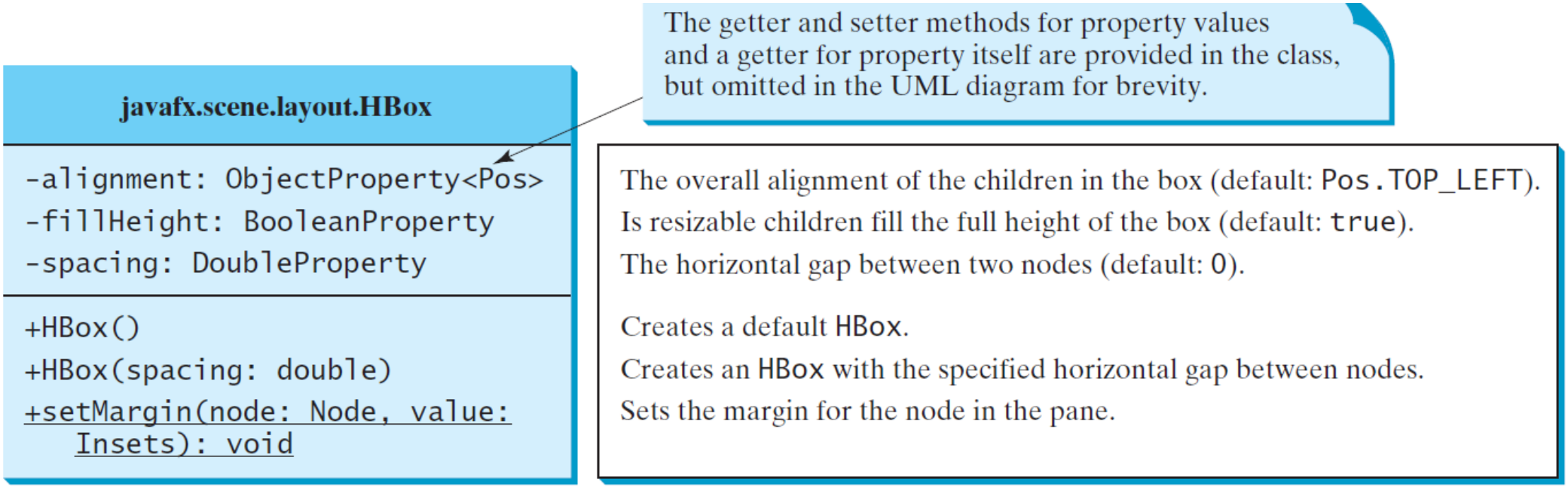
Chapter 14 JavaFX Basics

HBox and Vbox
Shapes

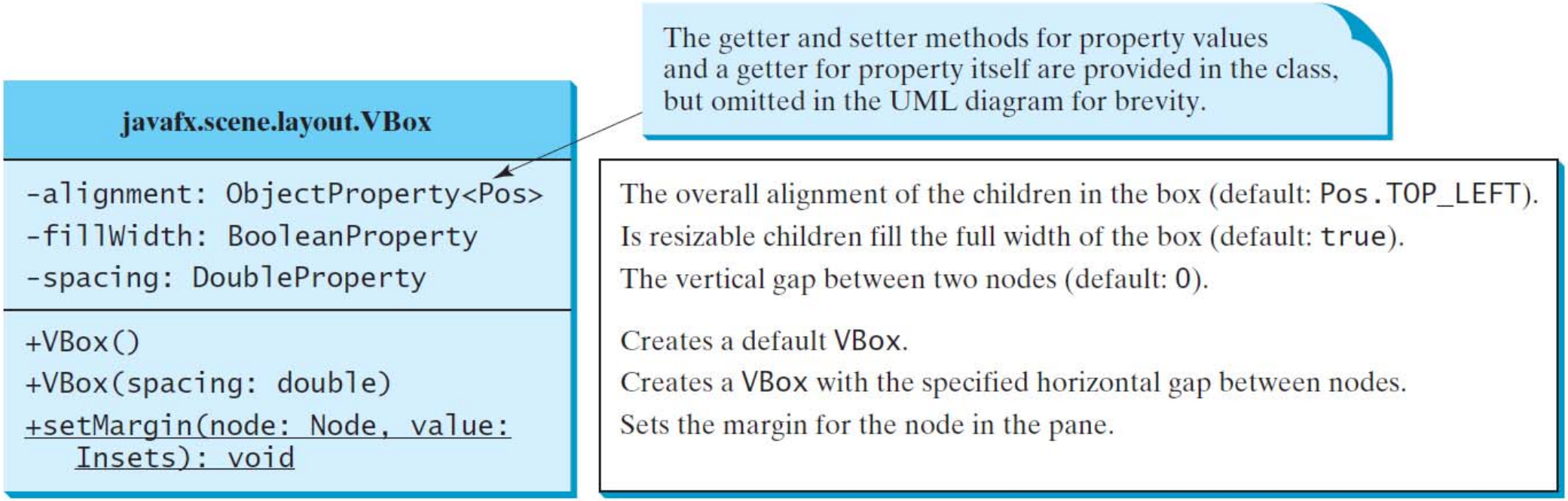
Hbox

- An **HBox** lays out its children in a single horizontal row.



VBox

- A **VBox** lays out its children in a single vertical column.



Hbox and Vbox Example

```
public class ShowHBoxVBox 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(getHBox());
    pane.setLeft(getVBox());

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

  private HBox getHBox() {
    HBox hBox = new HBox(15);
    hBox.setPadding(new Insets(15, 15, 15, 15));
    hBox.setStyle("-fx-background-color: gold");
    hBox.getChildren().add(new Button("Computer Science"));
    hBox.getChildren().add(new Button("Chemistry"));
    ImageView imageView = new ImageView(new Image("image/us.gif"));
    hBox.getChildren().add(imageView);
    return hBox;
  }
```

Hbox and Vbox Example (continued)

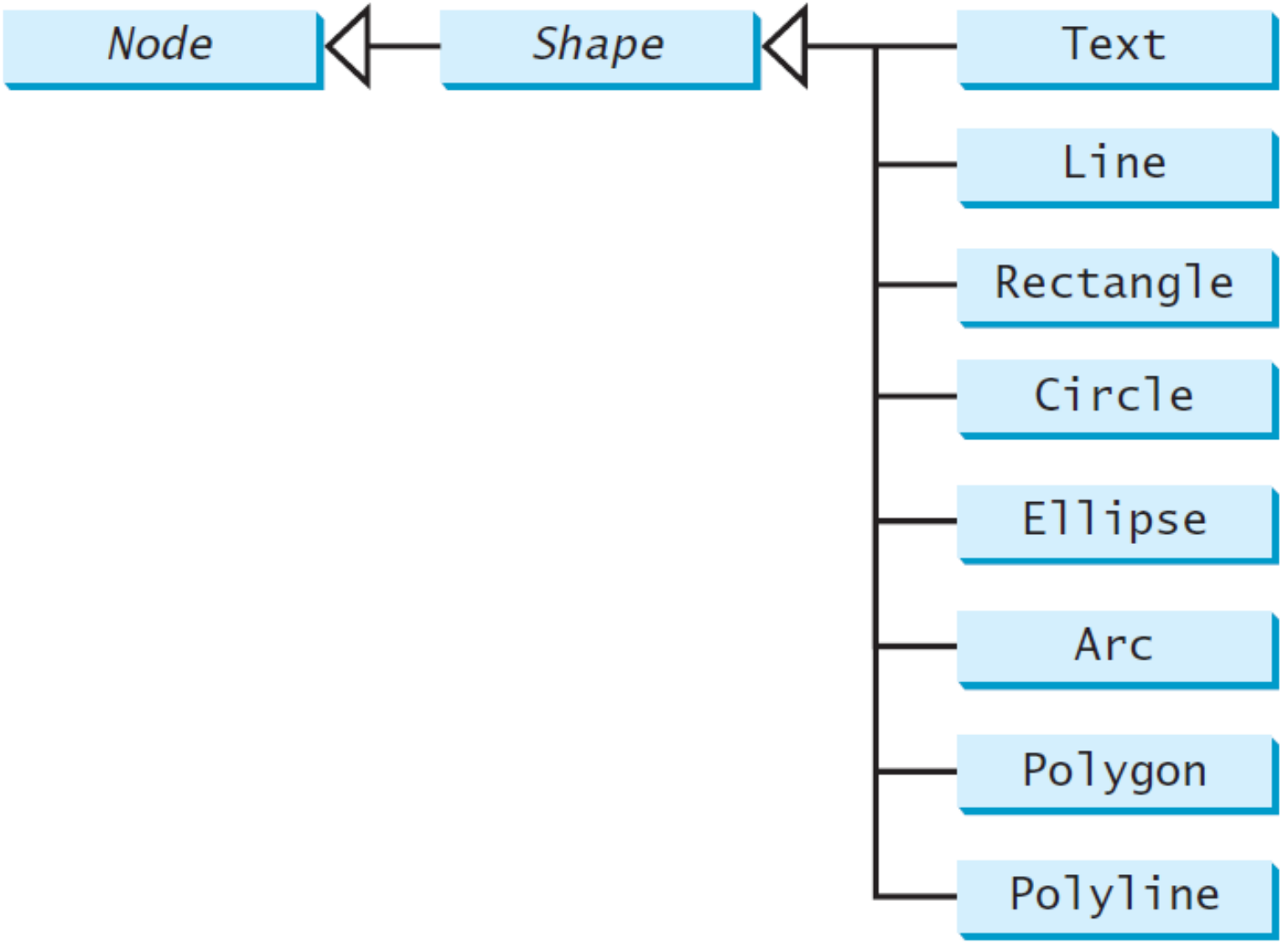
```
private VBox getVBox() {
  VBox vBox = new VBox(15);
  vBox.setPadding(new Insets(15, 5, 5, 5));
  vBox.getChildren().add(new Label("Courses"));

  Label[] courses = {new Label("CSCI 1301"), new Label("CSCI 1302"),
      new Label("CSCI 2410"), new Label("CSCI 3720")};

  for (Label course: courses) {
    VBox.setMargin(course, new Insets(0, 0, 0, 15));
    vBox.getChildren().add(course);
  }

  return vBox;
  }
}
```

Shapes

- JavaFX provides many shape classes for drawing texts, lines, circles, rectangles, ellipses, arcs, polygons, and polylines.
- The **Shape** class is the abstract base class that defines the common properties for all shapes.
- Among them are the **fill**, **stroke**, and **strokeWidth** properties.
- The **fill** property specifies a color that fills the interior of a shape.
- The **stroke** property specifies a color that is used to draw the outline of a shape.
- The **strokeWidth** property specifies the width of the outline of a shape.

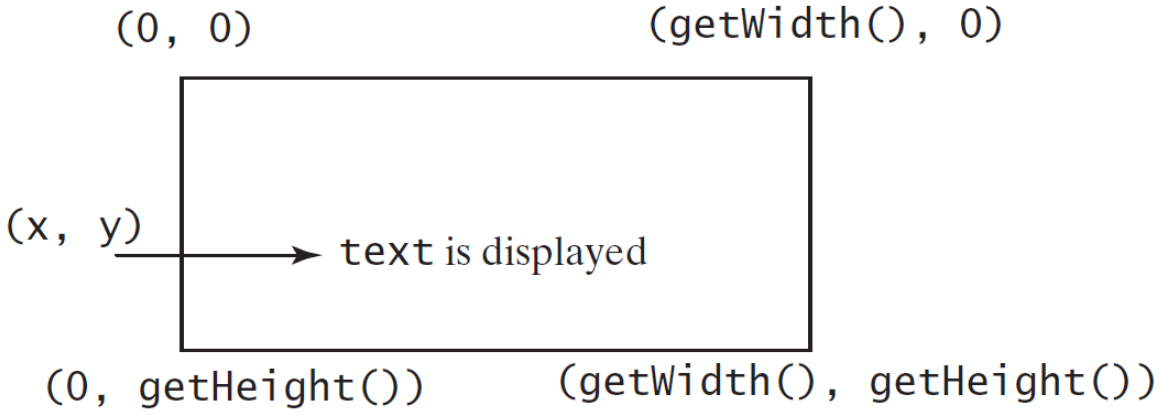
Shapes

Subclasses of abstract class **Shape**



Text

- The **Text** class defines a node that displays a string at a starting point (**x**, **y**).
- A **Text** object is usually placed in a pane.
- The pane's upper-left corner point is (**0**, **0**) and the bottom-right point is (**pane.getWidth**(), **pane.getHeight**()).
- A string may be displayed in multiple lines separated by **n**.



(a) Text(x, y, text)

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.

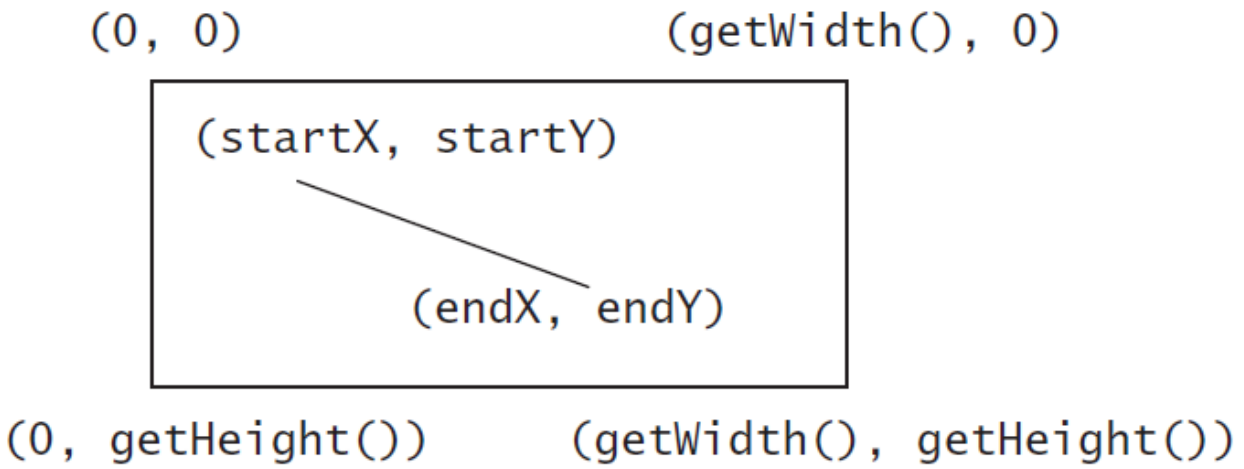
Text Example

```
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
  }
}
```

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

Line

- The **Line** class is used to connect two points with four parameters startX, startY, endX, and endY.



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

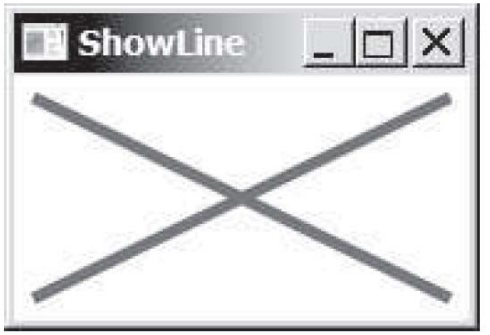
```
-startX: DoubleProperty
-startY: DoubleProperty
-endX: DoubleProperty
-endY: DoubleProperty
```

```
+Line()
+Line(startX: double, startY:
   double, endX: double, endY:
   double)
```

The x-coordinate of the start point.
The y-coordinate of the start point.
The x-coordinate of the end point.
The y-coordinate of the end point.

Creates an empty `Line`.
Creates a `Line` with the specified starting and ending points.

Line Example



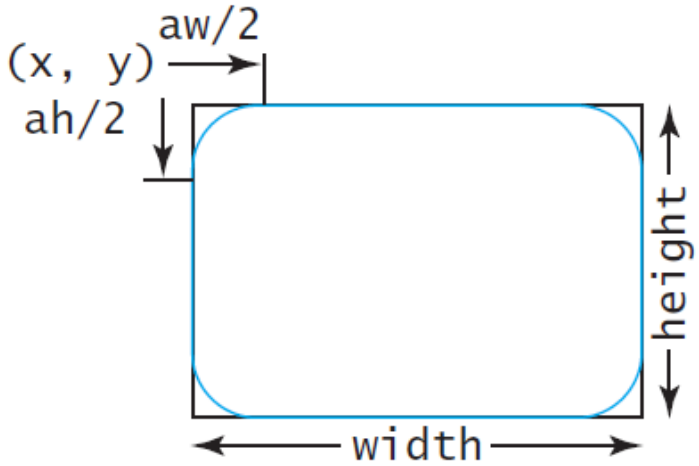
(b) Two lines are displayed across the pane.

```
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

- A rectangle is defined by the parameters **x**, **y**, **width**, **height**, **arcWidth**, and **arcHeight**.
- The rectangle's upper-left corner point is at (x, y)
- Parameter **arcWidth** is the horizontal diameter of the arcs at the corner, and **arcHeight** is the vertical diameter of the arcs at the corner.
- In class Rectangle, the fill color is black and the stroke color is white by default.
- If the method **setFill(null**) of **Rectangle** is called, the rectangle is not filled with a color.

Rectangle



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

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()
+Rectanlge(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).

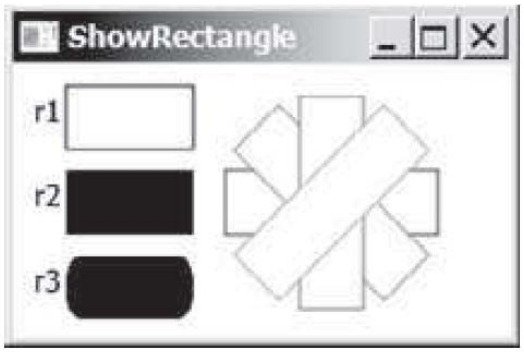
The `arcWidth` of the rectangle (default: 0). `arcWidth` is the horizontal diameter of the arcs at the corner (see Figure 14.31a).

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`.

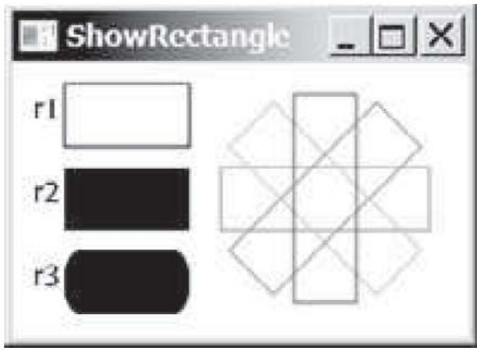
Creates a `Rectangle` with the specified upper-left corner point, width, and height.

Rectangle Example



(b) Multiple rectangles are displayed

```
r.setFill(null);
```

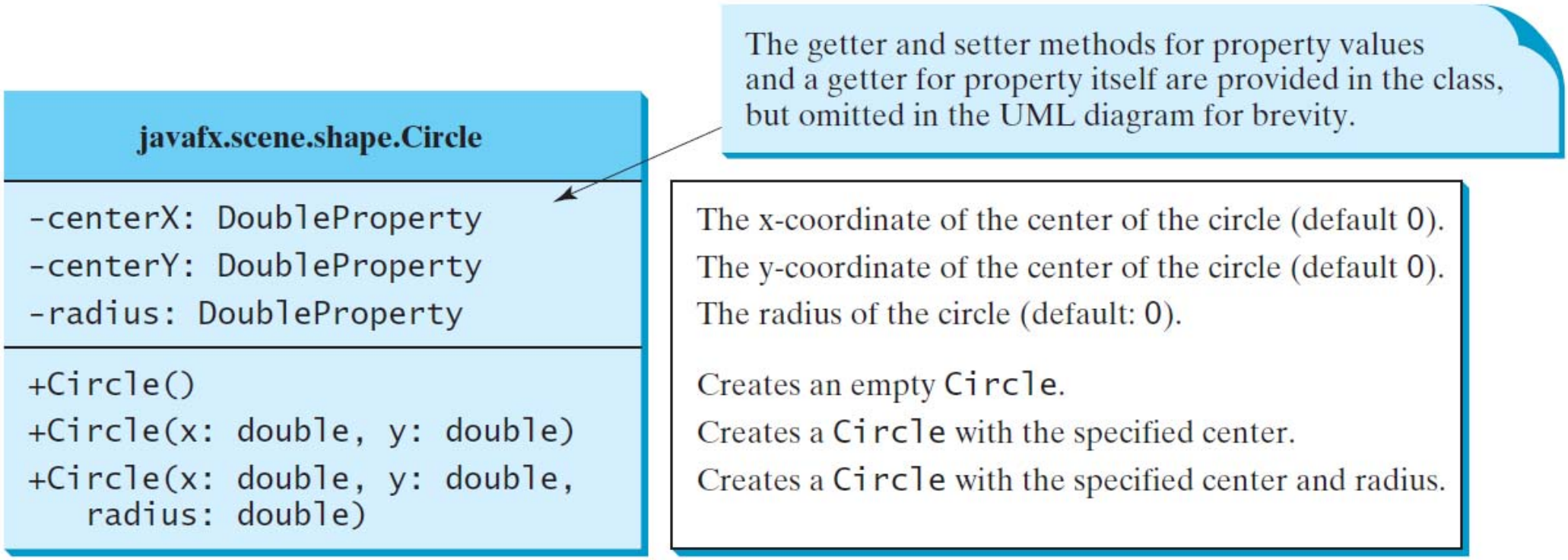


(c) Transparent rectangles are displayed

```
Pane pane = new Pane();

// Create rectangles and add to pane
Rectangle r1 = new Rectangle(25, 10, 60, 30);
r1.setStroke(Color.BLACK);
r1.setFill(Color.WHITE);
pane.getChildren().add(new Text(10, 27, "r1"));
pane.getChildren().add(r1);

Rectangle r2 = new Rectangle(25, 50, 60, 30);
pane.getChildren().add(new Text(10, 67, "r2"));
pane.getChildren().add(r2);

Rectangle r3 = new Rectangle(25, 90, 60, 30);
r3.setArcWidth(15);
r3.setArcHeight(25);
pane.getChildren().add(new Text(10, 107, "r3"));
pane.getChildren().add(r3);

for (int i = 0; i < 4; i++) {
  Rectangle r = new Rectangle(100, 50, 100, 30);
  r.setRotate(i * 360 / 8);
  r.setStroke(Color.color(Math.random(), Math.random(),
      Math.random()));
  r.setFill(Color.WHITE);
  pane.getChildren().add(r);
}

// Create a scene and place it in the stage
Scene scene = new Scene(pane, 250, 150);
primaryStage.setTitle("ShowRectangle"); // Set the stage title
primaryStage.setScene(scene); // Place the scene in the stage
primaryStage.show(); // Display the stage
  }
}
```

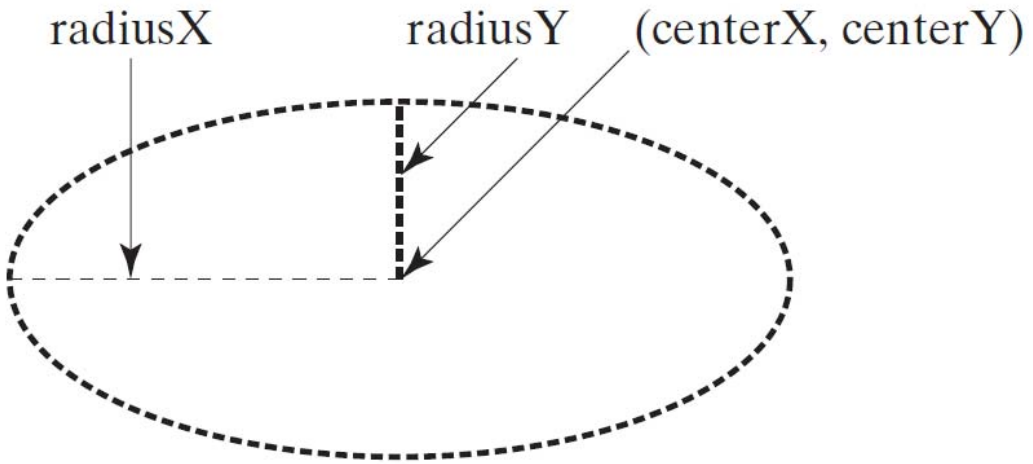
Circle

A circle is defined by its parameters **centerX**, **centerY**, and **radius**.



Ellipse

An ellipse is defined by its parameters **centerX**, **centerY**, **radiusX**, and **radiusY**



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

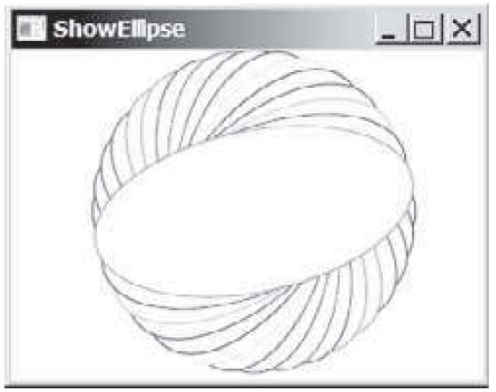
```
-centerX: DoubleProperty
-centerY: DoubleProperty
-radiusX: DoubleProperty
-radiusY: DoubleProperty
```

```
+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.

Ellipse Example



(b) Multiple ellipses are displayed.

```
public void start(Stage primaryStage) {
  // Create a pane
  Pane pane = new Pane();

  for (int i = 0; i < 16; i++) {
    // Create an ellipse and add it to pane
    Ellipse e1 = new Ellipse(150, 100, 100, 50);
    e1.setStroke(Color.color(Math.random(), Math.random(),
      Math.random()));
    e1.setFill(Color.WHITE);
    e1.setRotate(i * 180 / 16);
    pane.getChildren().add(e1);
  }

  // Create a scene and place it in the stage
  Scene scene = new Scene(pane, 300, 200);
  primaryStage.setTitle("ShowEllipse"); // Set the stage title
  primaryStage.setScene(scene); // Place the scene in the stage
  primaryStage.show(); // Display the stage
}
```