CS-202 Introduction to Object Oriented Programming

California State University, Los Angeles
Computer Science Department

Lecture 13
JavaFX Basics Part 2
Slides by Keenan Knaur

The Image and ImageView Classes

The Image and ImageView Classes

- **`javafx.scene.image.Image`** represents an image specified by a URL or filename.
  - `new Image("image/us.gif")`
  - `new Image("http://somewebsite.com/us.gif")`
- **`javafx.scene.image.ImageView`** is a node for displaying an image and can be created from an Image object

```
Image image = new Image("image/us.gif");
ImageView imageView = new ImageView(image);
```

- You can also make an **`ImageView`** directly from a file or URL:

```
ImageView imageView = new ImageView("image/us.gif");
```

The Image and ImageView Classes

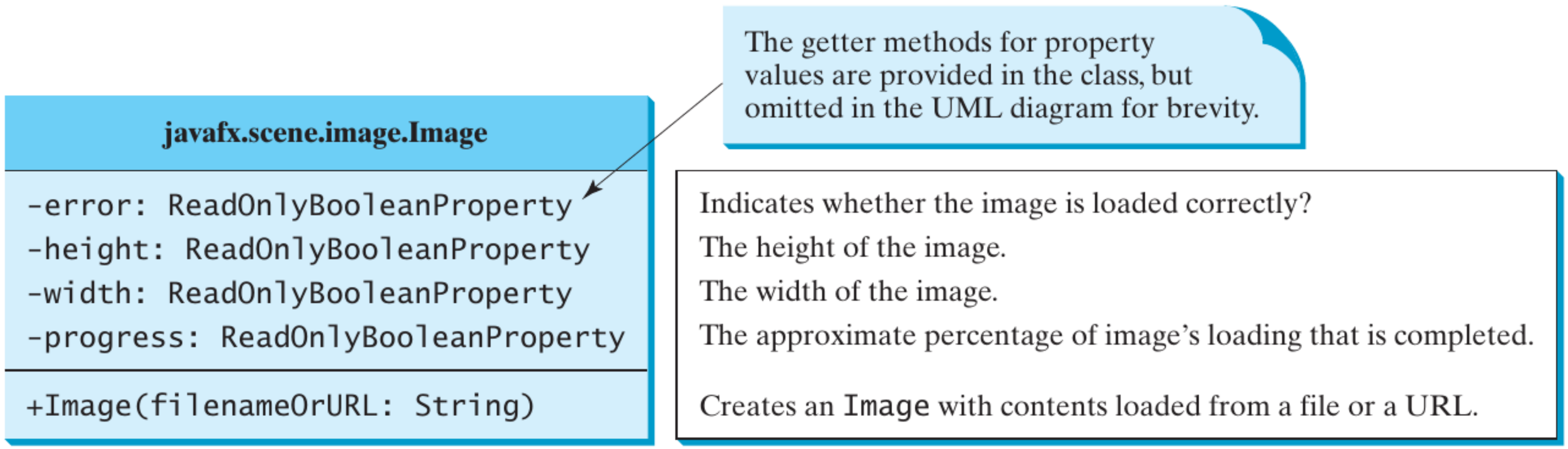


FIGURE 14.12 `Image` encapsulates information about images.

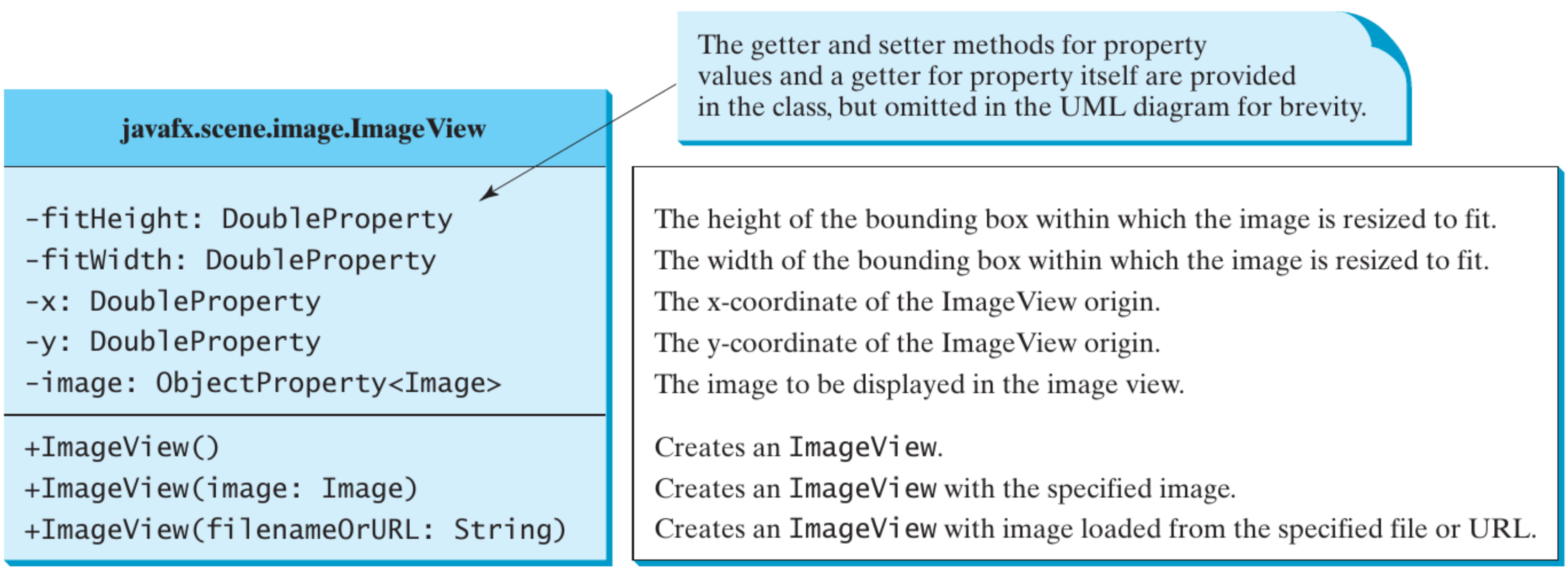


FIGURE 14.13 `ImageView` is a node for displaying an image.

The Image and ImageView Classes

- See Code: **`ShowImage.java`**

- NOTE: The image file must be in the same directory as the class file.
  - For Eclipse make sure the images are in the project folder.

Layout Panes

Layout Panes

- JavaFX provides various layout panes to automatically lay out nodes.

TABLE 14.1 Panes for Containing and Organizing Nodes

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.

Layout Panes - FlowPane

- Arranges the nodes in the pane horizontally from left to right, or vertically from top to bottom in the order that the nodes were added to the pane.
- If a row or column is filled, and new row or column is started.
- Constants:
  - **`Orientation.HORIZONTAL`** (place the nodes horizontally)
  - **`Orientation.VERTICAL`** (place the nodes vertically)
- **`FlowPane`** has data fields **`alignment`**, **`orientation`**, **`hgap`**, and **`vgap`** which are **binding properties**
- Each of the binding properties has a getter method (**`getHgap()`**) which returns its value, a setter methods (**`setHgap()`**) for setting the value, and a getter that returns the property itself (**`hGapProperty()`**)

Layout Panes - 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.

FIGURE 14.15 FlowPane lays out nodes row by row horizontally or column by column vertically.

Layout Panes - FlowPane

- See: ShowFlowPane.java

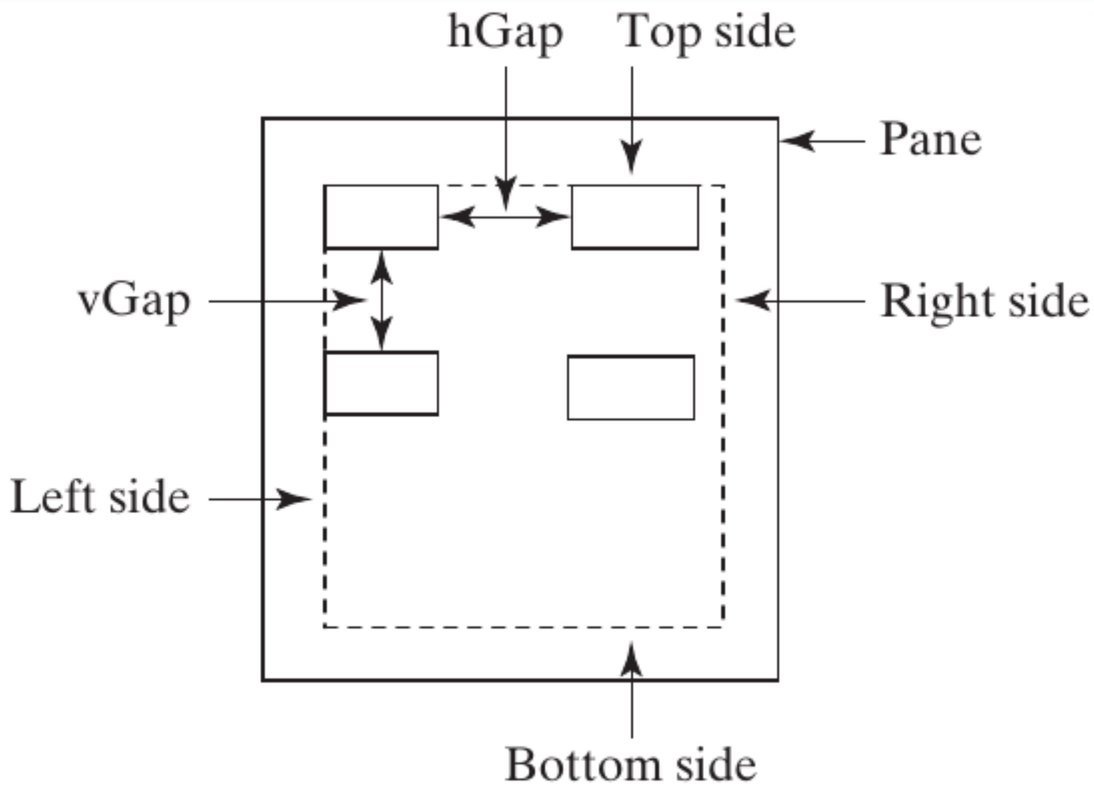


FIGURE 14.17 You can specify **hGap** and **vGap** between the nodes in a **FlowLPane**.

Layout Panes - GridPane

- Arranges nodes in a grid (matrix) formation, nodes are placed in the specified row and indicies.

- See: **`ShowGridPane.java`**

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.

FIGURE 14.18 `GridPane` lays out nodes in the specified cell in a grid.

Layout Panes - BorderPane

- Can place nodes in five regions:
  - top `setTop(node)`
  - bottom `setBottom(node)`
  - left `setLeft(node)`
  - right `setRight(node)`
  - center `setCenter(node)`
- See Code: `ShowBorderPane.java`

Layout Panes - BorderPane

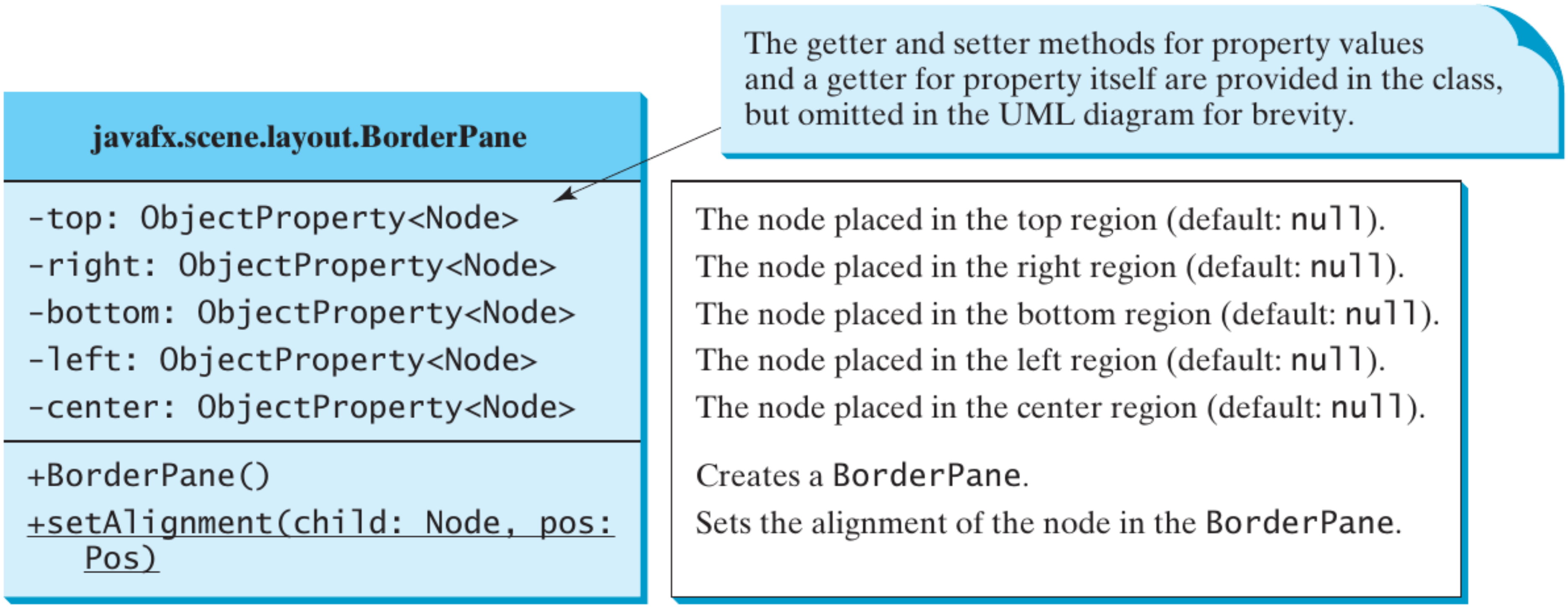


FIGURE 14.20 BorderPane places the nodes in top, bottom, left, right, and center regions.

HBox and VBox

- **`HBox`** lays out its children in a single row.
- **`VBox`** lays out its children in a single vertical column

- Recall that LayoutPanes use multiple rows and columns where as **`HBox`** and **`VBox`** use only a single column.

- See Code: **`ShowHBoxVBox.java`**

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.

FIGURE 14.22 HBox places the nodes in one row.

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.

FIGURE 14.23 VBox places the nodes in one column.

Shapes

Shapes

- JavaFX provides many ways to draw text, lines, circles, rectangles, ellipses, arcs, polygons, and polylines.

- `Shape` is an abstract base class that defines the common properties of all shapes.

Shapes - Text

- Text class defines a node that displays a string at a starting point (x, y)
- Usually placed in a pane.
  - upper-left corner of a pane is (0, 0)
  - bottom-right point is (**`pane.getWidth()`**, **`pane.getHeight()`**)
- Use **`\n`** to break a string over multiple lines.
- See Code: **`ShowText.java`**

Shapes - 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.

FIGURE 14.26 Text defines a node for displaying a text.

Shapes - Text

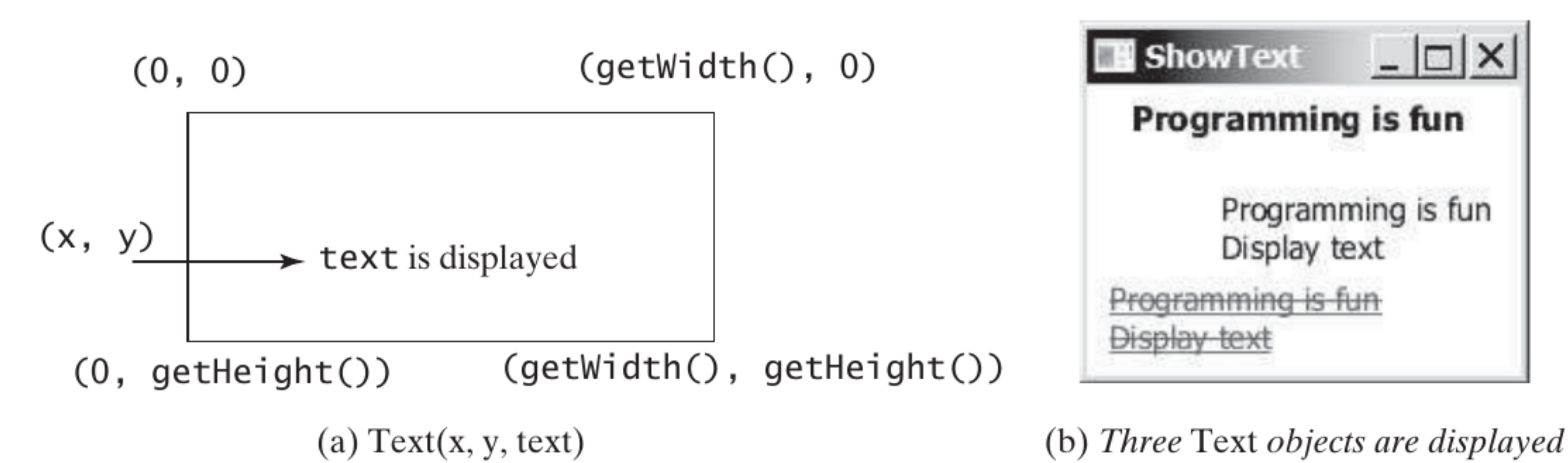


FIGURE 14.27 A Text object is created to display a text.

Shapes - Line

- A line connects two points with four parameters **`startX`**, **`startY`**, **`endX`**, and **`endY`**
- See Code: **`ShowLine.java`**

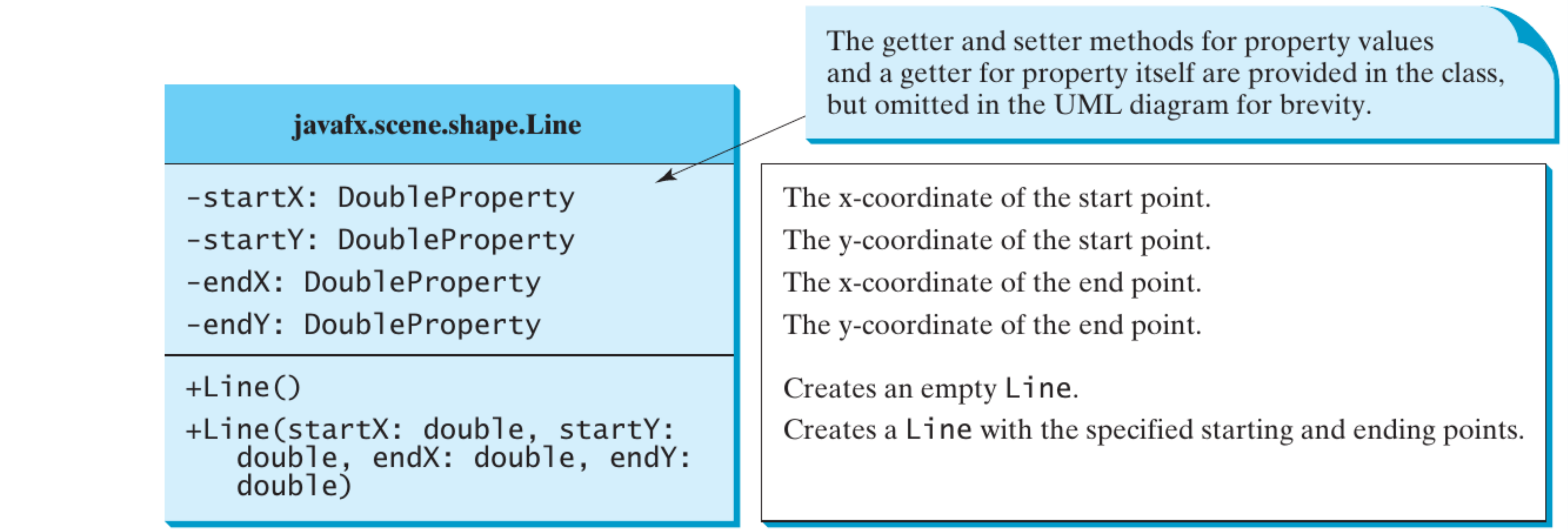
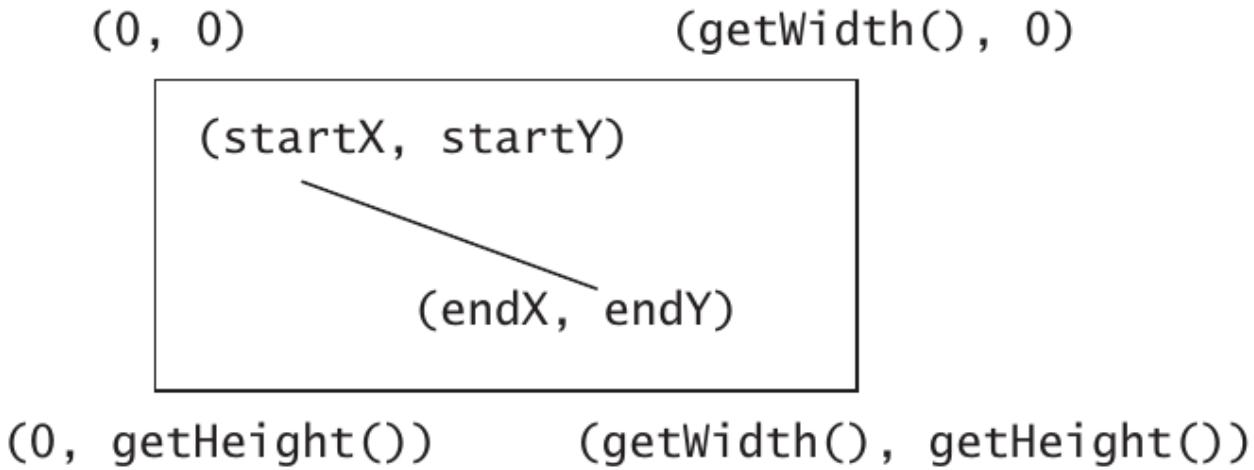
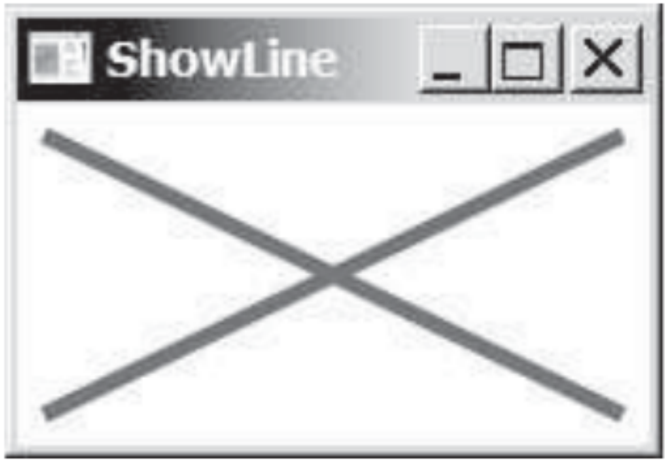


FIGURE 14.28 The `Line` class defines a line.

Shapes - Line



(a) Line(startX, startY, endX, endY)



(b) Two lines are displayed across the pane.

FIGURE 14.29 A Line object is created to display a line.

Shapes - Rectangle

- defined by parameters **`x`**, **`y`**, **`width`**, **`height`**, **`arcWidth`**, and **`arcHeight`**
- upper-left corner is point at (**`x`**, **`y`**) and parameter **`aw`** (**`arcWidth`**) the horizontal diameter of the arcs at the corner, and **`ah`** (**`arcHeight`**) the vertical diameter of the arcs at the corner
- See Code: **`ShowRectangle.java`**

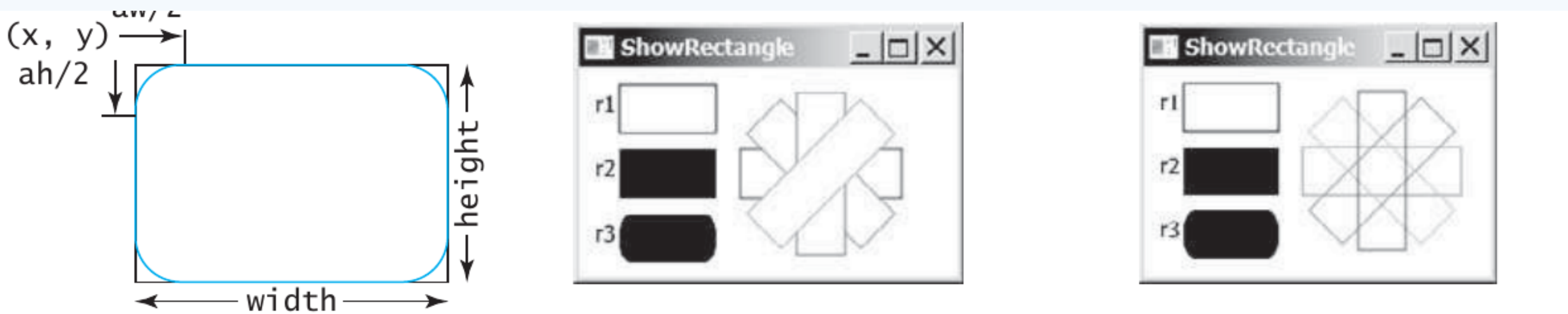
Shapes - Rectangle

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.

FIGURE 14.30 The Rectangle class defines a rectangle.

Shapes - Rectangle



(a) Rectangle(x, y, w, h) (b) Multiple rectangles are displayed (c) Transparent rectangles are displayed

FIGURE 14.31 A Rectangle object is created to display a rectangle.

Shapes – Circle and Ellipse

- Circles are defined by **`centerX`**, **`centerY`**, and **`radius`**
- An Ellipse is defined by **`centerX`**, **`centerY`**, **`radiusX`**, and **`radiusY`**
- See Code: **`ShowEllipse.java`** and **`ShowCircle.java`**

Shapes – Circle and 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.Circle	
-centerX: DoubleProperty	The x-coordinate of the center of the circle (default 0).
-centerY: DoubleProperty	The y-coordinate of the center of the circle (default 0).
-radius: DoubleProperty	The radius of the circle (default: 0).
+Circle()	Creates an empty Circle.
+Circle(x: double, y: double)	Creates a Circle with the specified center.
+Circle(x: double, y: double, radius: double)	Creates a Circle with the specified center and radius.

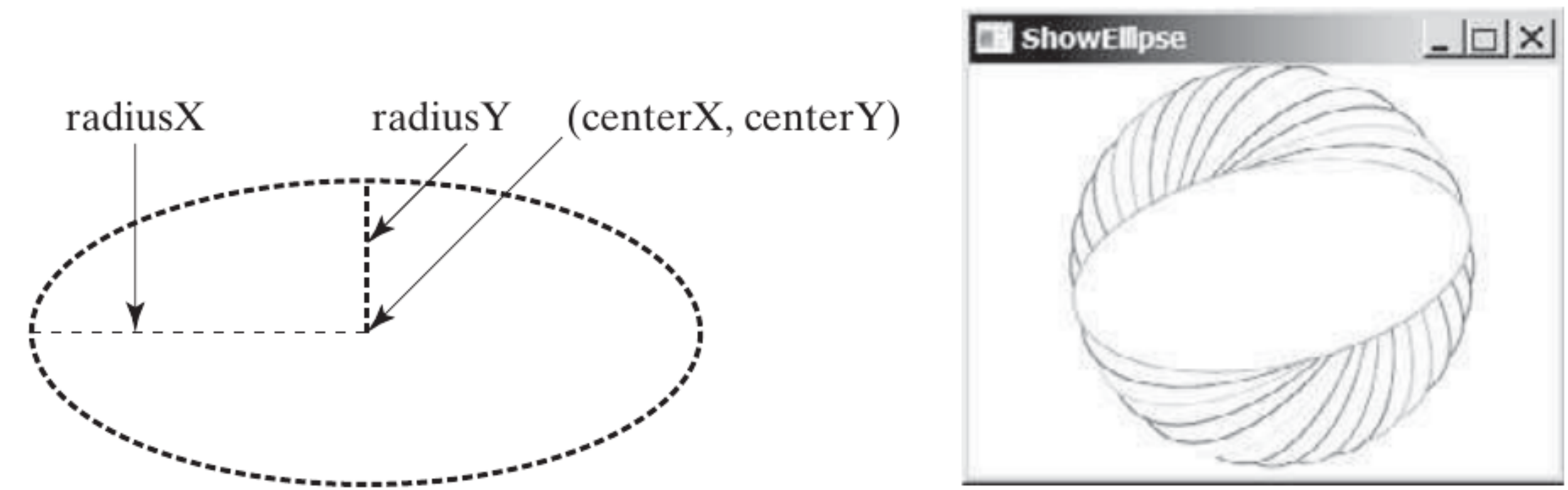
FIGURE 14.32 The Circle class defines circles.

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.

FIGURE 14.33 The Ellipse class defines ellipses.

Shapes – Circle and Ellipse



(a) `Ellipse(centerX, centerY, radiusX, radiusY)`

(b) Multiple ellipses are displayed.

FIGURE 14.34 An `Ellipse` object is created to display an ellipse.

Shapes – Arc

- Part of an ellipse
- defined by **`centerX`**, **`centerY`**, **`radiusX`**, **`radiusY`**, **`startAngle`**, **`length`**, and an arc type (**`ArcType.OPEN, ArchType.CHORD, or ArcType.ROUND`**)
- Angles are measured in degrees and follow the normal math conventions (0 is in the easterly direction, positive angles go counterclockwise from the easterly direction)
- See Code: **`ShowArc.java`**

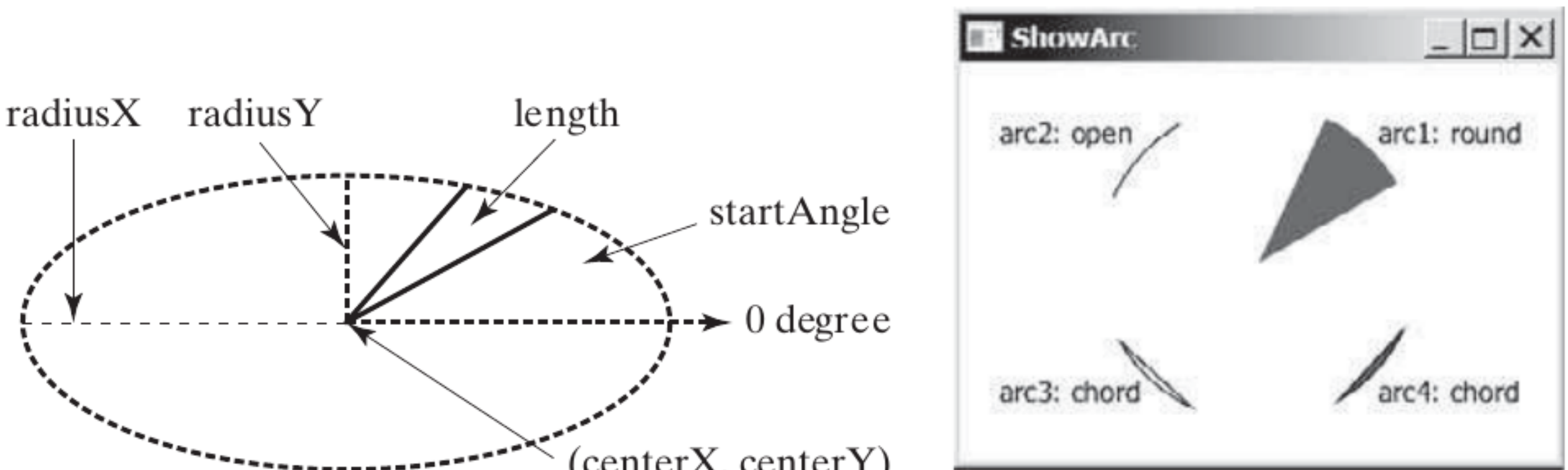
Shapes – Arc

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.

FIGURE 14.35 The Arc class defines an arc.

Shapes – Arc

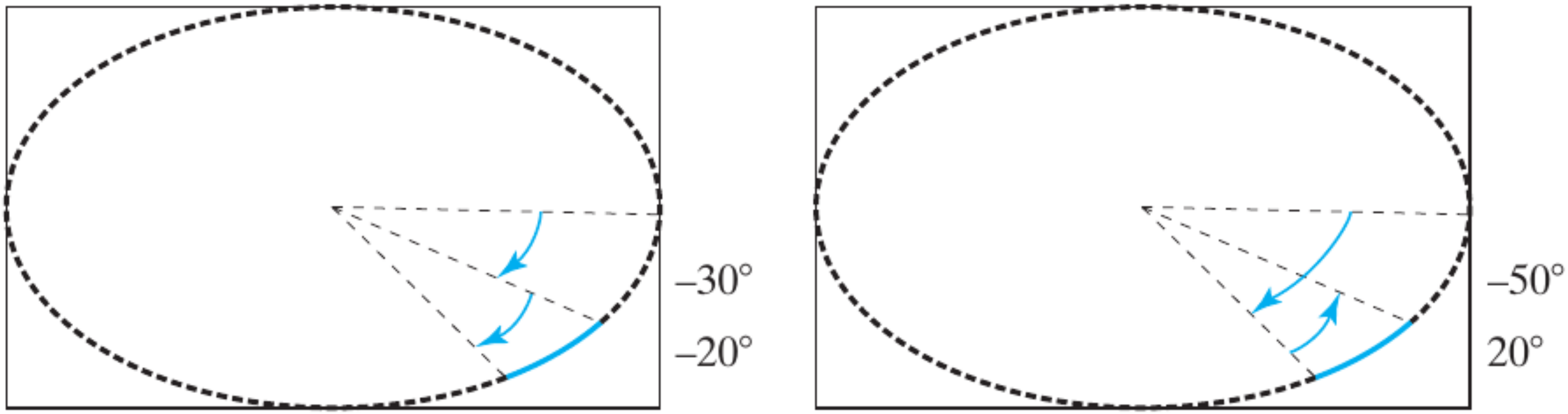


(a) `Arc(centerX, centerY, radiusX, radiusY, startAngle, length)`

(b) Multiple ellipses are displayed

FIGURE 14.36 An **Arc** object is created to display an arc.

Shapes – Arc



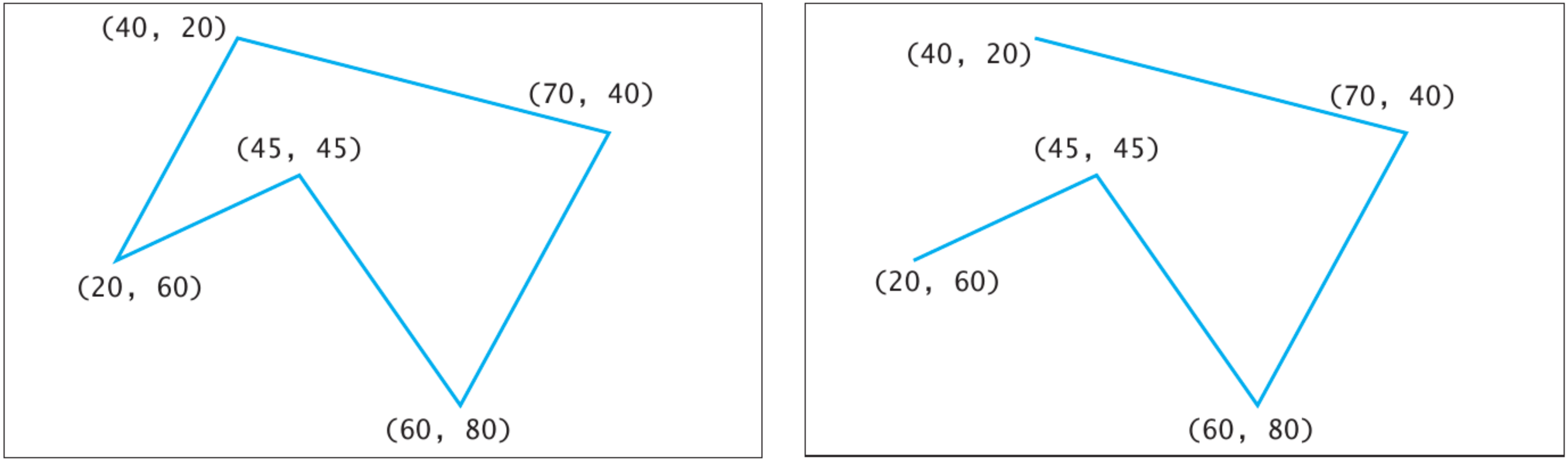
(a) Negative starting angle –30° and negative spanning angle –20°

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

FIGURE 14.37 Angles may be negative.

Shapes – Polygon and Polyline

- **`Polygon`** defines a polygon that connects a sequence of points.
- **`Polyline`** is similar to Polygon but the class is not automatically closed.
- See Code: **`ShowPolygon.java`**



(a) Polygon (b) Polyline

FIGURE 14.38 `Polygon` is closed and `Polyline` is not closed.

Shapes – Polygon and Polyline

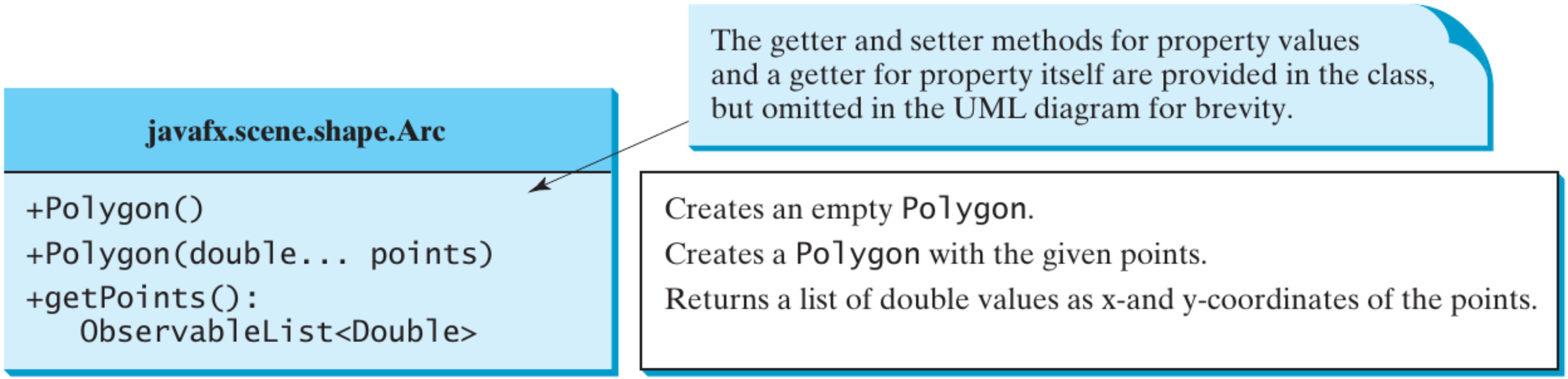


FIGURE 14.39 Polygon defines a polygon.

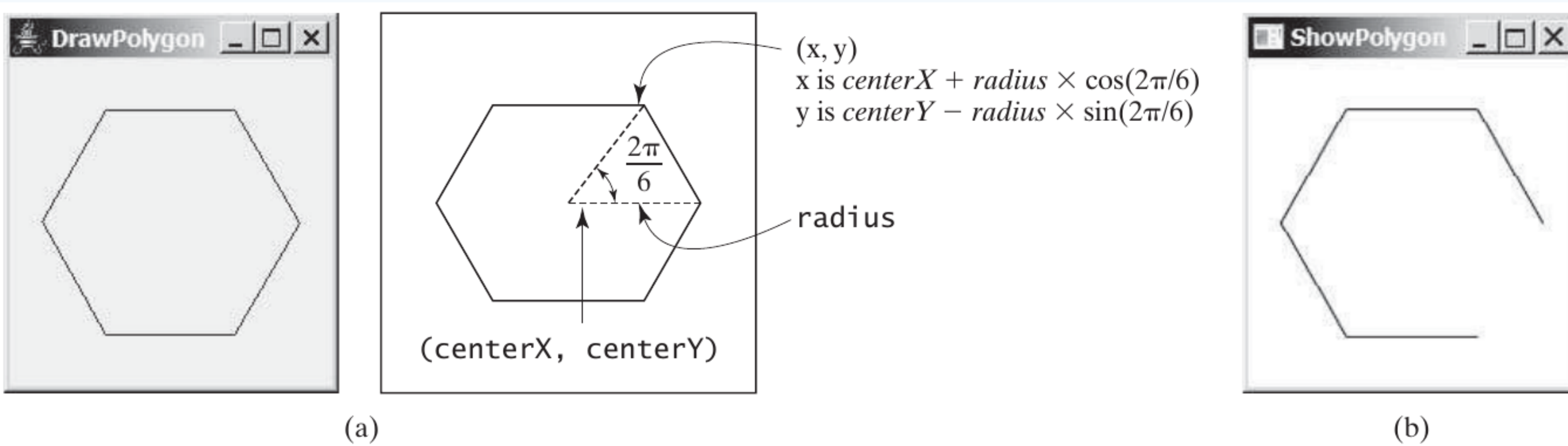


FIGURE 14.40 (a) A Polygon is displayed. (b) A Polyline is displayed.