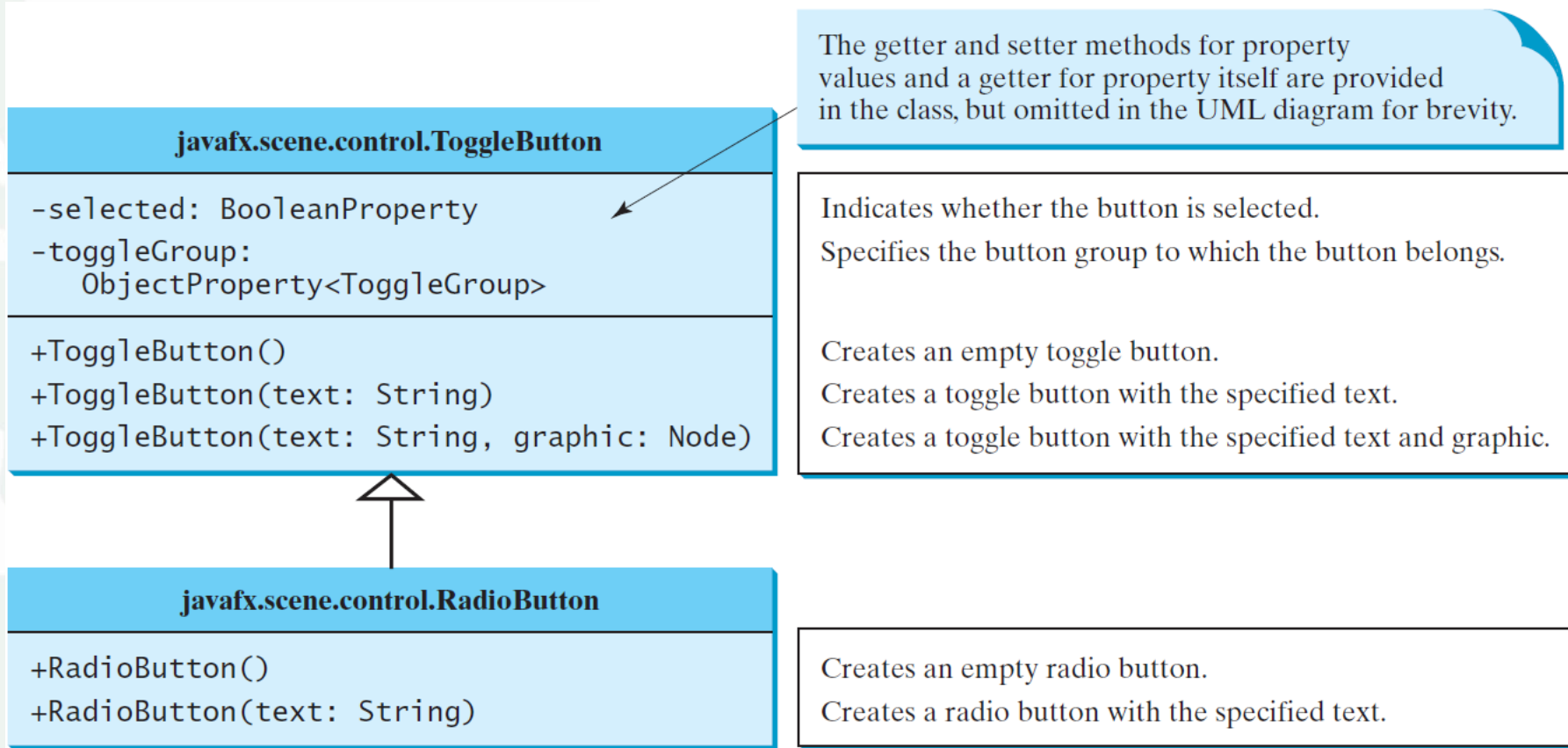


CheckBoxDemo

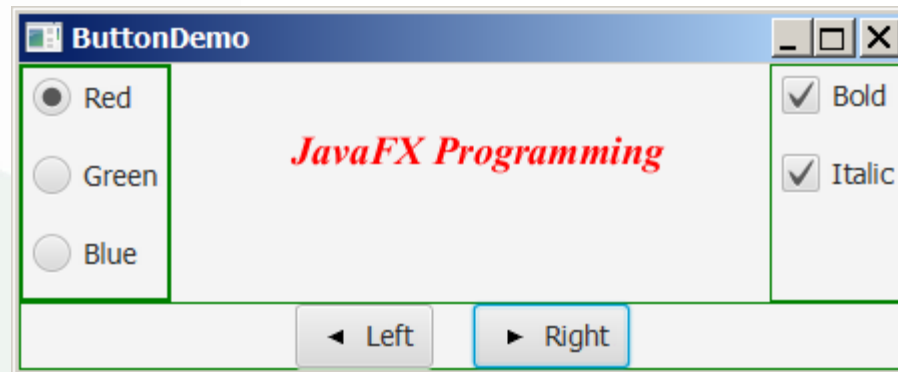
Run

RadioButton

Radio buttons, also known as *option buttons*, enable you to choose a single item from a group of choices. In appearance radio buttons resemble check boxes, but check boxes display a square that is either checked or blank, whereas radio buttons display a circle that



RadioButton Example

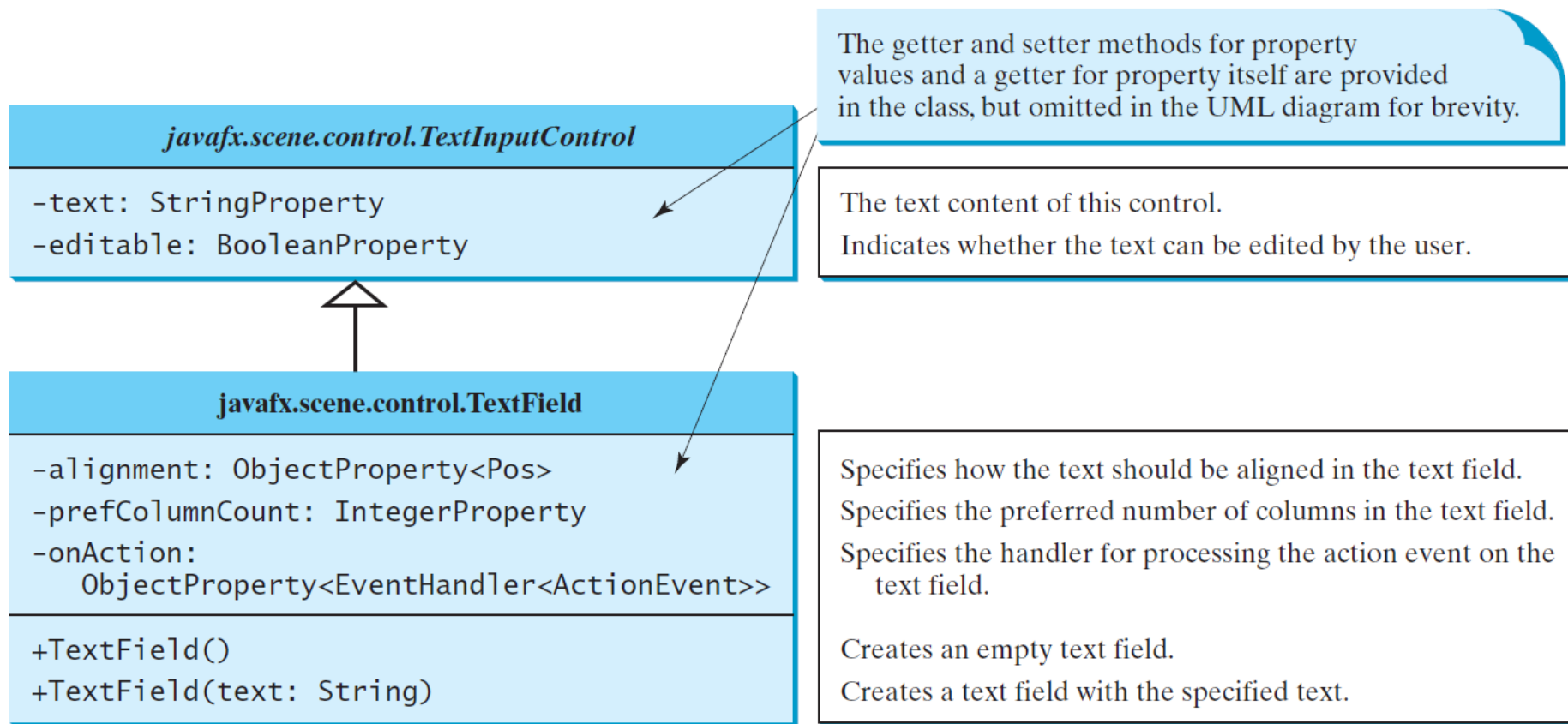


RadioButtonDemo

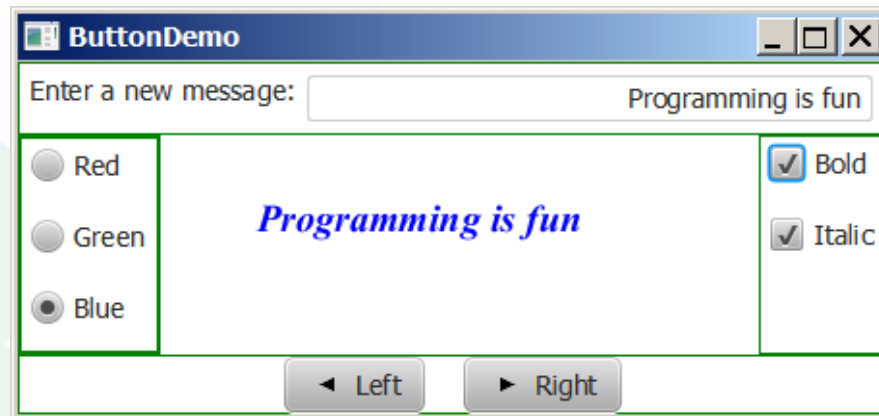
Run

TextField

A text field can be used to enter or display a string. **TextField** is a subclass of **TextInputControl**.



TextField Example

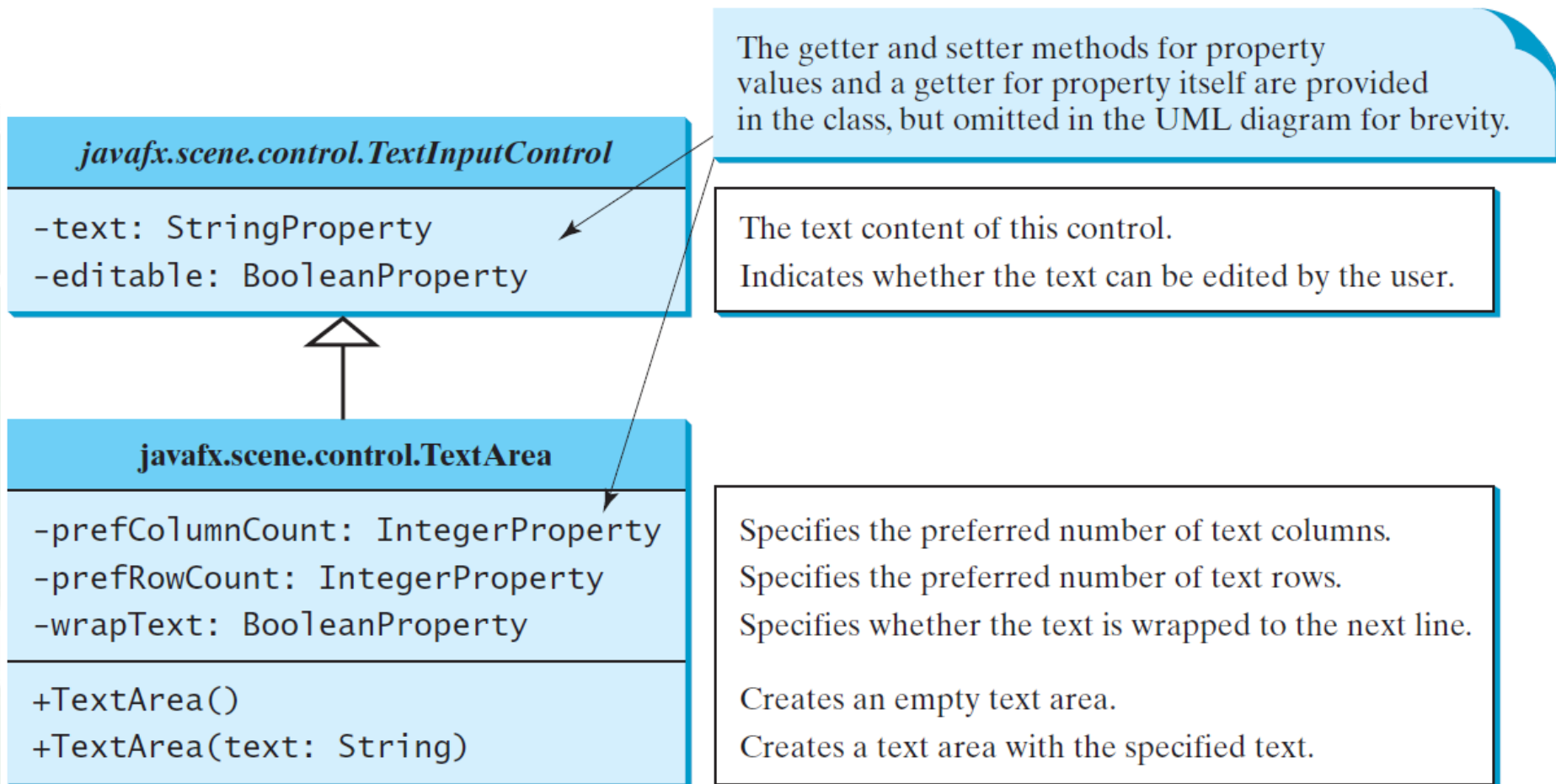


TextFieldDemo

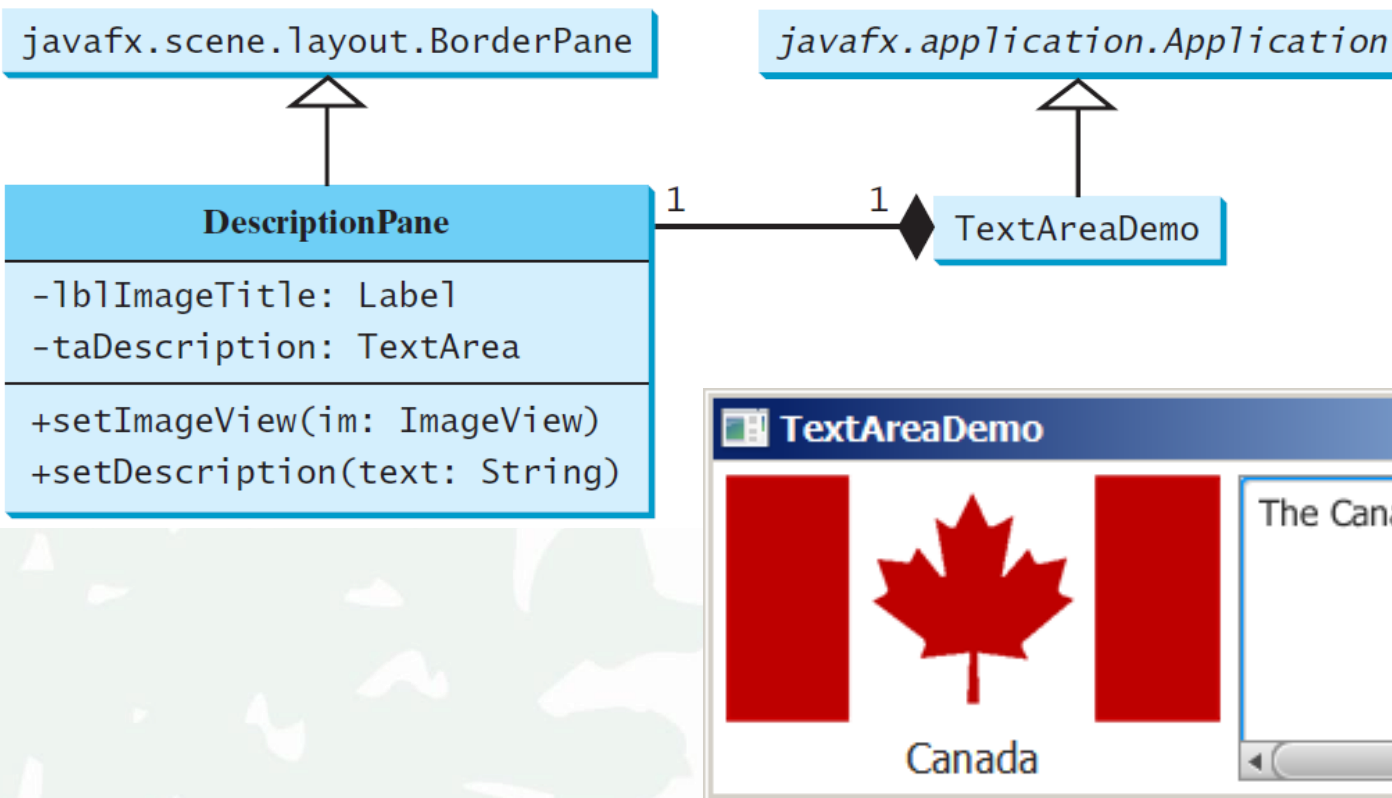
Run

TextArea

A **TextArea** enables the user to enter multiple lines of text.



TextArea Example



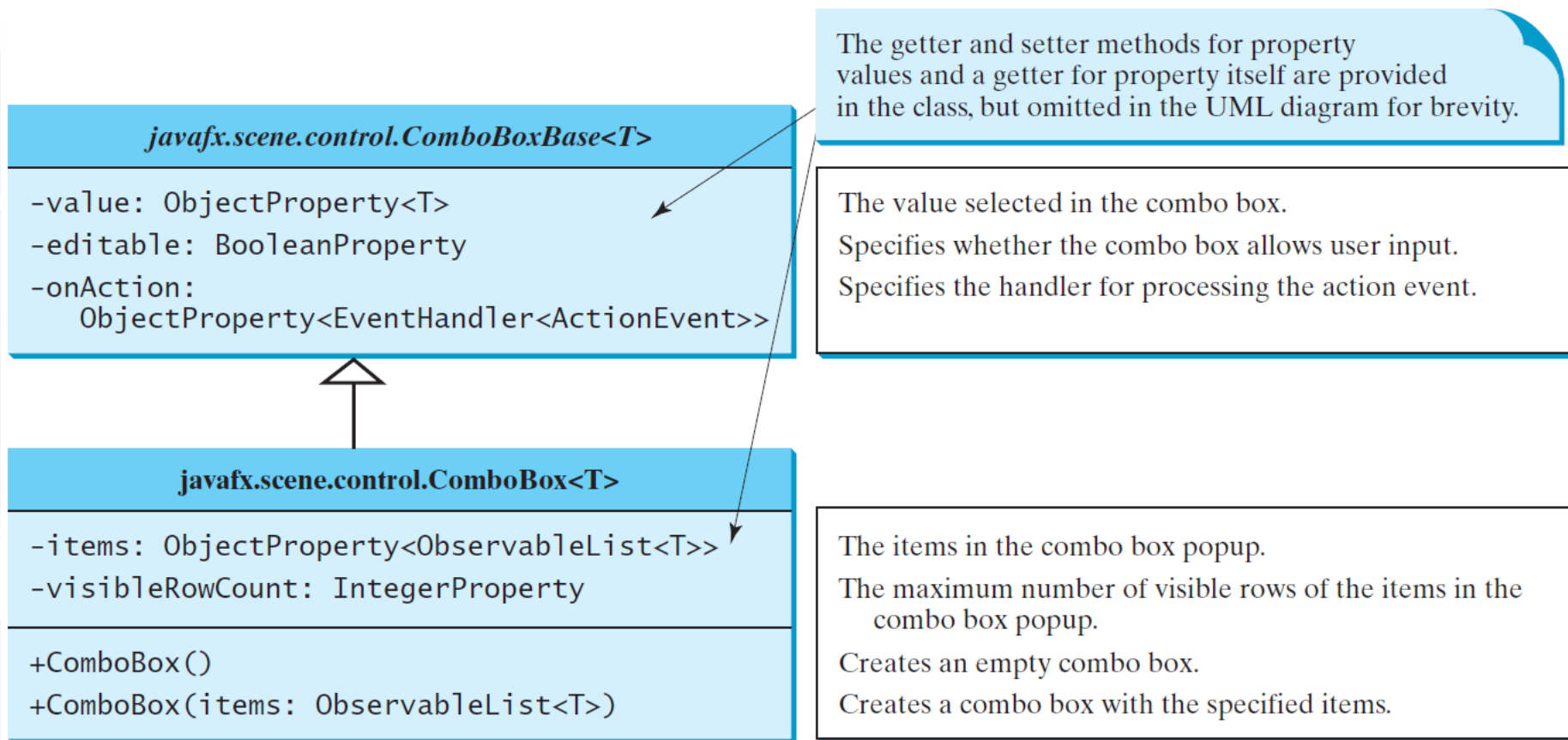
DescriptionPane

TextAreaDemo

Run

ComboBox

A combo box, also known as a choice list or drop-down list, contains a list of items from which the user can choose.



ComboBox Example

This example lets users view an image and a description of a country's flag by selecting the country from a combo box.



ComboBoxDemo

Run

List View

A *list view* is a component that performs basically the same function as a combo box, but it enables the user to choose a single value or multiple values.

javafx.scene.control.ListView<T>

```
-items: ObjectProperty<ObservableList<T>>
-orientation: BooleanProperty
-selectionModel:
  ObjectProperty<MultipleSelectionModel<T>>
```

```
+ListView()
+ListView(items: ObservableList<T>)
```

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 items in the list view.

Indicates whether the items are displayed horizontally or vertically in the list view.

Specifies how items are selected. The `SelectionModel` is also used to obtain the selected items.

Creates an empty list view.

Creates a list view with the specified items.

Example: Using ListView

This example gives a program that lets users select countries in a list and display the flags of the selected countries in the labels.



ListViewDemo

Run

ScrollBar

A *scroll bar* is a control that enables the user to select from a range of values. The scrollbar appears in two styles: *horizontal* and *vertical*.

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.control.ScrollBar

-blockIncrement: DoubleProperty
 -max: DoubleProperty
 -min: DoubleProperty
 -unitIncrement: DoubleProperty

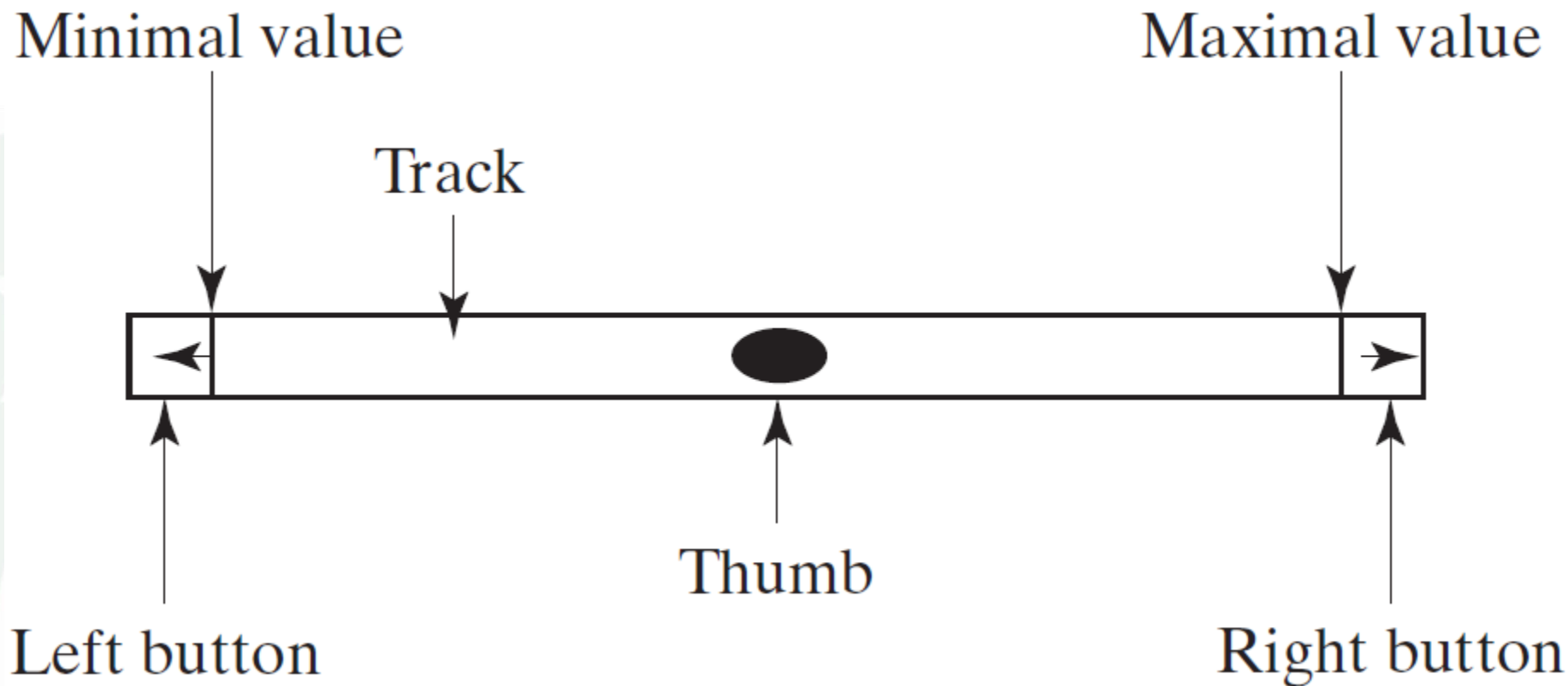
 -value: DoubleProperty
 -visibleAmount: DoubleProperty
 -orientation: ObjectProperty<Orientation>

+ScrollBar()
 +increment()
 +decrement()

The amount to adjust the scroll bar if the track of the bar is clicked (default: 10).
 The maximum value represented by this scroll bar (default: 100).
 The minimum value represented by this scroll bar (default: 0).
 The amount to adjust the scroll bar when the `increment()` and `decrement()` methods are called (default: 1).
 Current value of the scroll bar (default: 0).
 The width of the scroll bar (default: 15).
 Specifies the orientation of the scroll bar (default: HORIZONTAL).

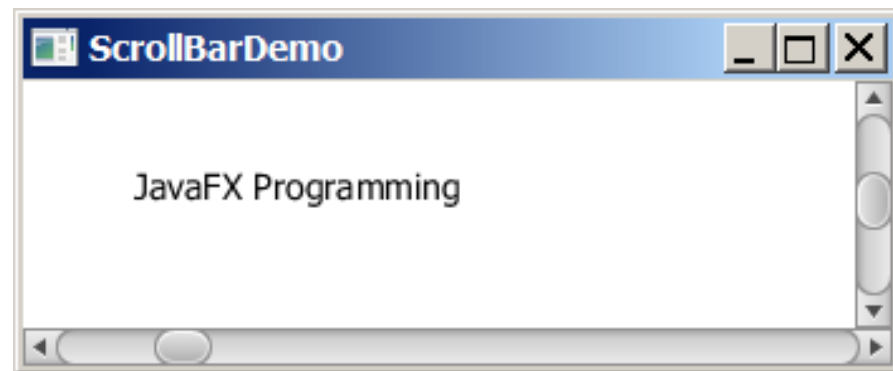
 Creates a default horizontal scroll bar.
 Increments the value of the scroll bar by `unitIncrement`.
 Decrements the value of the scroll bar by `unitIncrement`.

Scroll Bar Properties



Example: Using Scrollbars

This example uses horizontal and vertical scrollbars to control a message displayed on a panel. The horizontal scrollbar is used to move the message to the left or the right, and the vertical scrollbar to move it up and down.



ScrollBarDemo

Run

Slider

Slider is similar to ScrollBar, but Slider has more properties and can appear in many forms.

javafx.scene.control.Slider

```

-blockIncrement: DoubleProperty
-max: DoubleProperty
-min: DoubleProperty
-value: DoubleProperty
-orientation: ObjectProperty<Orientation>
-majorTickUnit: DoubleProperty
-minorTickCount: IntegerProperty
-showTickLabels: BooleanProperty
-showTickMarks: BooleanProperty

+Slider()
+Slider(min: double, max: double,
value: double)
    
```

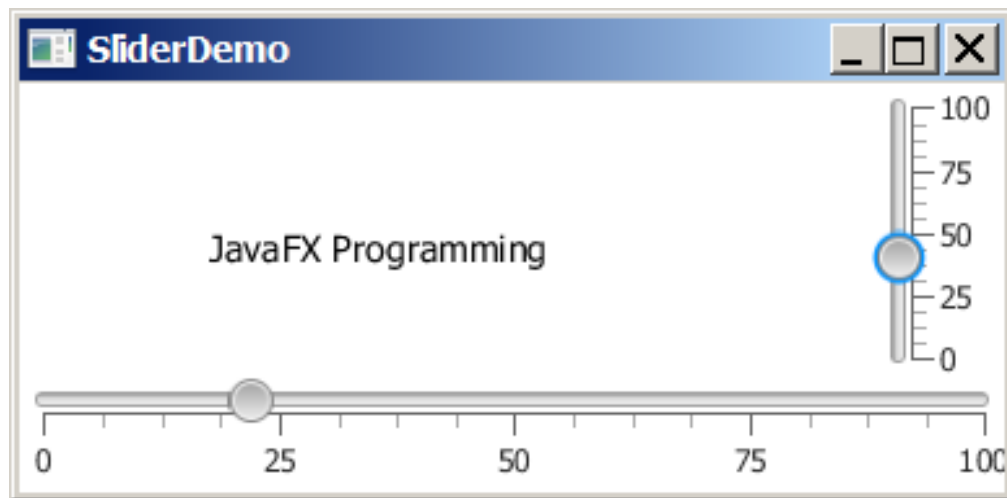
The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

The amount to adjust the slider if the track of the bar is clicked (default: 10).
 The maximum value represented by this slider (default: 100).
 The minimum value represented by this slider (default: 0).
 Current value of the slider (default: 0).
 Specifies the orientation of the slider (default: HORIZONTAL).
 The unit distance between major tick marks.
 The number of minor ticks to place between two major ticks.
 Specifies whether the labels for tick marks are shown.
 Specifies whether the tick marks are shown.

Creates a default horizontal slider.
 Creates a slider with the specified min, max, and value.

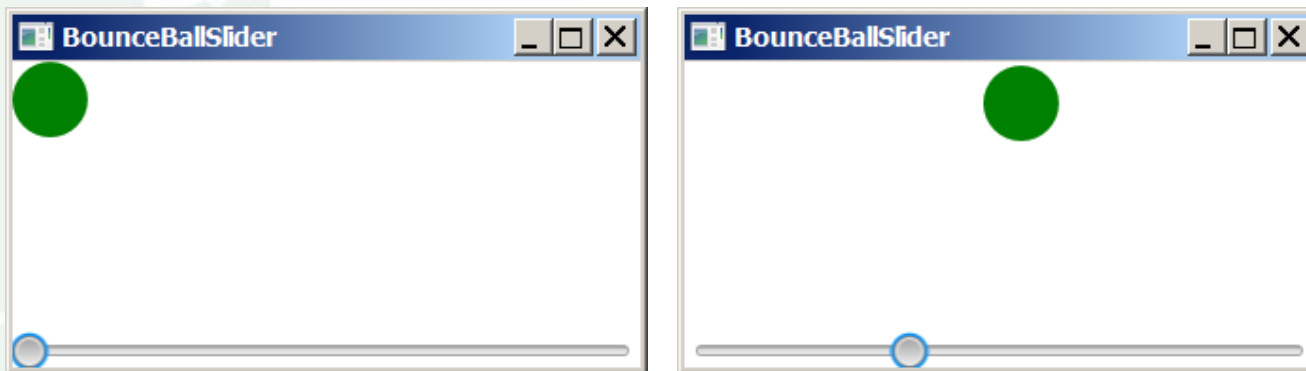
Example: Using Sliders

Rewrite the preceding program using the sliders to control a message displayed on a panel instead of using scroll bars.



Case Study: Bounce Ball

Listing 15.17 gives a program that displays a bouncing ball. You can add a slider to control the speed of the ball movement.



SliderDemo

Run