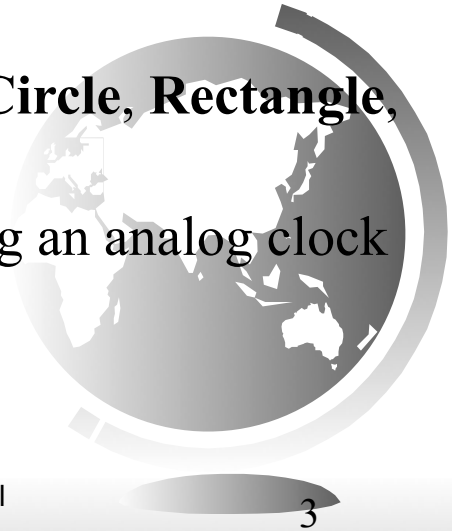
Chapter 14 JavaFX Basics

Motivations

JavaFX is a new framework for developing Java GUI programs. The JavaFX API is an excellent example of how the object-oriented principle is applied. This chapter serves two purposes. First, it presents the basics of JavaFX programming. Second, it uses JavaFX to demonstrate OOP. Specifically, this chapter introduces the framework of JavaFX and discusses JavaFX GUI components and their relationships.

Objectives

- To distinguish between JavaFX, Swing, and AWT (§14.2).
- To write a simple JavaFX program and understand the relationship among stages, scenes, and nodes (§14.3).
- To create user interfaces using panes, UI controls, and shapes (§14.4).
- To use binding properties to synchronise property values (§14.5).
- To use the common properties **style** and **rotate** for nodes (§14.6).
- To create colors using the **Color** class (§14.7).
- To create fonts using the **Font** class (§14.8).
- To create images using the **Image** class and to create image views using the **ImageView** class (§14.9).
- To layout nodes using **Pane**, **StackPane**, **FlowPane**, **GridPane**, **BorderPane**, **HBox**, and **VBox** (§14.10).
- To display text using the **Text** class and create shapes using **Line**, **Circle**, **Rectangle**, **Ellipse**, **Arc**, **Polygon**, and **Polyline** (§14.11).
- To develop the reusable GUI components **ClockPane** for displaying an analog clock (§14.12).

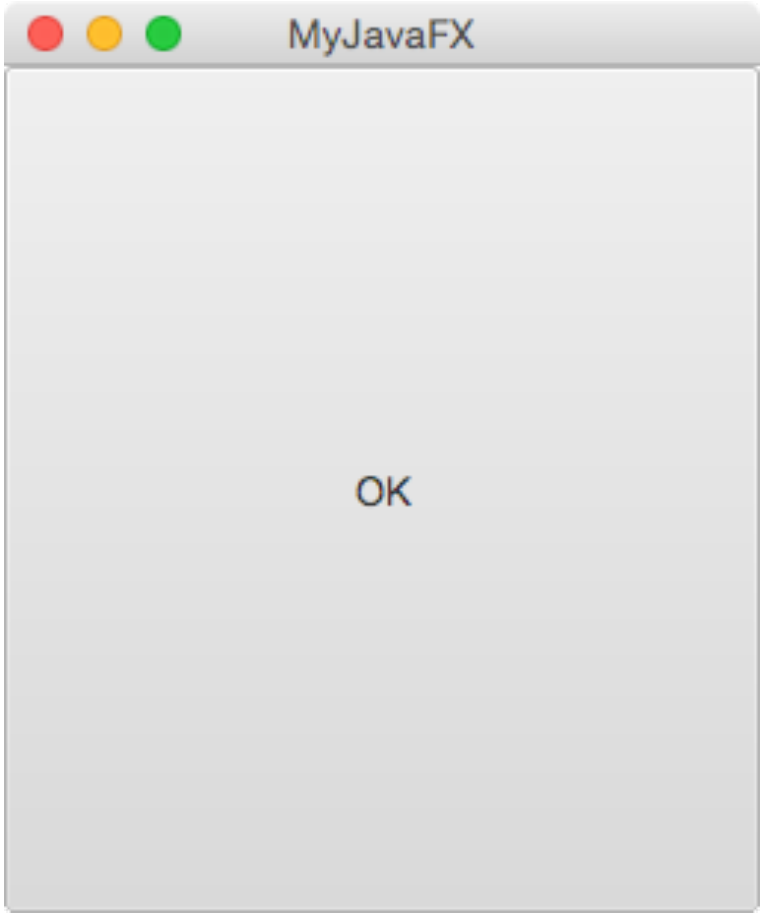


JavaFX vs Swing and AWT

Swing and AWT are replaced by the JavaFX platform for developing rich Internet applications.

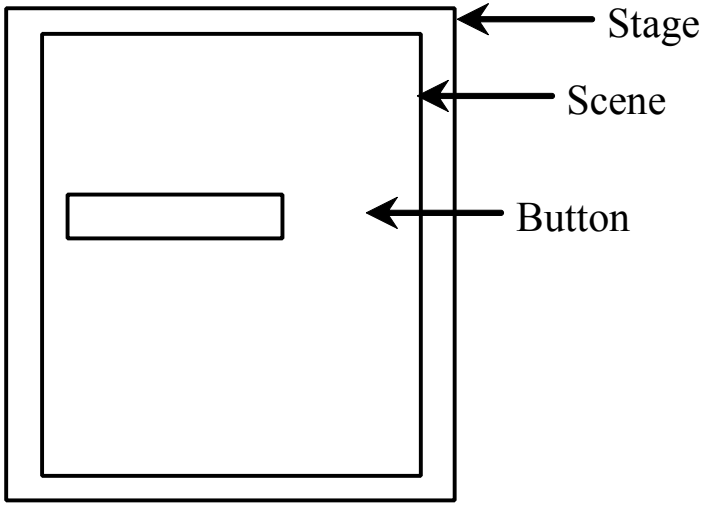
When Java was introduced, the GUI classes were bundled in a library known as the *Abstract Windows Toolkit (AWT)*. AWT is fine for developing simple graphical user interfaces, but not for developing comprehensive GUI projects. In addition, AWT is prone to platform-specific bugs. The AWT user-interface components were replaced by a more robust, versatile, and flexible library known as *Swing components*. Swing components are painted directly on canvases using Java code. Swing components depend less on the target platform and use less of the native GUI resource. With the release of Java 8, Swing is replaced by a completely new GUI platform known as *JavaFX*.

```
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);
  }
}
```



Basic Structure of JavaFX

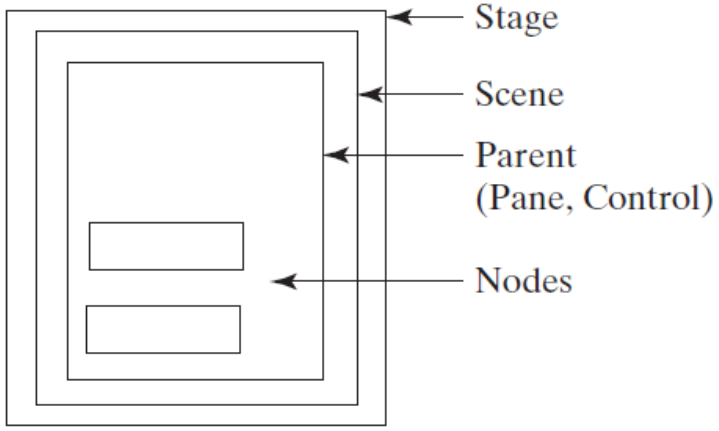
- Application
- Override the start(Stage) method
- Stage, Scene, and Nodes



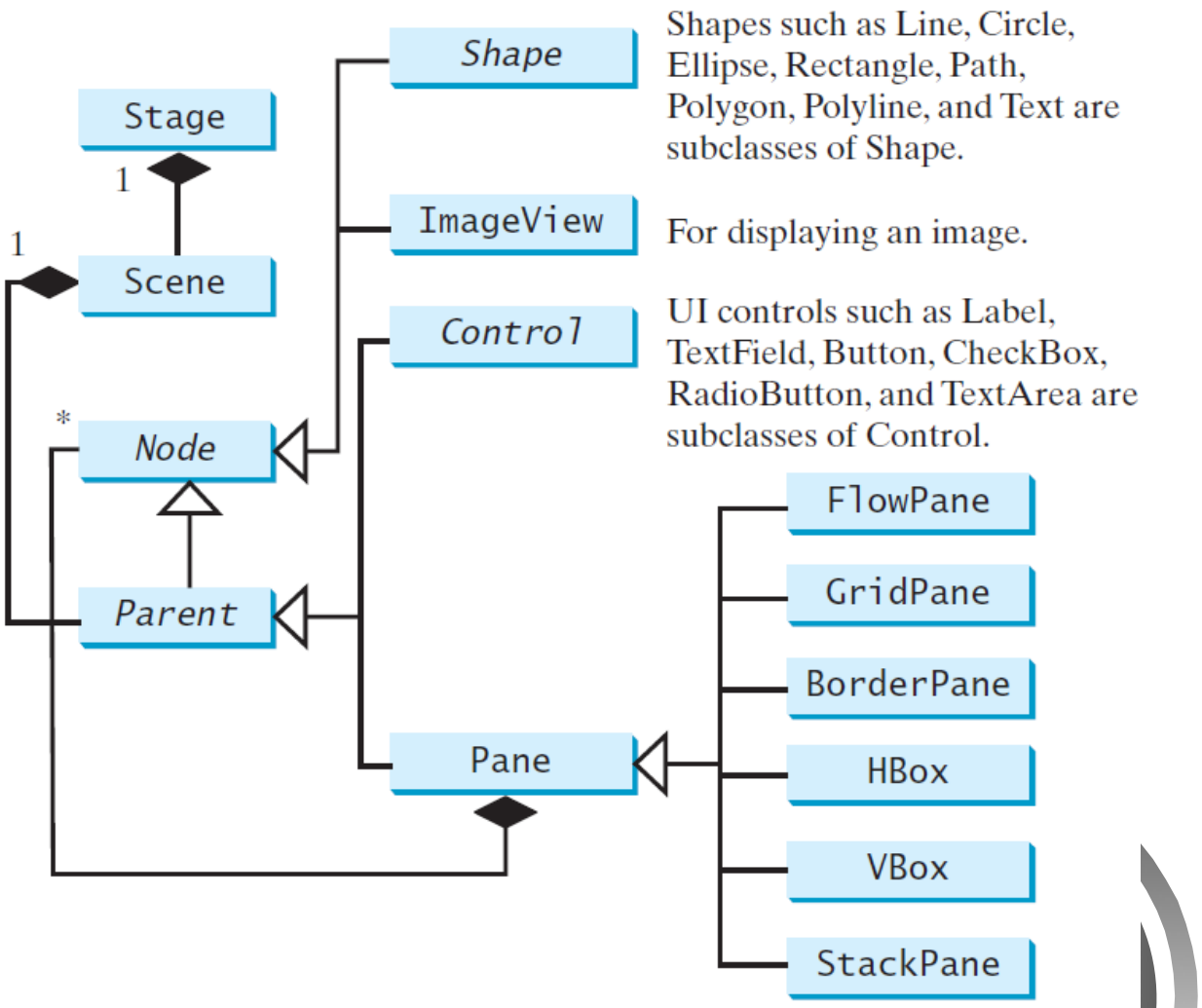
MyJavaFX

MultipleStageDemo

Panes, UI Controls, and Shapes



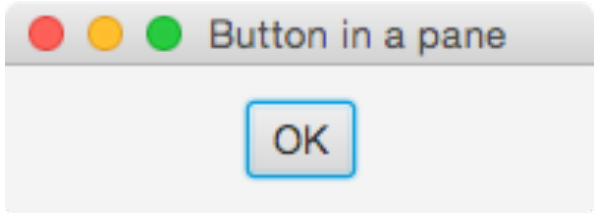
(a)



(b)

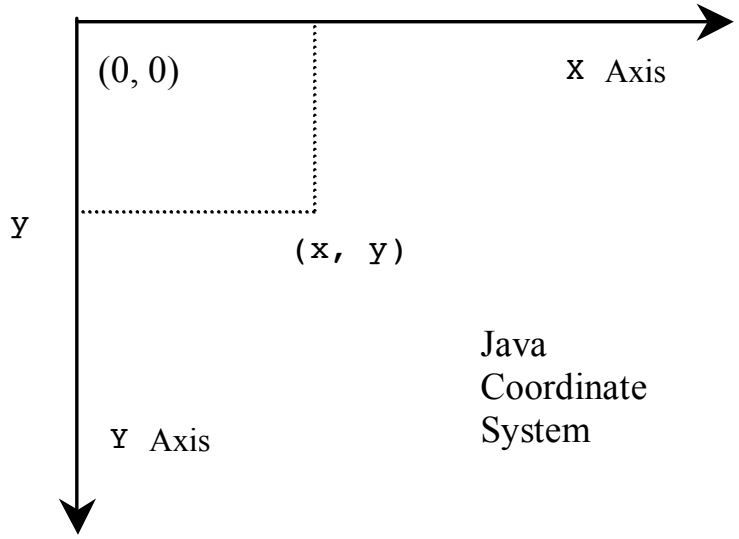
ButtonInPane

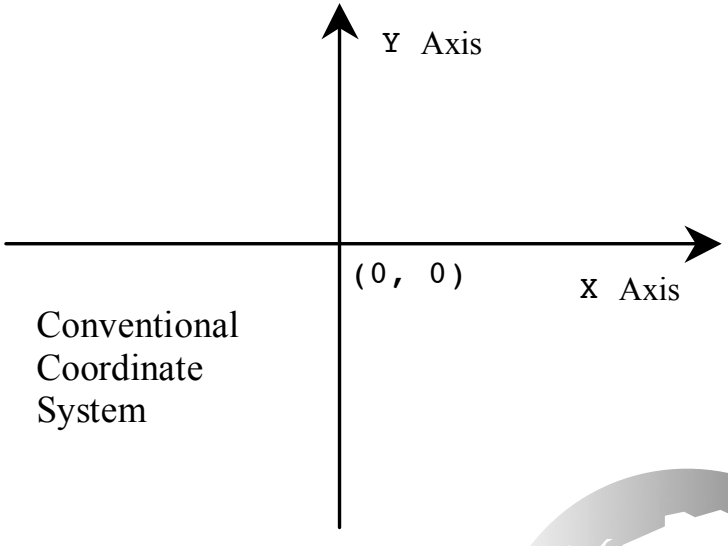
```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

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);
  }
}
```



Display a Shape

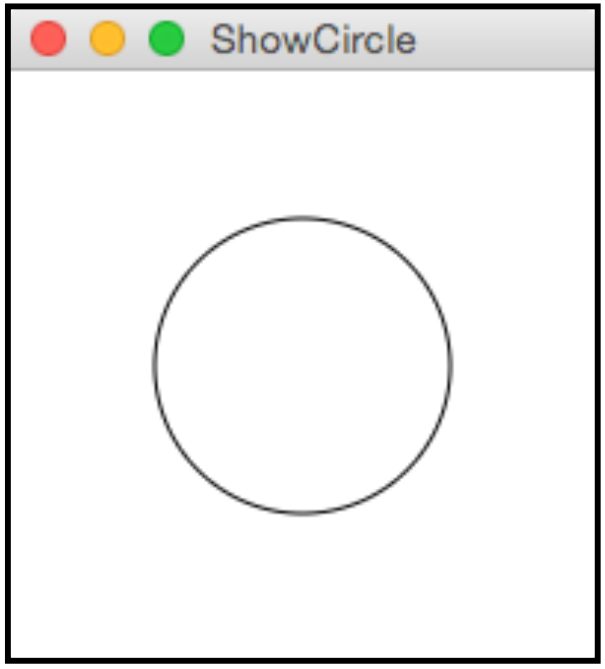
This example displays a circle in the center of the pane.





ShowCircle

```
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(null);

  // Create a pane to hold the circle
  Pane pane = new Pane();
  pane.getChildren().add(circle);

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



Binding Properties

JavaFX introduces a new concept called *binding property* that enables a *target object* to be bound to a *source object*. If the value in the source object changes, the target property is also changed automatically. The target object is simply called a *binding object* or a *binding property*.

ShowCircleCentered

Binding Property: getter, setter, and property getter

```
public class SomeClassName {

  private PropertyType x;

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

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

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

(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

```
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
 }
```

Uni/Bidirectional Binding

```
import javafx.beans.property.DoubleProperty;
import javafx.beans.property.SimpleDoubleProperty;

public class BindingDemo {
  public static void main(String[] args) {
    DoubleProperty d1 = new SimpleDoubleProperty(1);
    DoubleProperty d2 = new SimpleDoubleProperty(2);
    d1.bind(d2);
    System.out.println("d1 is " + d1.getValue()
      + " and d2 is " + d2.getValue());
    d2.setValue(70.2);
    System.out.println("d1 is " + d1.getValue()
      + " and d2 is " + d2.getValue());
  }
}
```

```
d1 is 2.0 and d2 is 2.0
d1 is 70.2 and d2 is 70.2
```

UnidirctionalBindingDemo

```
public static void main(String[] args) {
  DoubleProperty d1 = new SimpleDoubleProperty(1);
  DoubleProperty d2 = new SimpleDoubleProperty(2);
  d1.bindBidirectional (d2);

  System.out.println("d1 is " + d1.getValue()
    + " and d2 is " + d2.getValue());
  d2.setValue(70.2);
  System.out.println("d1 is " + d1.getValue()
    + " and d2 is " + d2.getValue());

  d1.setValue(80.2);
  System.out.println("d1 is " + d1.getValue()
    + " and d2 is " + d2.getValue());
}
```

BidirctionalBindingDemo

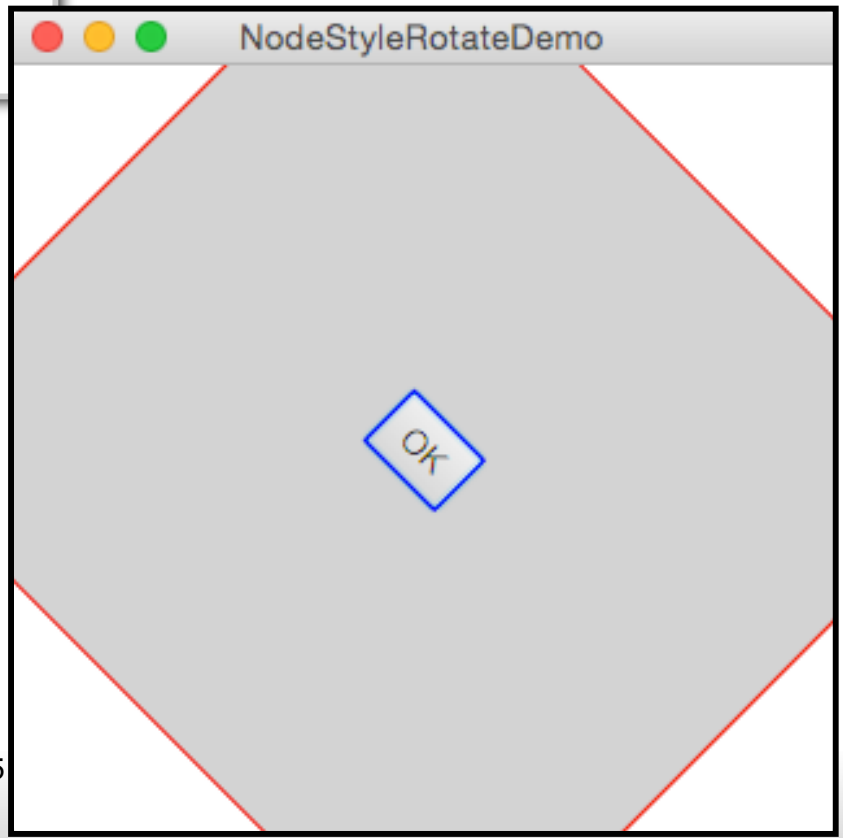
Common Properties and Methods for Nodes

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

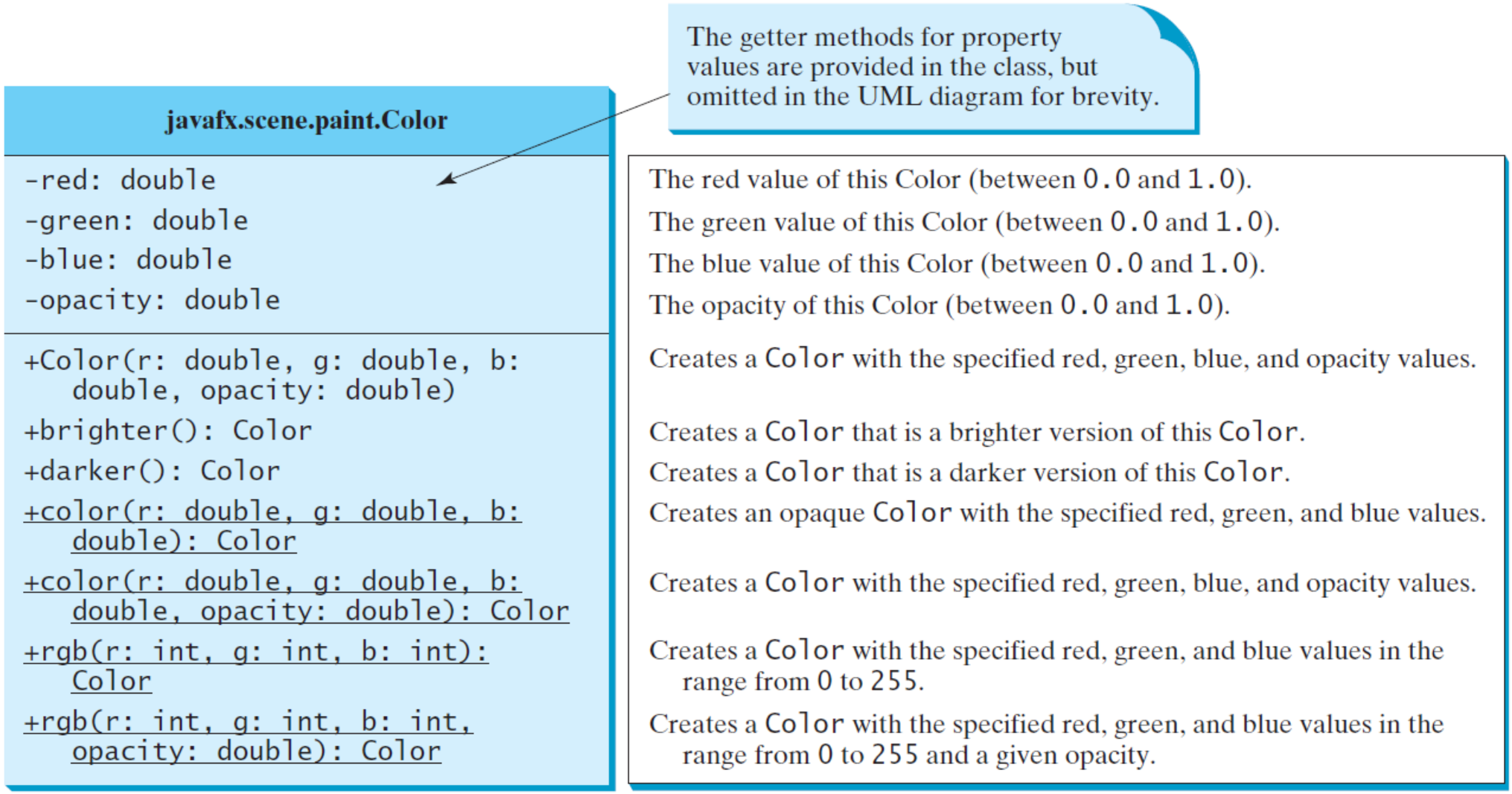
NodeStyleRotateDemo

```
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
 }
```

https://docs.oracle.com/javafx/2/api/javafx/scene/doc-files/cssref.html



The Color Class



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>
```

-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

```
// 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);
```

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
-height: ReadOnlyBooleanProperty
-width: ReadOnlyBooleanProperty
-progress: ReadOnlyBooleanProperty
```

```
+Image(filenameOrURL: String)
```

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

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

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

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

```
// 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);
```

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>
-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.

ShowFlowPane

```
// 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);
.
.
.
```

javafx.geometry.Insets.Insets(@NamedArg(value="top") **double top,** @NamedArg(value="right") **double right,** @NamedArg(value="bottom") **double bottom,** @NamedArg(value="left") **double 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
+setHalighnment(child: Node,
   value: HPos): void
+setValighnment(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.

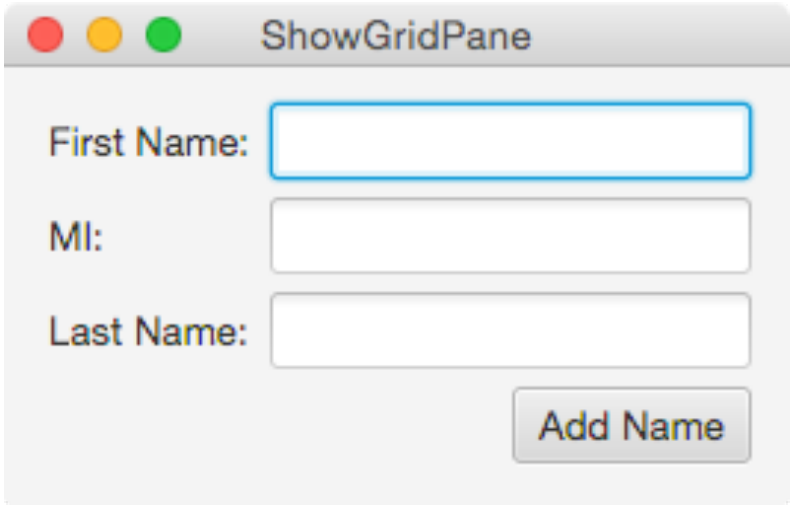
Returns the row index for the specified node.
Sets a node to a new row. This method repositions the node.

Sets the horizontal alignment for the child in the cell.

Sets the vertical alignment for the child in the cell.

ShowGridPane

```
// 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);
.
.
.
```



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.

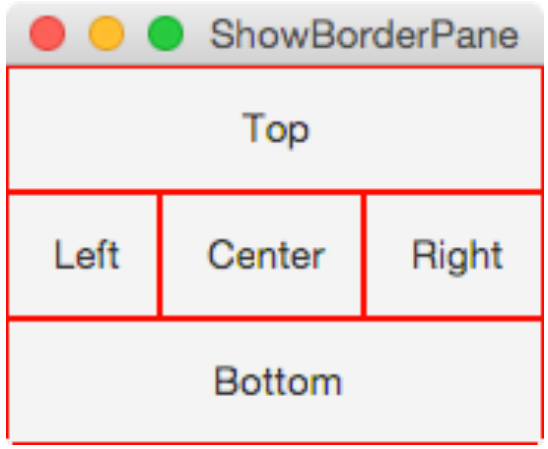
The node placed in the top region (default: `null`).
The node placed in the right region (default: `null`).
The node placed in the bottom region (default: `null`).
The node placed in the left region (default: `null`).
The node placed in the center region (default: `null`).

Creates a `BorderPane`.

Sets the alignment of the node in the `BorderPane`.

ShowBorderPane

```
// 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);
.
.
.
```




```
// 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>
-fillHeight: BooleanProperty
-spacing: DoubleProperty
```

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

The overall alignment of the children in the box (default: `Pos.TOP_LEFT`).

Is resizable children fill the full height of the box (default: `true`).

The horizontal gap between two nodes (default: `0`).

Creates a default `HBox`.

Creates an `HBox` with the specified horizontal gap between nodes.

Sets the margin for the node in the pane.

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`).

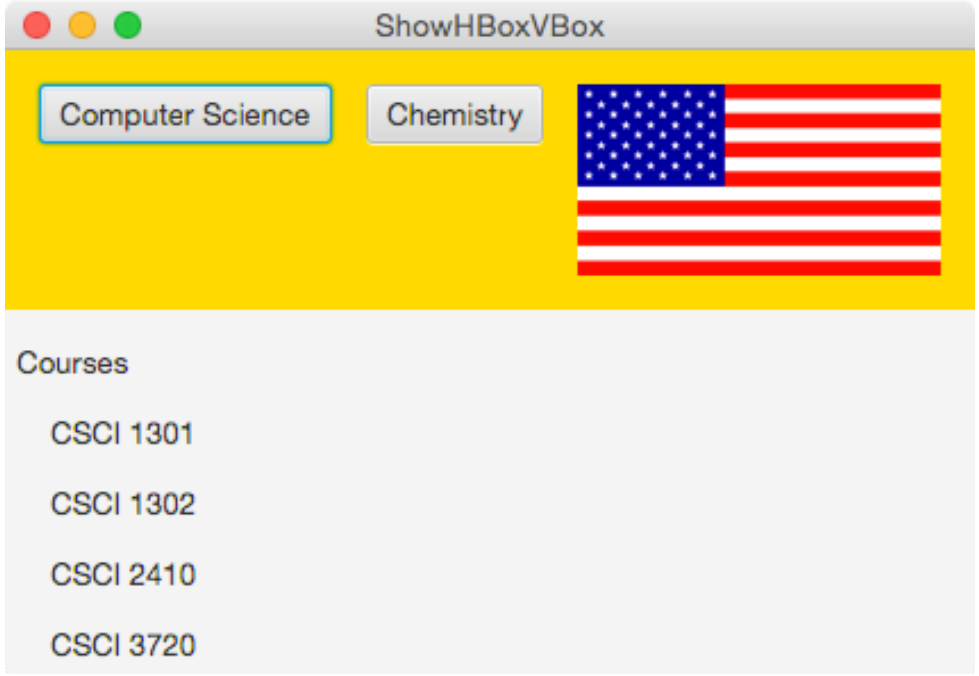
Creates a default `VBox`.

Creates a `VBox` with the specified horizontal gap between nodes.

Sets the margin for the node in the pane.

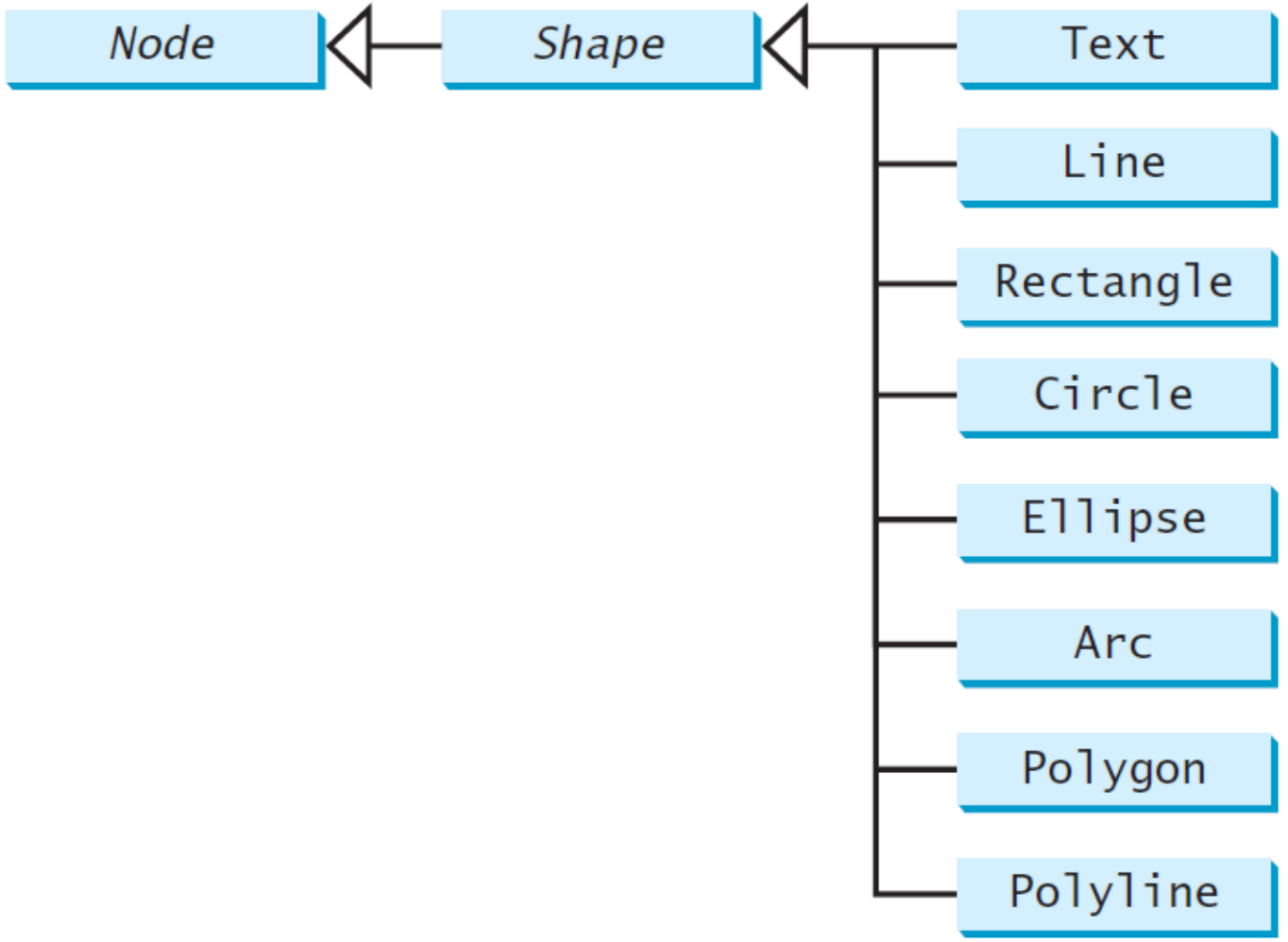
ShowHBoxVBox

```
    // 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);
.
.
.
  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;
  }
  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.



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

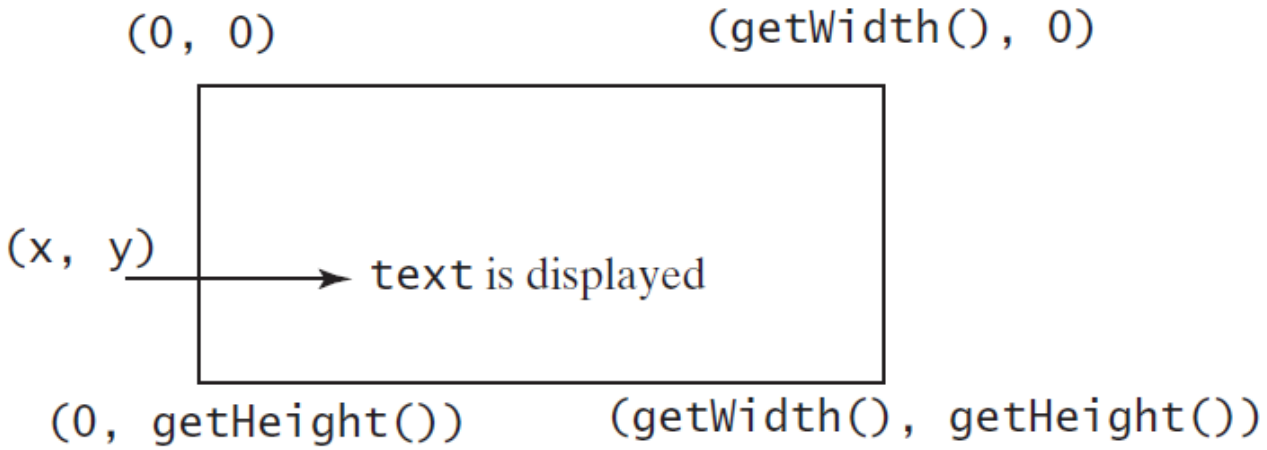
```
-text: StringProperty
-x: DoubleProperty
-y: DoubleProperty
-underline: BooleanProperty
-strikethrough: BooleanProperty
-font: ObjectProperty<Font>
```

```
+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

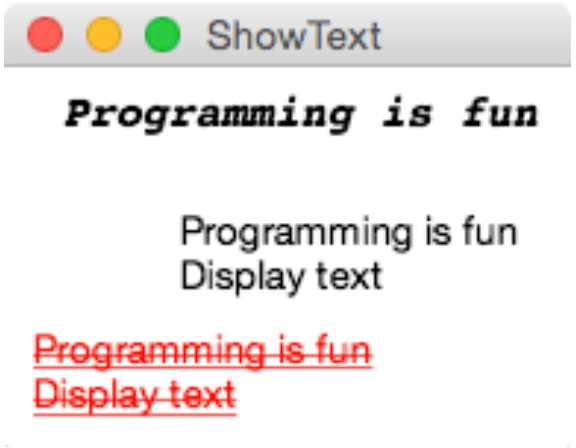


(a) Text(x, y, text)

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

ShowText

```
// 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);
```



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.

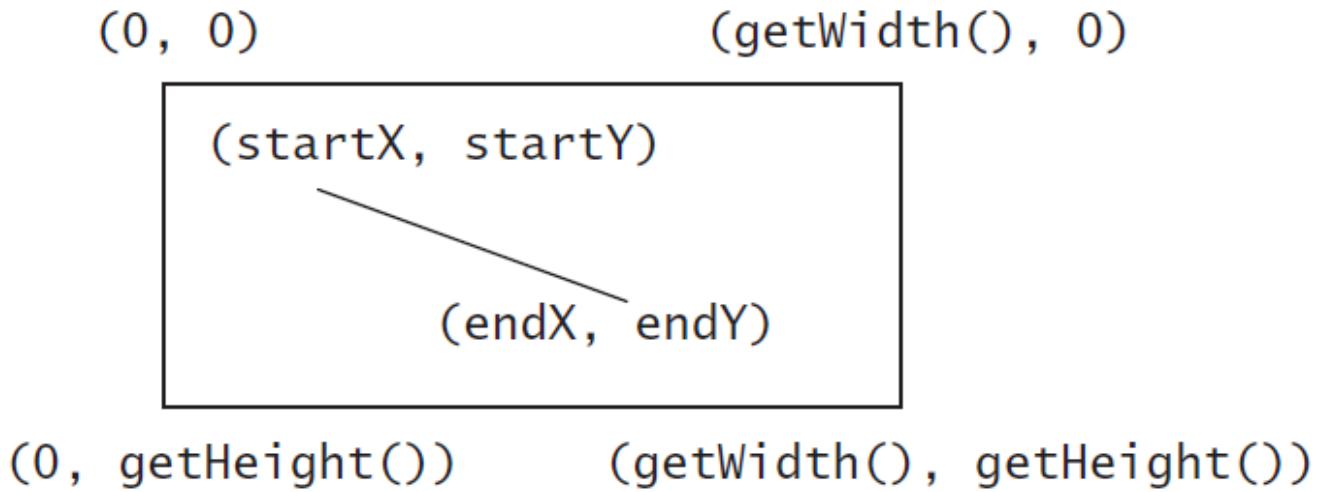
javafx.scene.shape.Line

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

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

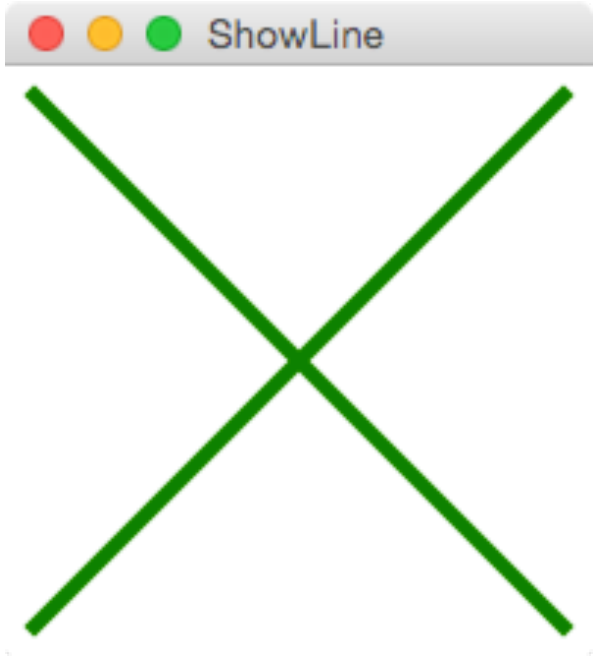
The x-coordinate of the start point.
The y-coordinate of the start point.
The x-coordinate of the end point.
The y-coordinate of the end point.

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



ShowLine

```
  public void start(Stage primaryStage) {
    // Create a scene and place it in the stage
    Scene scene = new Scene(new LinePane(), 200, 200);
.
.
.
}

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
-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).

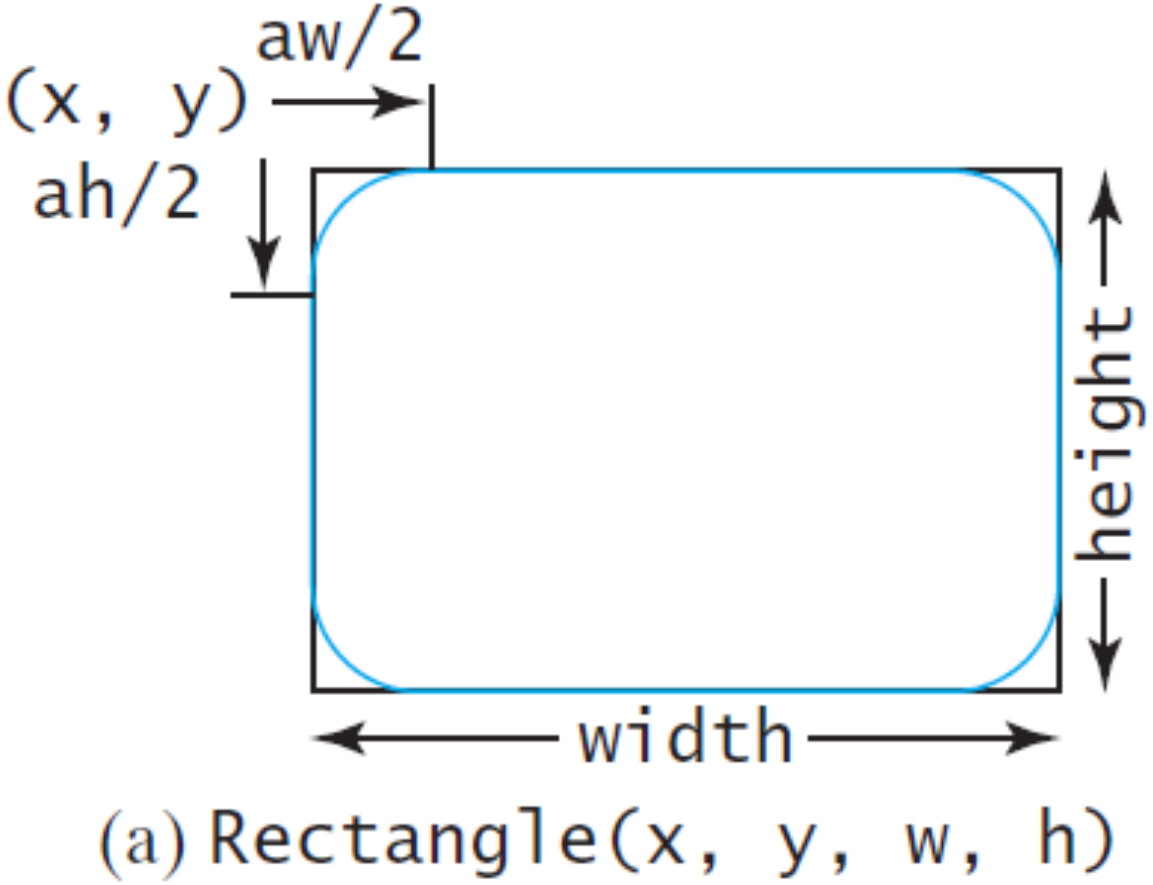
The `arcWidth` of the rectangle (default: 0). `arcWidth` is the horizontal diameter of the arcs at the corner (see Figure 14.31a).

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.

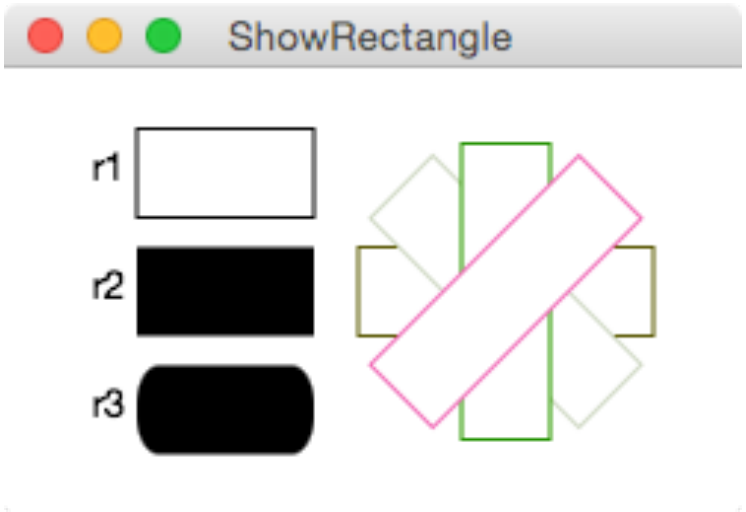
Rectangle Example



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

ShowRectangle

```
// Create rectangles
Rectangle r1 = new Rectangle(25, 10, 60, 30);
r1.setStroke(Color.BLACK);
r1.setFill(Color.WHITE);
Rectangle r2 = new Rectangle(25, 50, 60, 30);
Rectangle r3 = new Rectangle(25, 90, 60, 30);
r3.setArcWidth(15);
r3.setArcHeight(25);

// Create a group and add nodes to the group
Group group = new Group();
group.getChildren().addAll(new Text(10, 27, "r1"), r1,
  new Text(10, 67, "r2"), r2, new Text(10, 107, "r3"), 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);
  group.getChildren().add(r);
}

// Create a scene and place it in the stage
Scene scene = new Scene(new BorderPane(group), 250, 150);
```



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.

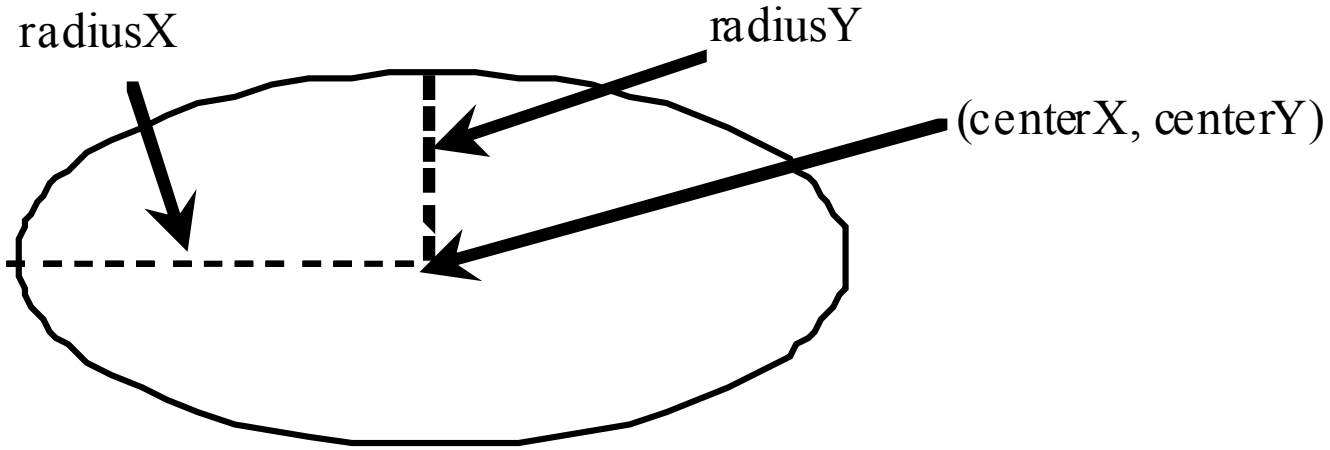
javafx.scene.shape.Ellipse

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

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

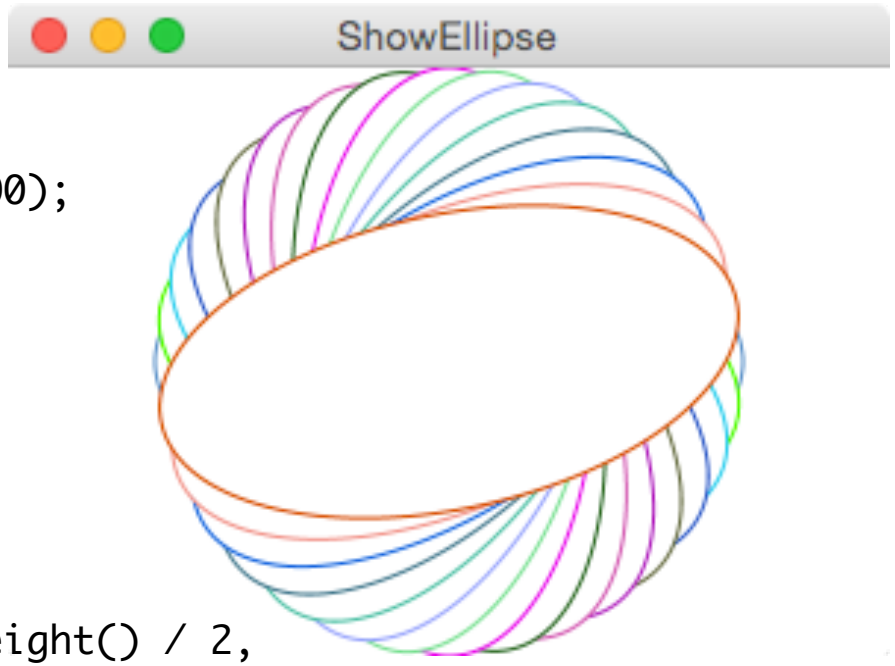
The x-coordinate of the center of the ellipse (default 0).
The y-coordinate of the center of the ellipse (default 0).
The horizontal radius of the ellipse (default: 0).
The vertical radius of the ellipse (default: 0).

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

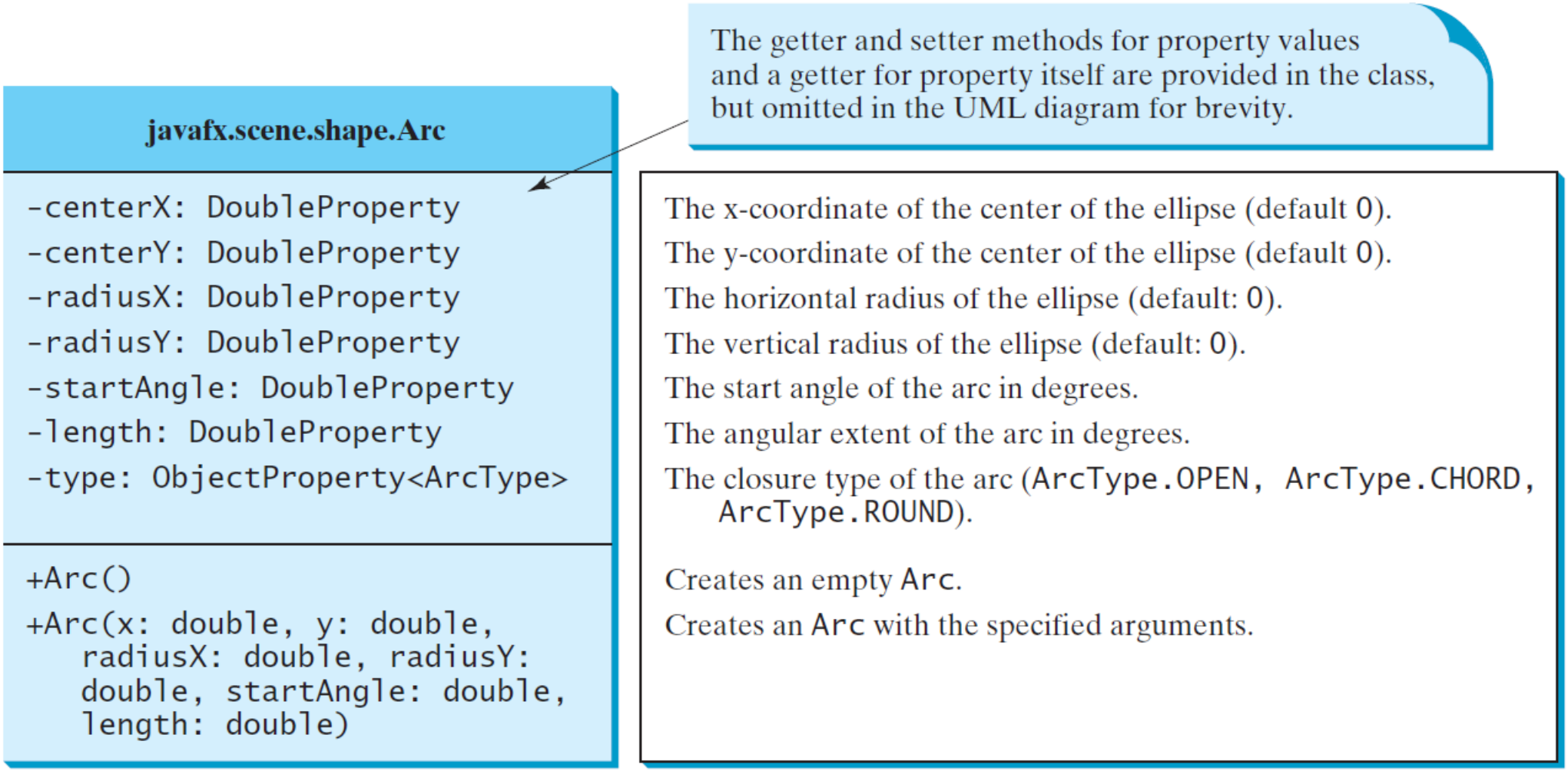


ShowEllipse

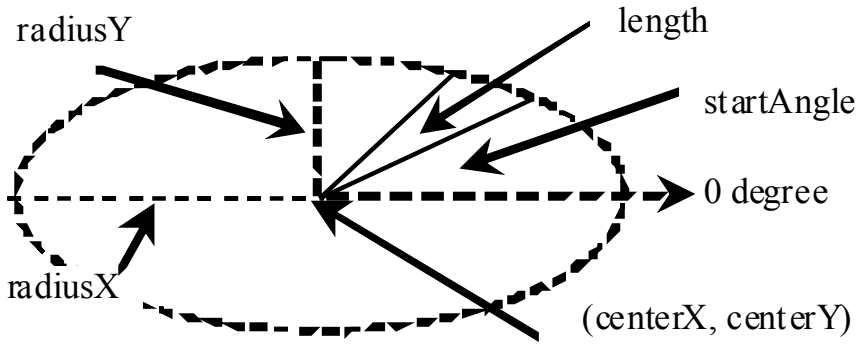
```
public void start(Stage primaryStage) {
    // Create a scene and place it in the stage
    Scene scene = new Scene(new MyEllipse(), 300, 200);
.
.
}
class MyEllipse extends Pane {
  private void paint() {
    getChildren().clear();
    for (int i = 0; i < 16; i++) {
      // Create an ellipse and add it to pane
      Ellipse e1 = new Ellipse(getWidth() / 2, getHeight() / 2,
        getWidth() / 2 - 50, getHeight() / 2 - 50);
      e1.setStroke(Color.color(Math.random(), Math.random(),
        Math.random()));
      e1.setFill(Color.WHITE);
      e1.setRotate(i * 180 / 16);
      getChildren().add(e1);
    }
  }
  .
  .

}
```

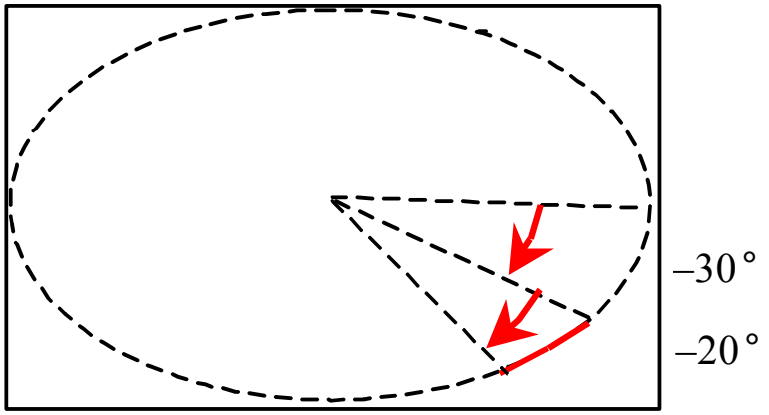


Arc

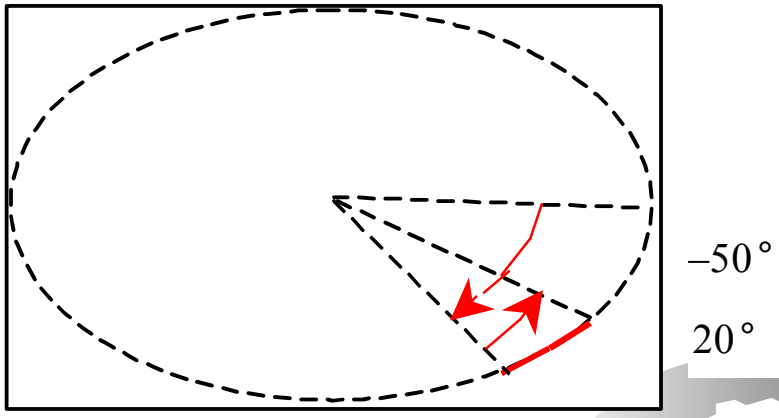

javafx.scene.shape.Arc
-centerX: DoubleProperty
-centerY: DoubleProperty
-radiusX: DoubleProperty
-radiusY: DoubleProperty
-startAngle: DoubleProperty
-length: DoubleProperty
-type: ObjectProperty<ArcType>
+Arc()
+Arc(x: double, y: double, radiusX: double, radiusY: double, startAngle: double, length: double)
The 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 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





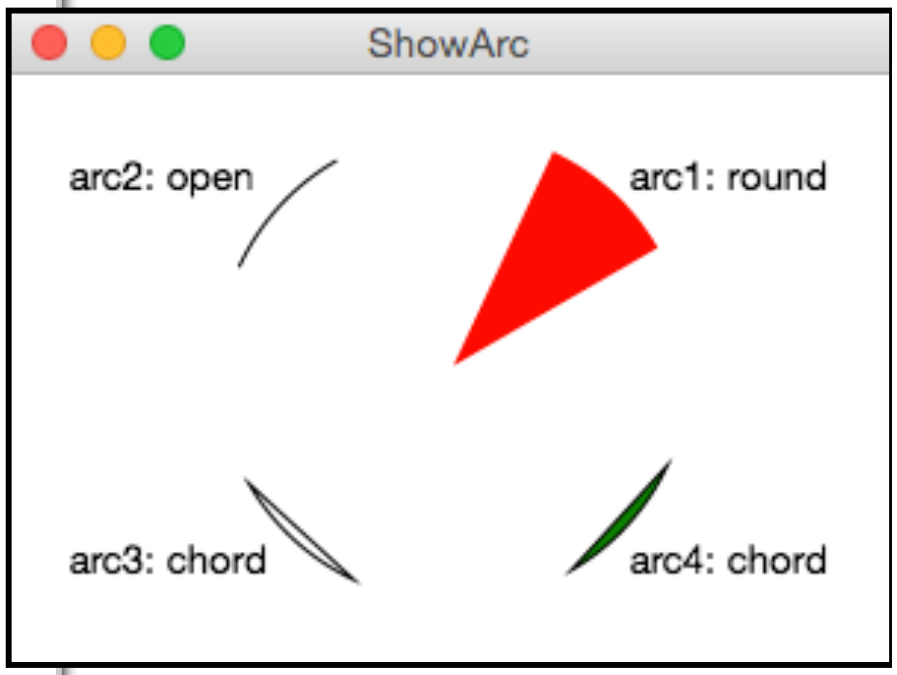
(a) Negative starting angle –30° and negative spanning angle –20°



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

Show Arc

```
Arc arc1 = new Arc(150, 100, 80, 80, 30, 35); // Create an arc
arc1.setFill(Color.RED); // Set fill color
arc1.setType(ArcType.ROUND); // Set arc type

Arc arc2 = new Arc(150, 100, 80, 80, 30 + 90, 35);
arc2.setFill(Color.WHITE);
arc2.setType(ArcType.OPEN);
arc2.setStroke(Color.BLACK);

Arc arc3 = new Arc(150, 100, 80, 80, 30 + 180, 35);
arc3.setFill(Color.WHITE);
arc3.setType(ArcType.CHORD);
arc3.setStroke(Color.BLACK);

Arc arc4 = new Arc(150, 100, 80, 80, 30 + 270, 35);
arc4.setFill(Color.GREEN);
arc4.setType(ArcType.CHORD);
arc4.setStroke(Color.BLACK);

// Create a group and add nodes to the group
Group group = new Group();
group.getChildren().addAll(new Text(210, 40, "arc1: round"),
  arc1, new Text(20, 40, "arc2: open"), arc2,
  new Text(20, 170, "arc3: chord"), arc3,
  new Text(210, 170, "arc4: chord"), arc4);

// Create a scene and place it in the stage
Scene scene = new Scene(new BorderPane(group), 300, 200);
```



Polygon

javafx.scene.shape.Polygon
+Polygon()
+Polygon(double... points)
+getPoints(): ObservableList<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.

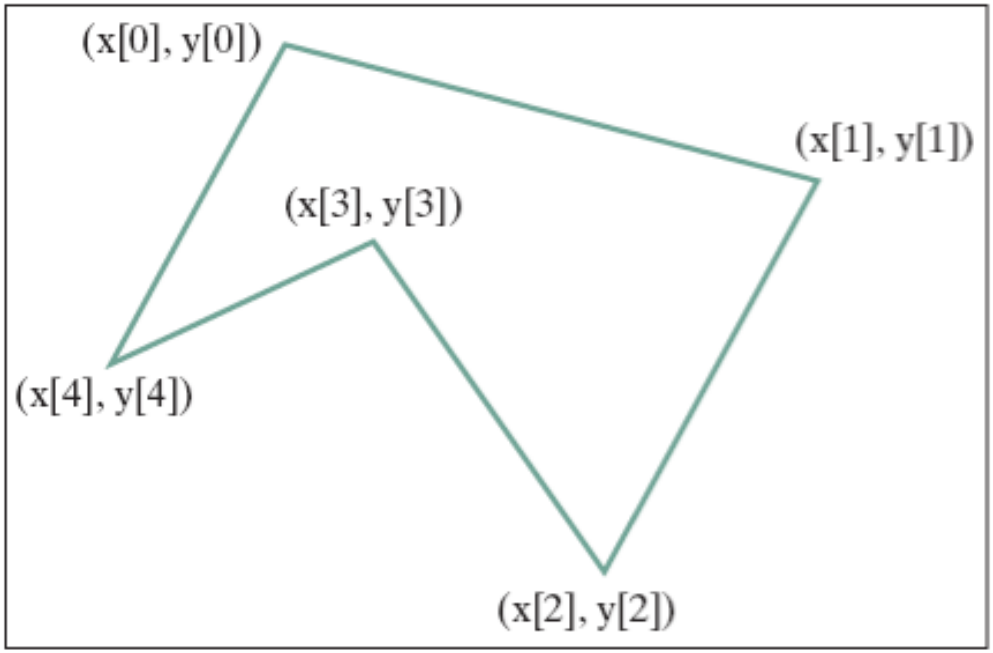
Creates an empty polygon.

Creates a polygon with the given points.

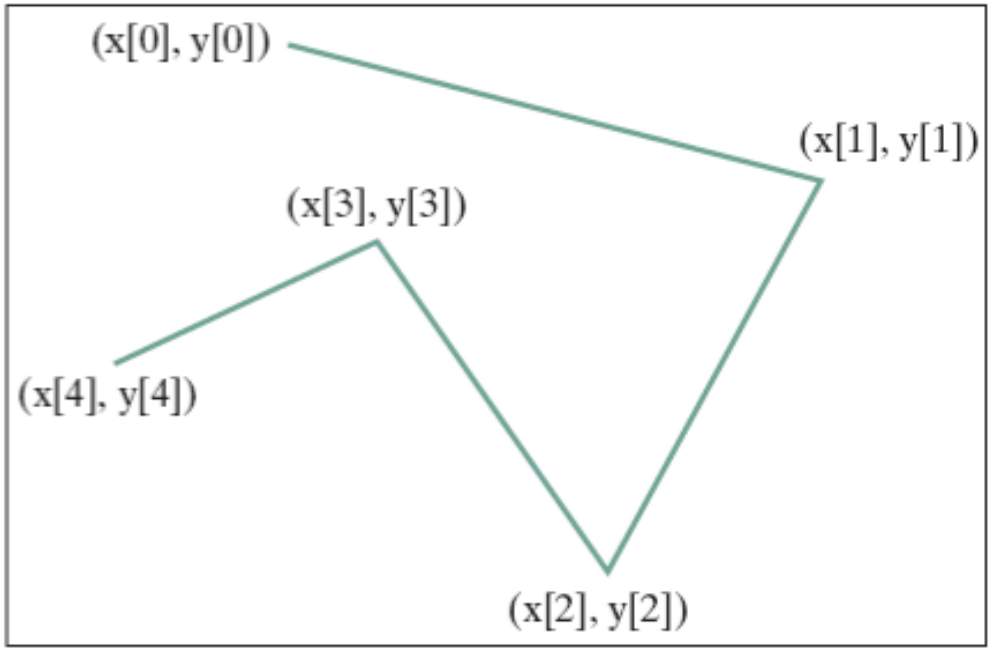
Returns a list of double values as x- and y-coordinates of the points.

ShowPolygon

Polygon and Polyline



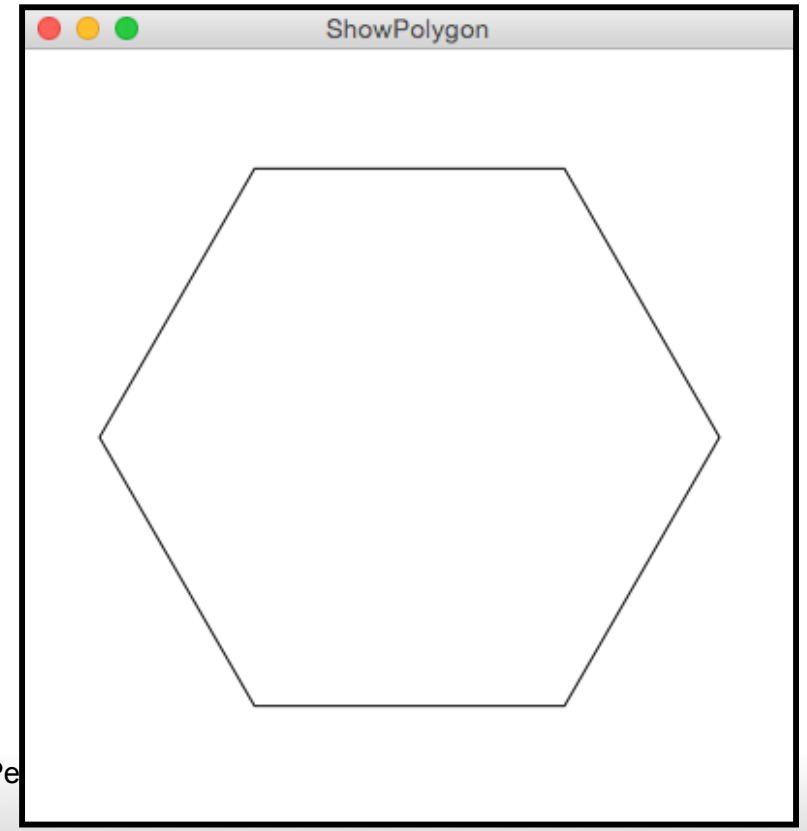
(a) Polygon



(b) Polyline

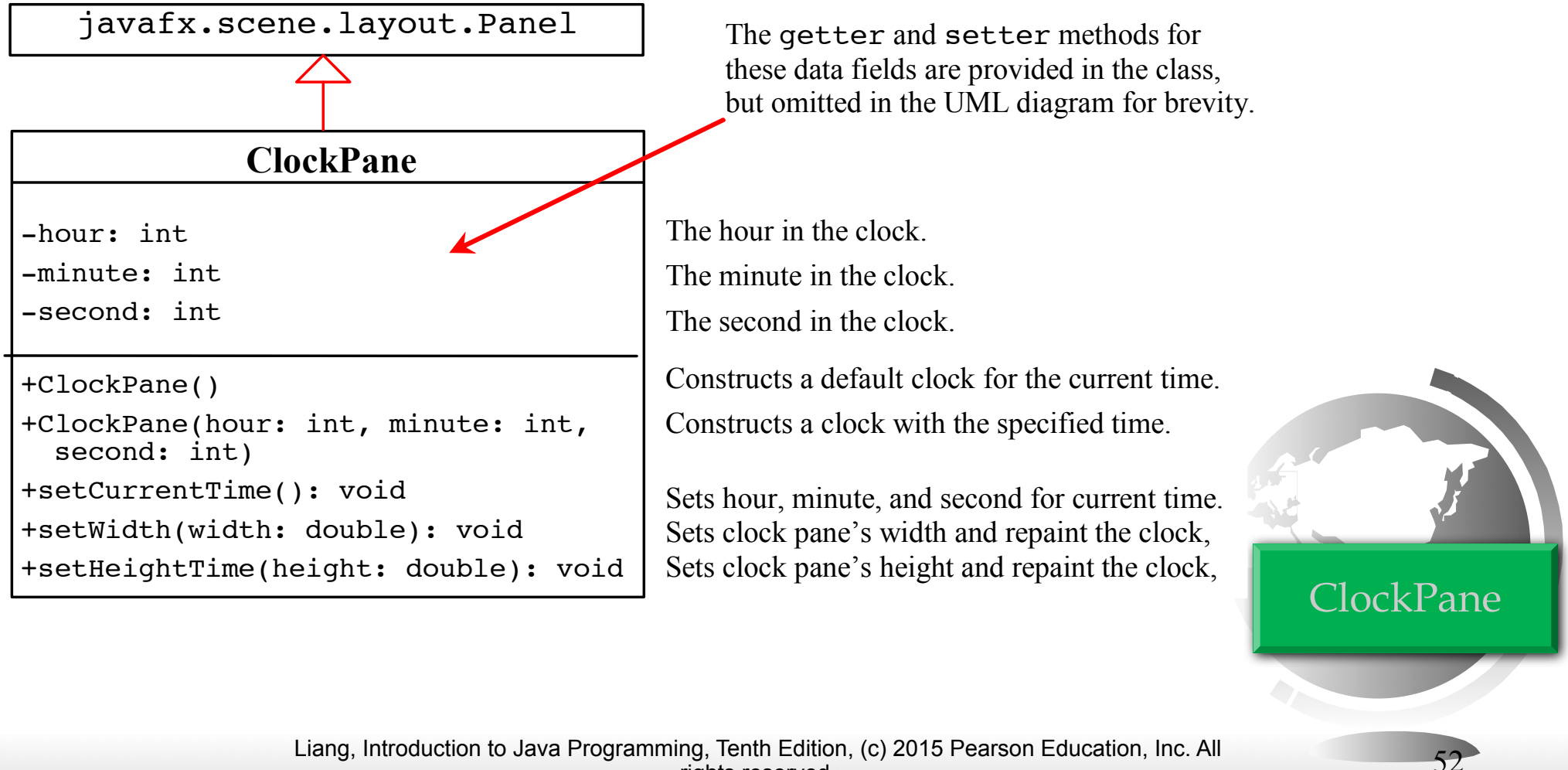
ShowPolygon

```
    // Create a scene and place it in the stage
    Scene scene = new Scene(new MyPolygon(), 400, 400);
    .
    .
class MyPolygon extends Pane {
  private void paint() {
    // Create a polygon and place polygon to pane
    Polygon polygon = new Polygon();
    polygon.setFill(Color.WHITE);
    polygon.setStroke(Color.BLACK);
    ObservableList<Double> list = polygon.getPoints();

    double centerX = getWidth() / 2, centerY = getHeight() / 2;
    double radius = Math.min(getWidth(), getHeight()) * 0.4;

    // Add points to the polygon list
    for (int i = 0; i < 6; i++) {
      list.add(centerX + radius * Math.cos(2 * i * Math.PI / 6));
      list.add(centerY - radius * Math.sin(2 * i * Math.PI / 6));
    }

    getChildren().clear();
    getChildren().add(polygon);
  }
  .
  .
  .
}
```

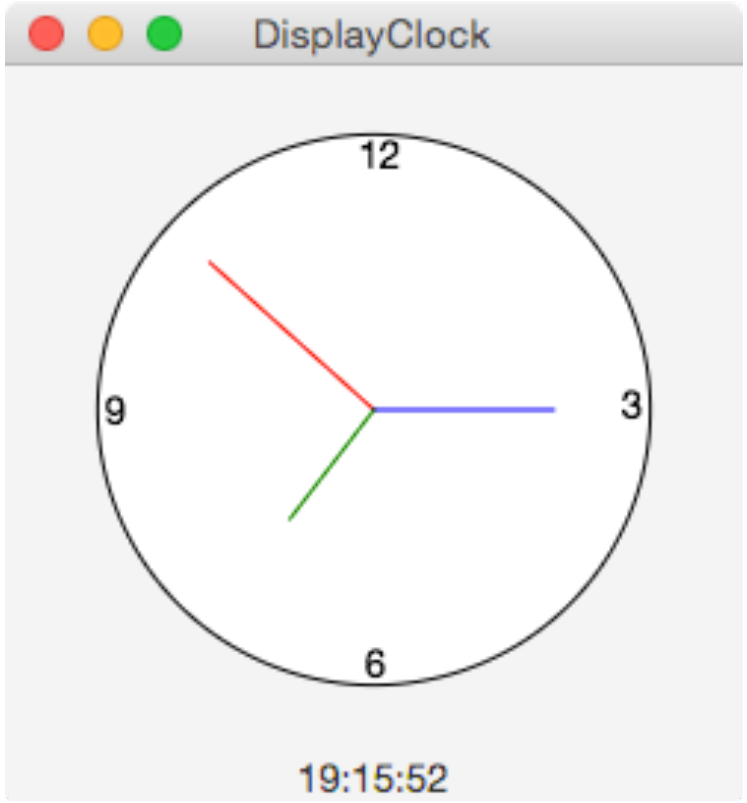


Case Study: The ClockPane Class

This case study develops a class that displays a clock on a pane.



Use the ClockPane Class



DisplayClock