COMPUTER SCIENCE DEPARTMENT FACULTY OF ENGINEERING AND TECHNOLOGY

ADVANCED PROGRAMMING COMP231

Instructor :Murad Njoum
Office : Masri322

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.

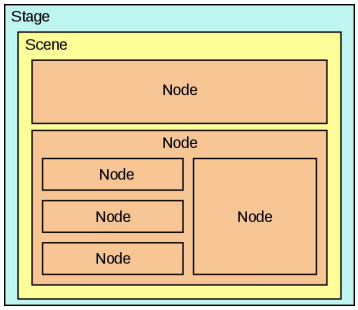
Specifically, this chapter introduces the framework of JavaFX and discusses **JavaFX GUI components** and their relationships.

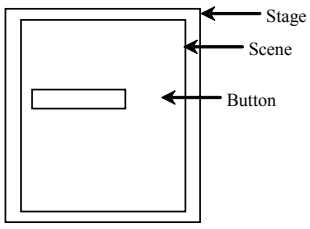
JavaFX vs Swing and AWT

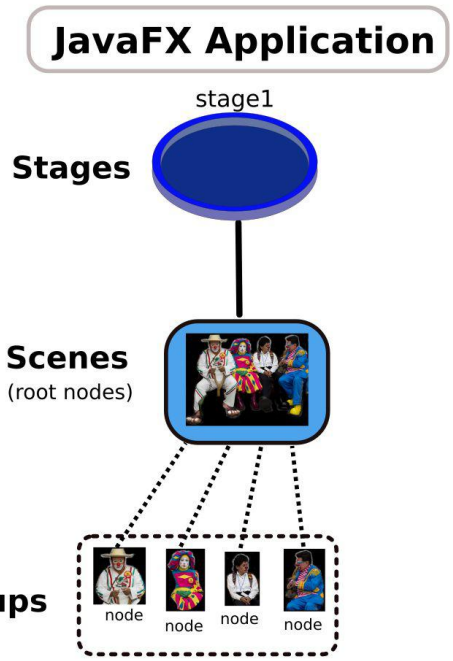
- When Java was introduced, the GUI classes were bundled in a library known as the **Abstract Windows Toolkit (AWT).**
- AWT is fine for developing simple graphical user interfaces, but not for **developing comprehensive GUI projects**.
- In addition, AWT is prone to platform-specific bugs. The AWT user-interface 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.**
- Swing and AWT are replaced by the JavaFX platform for developing **rich Internet applications**.

Basic Structure of JavaFX

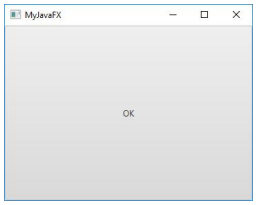
- Application
- Override the start(Stage) method
- Stage, Scene, and Nodes



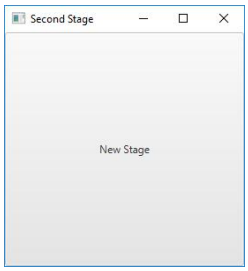




```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.stage.Stage;

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

 //add second stage add code here
  }

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



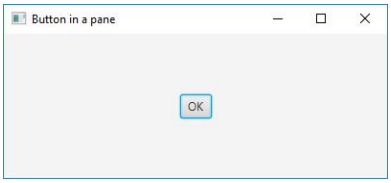
1 2 3 4 5 6 7



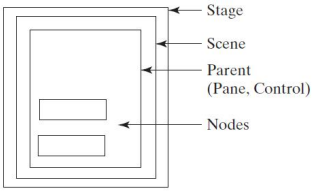

```
Stage stage = new Stage( ); // Create a new stage
stage.setTitle('Second Stage' ); // Set the stage title
// Set a scene with a button in the stage
stage.setScene(new Scene(new Button('New Stage'), 200, 250));
stage.show( ); // Display the stage
```

StackPane is a container which can contain different interface components, subcomponents stacked up to others, and at a certain moment, you can only see the subcomponent lying on the top of Stack.

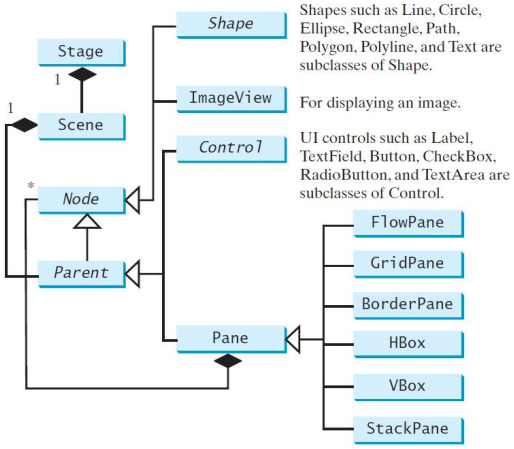
```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

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"));
//getChildren method is used to get the children components(such as checkboxes, buttons) in a container
    Scene scene = new Scene(pane, 400, 150);
    primaryStage.setTitle("Button in a pane"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }

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



Panes, UI Controls, and Shapes



(a)



Shapes such as Line, Circle, Ellipse, Rectangle, Path, Polygon, Polyline, and Text are subclasses of Shape.

For displaying an image.

UI controls such as Label, TextField, Button, CheckBox, RadioButton, and TextArea are subclasses of Control.

(b)

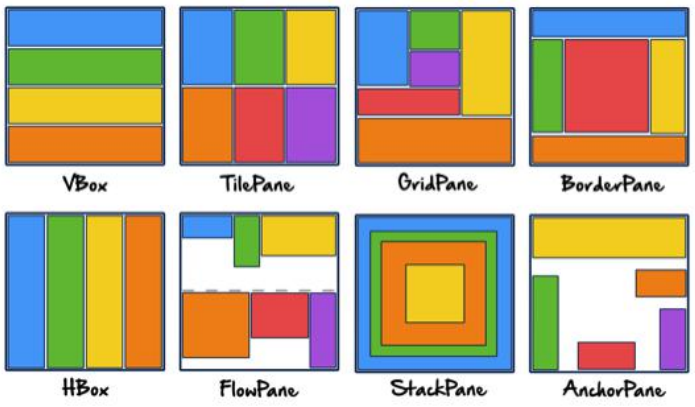
Layout Panes

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.

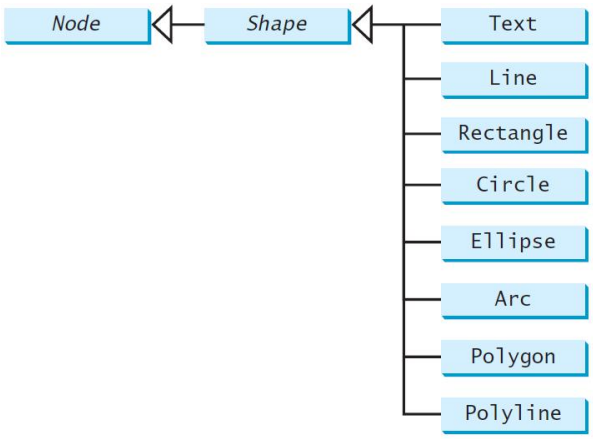
Example 1-1 Create a Border Pane

```
BorderPane border = new BorderPane();
HBox hbox = addHBox()
border.setTop(hbox);
border.setLeft(addVBox());
addStackPane(hbox);       // Add stack to HBox in top region

border.setCenter(addGridPane());
border.setRight(addFlowPane());
```



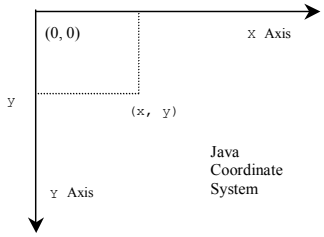
Shapes

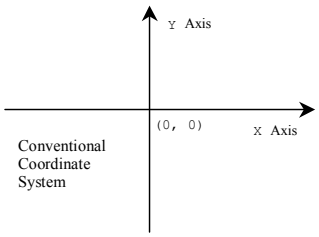
JavaFX provides many shape classes for drawing texts, lines, circles, rectangles, ellipses, arcs, polygons, and polylines.



Display a Shape

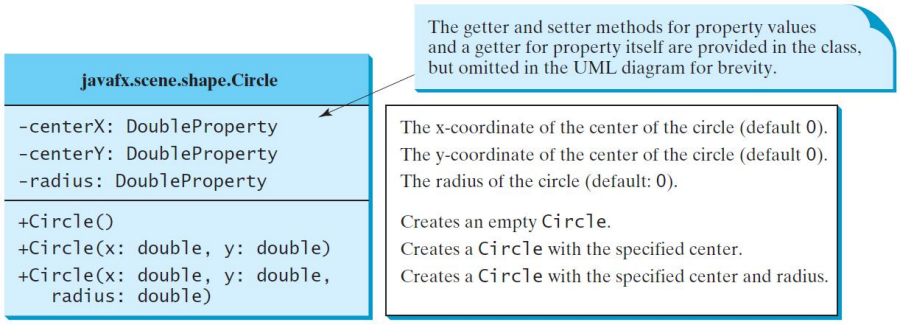
This example displays a circle in the center of the pane.



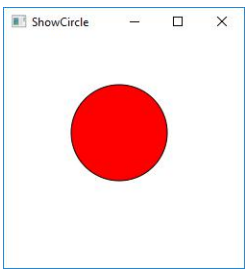


ShowCircle Run

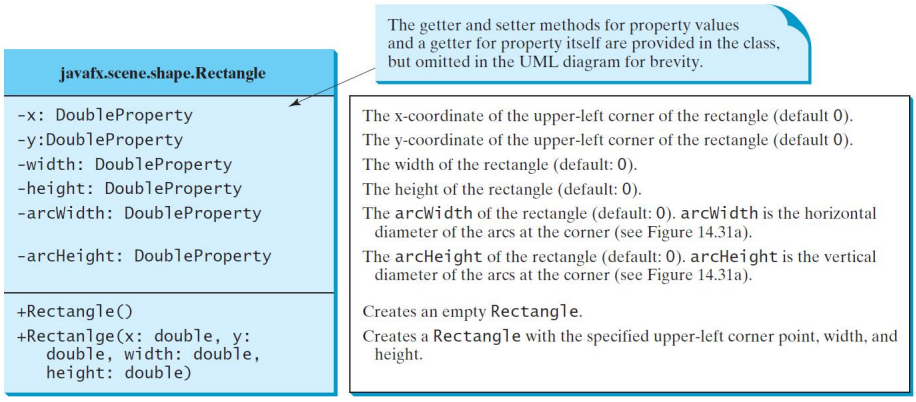
Circle



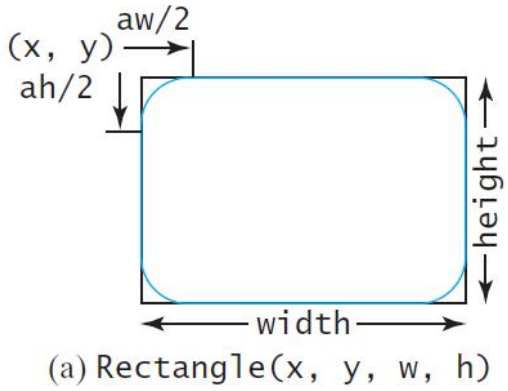

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.stage.Stage;

public class ShowCircle extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // Create a circle and set its properties
    Circle circle = new Circle();
    circle.setCenterX(120);
    circle.setCenterY(100);
    circle.setRadius(50);
    circle.setStroke(Color.BLACK);
    circle.setFill(Color.RED);

    // 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, 250, 250);
    primaryStage.setTitle("ShowCircle"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }

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



Rectangle



Rectangle Example



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

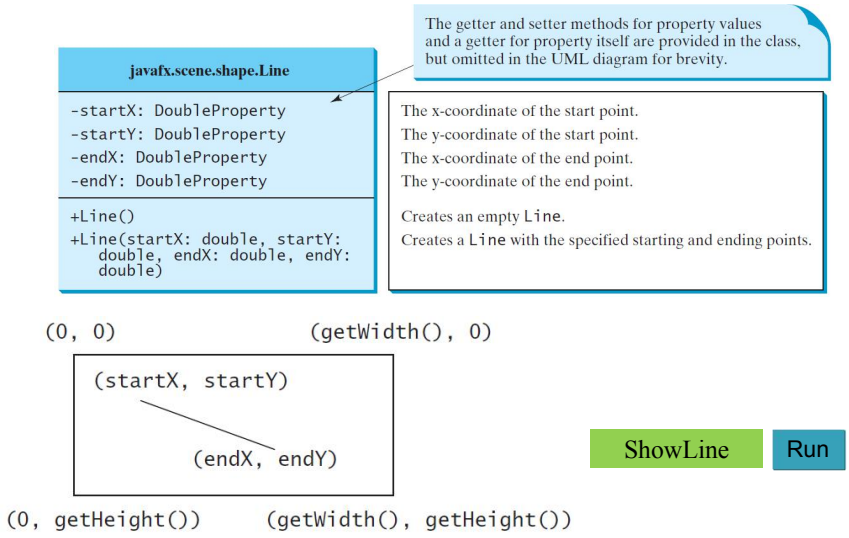
ShowRectangle Run

```
public class ShowRectangle extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // 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);
    primaryStage.setTitle("ShowRectangle"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }

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

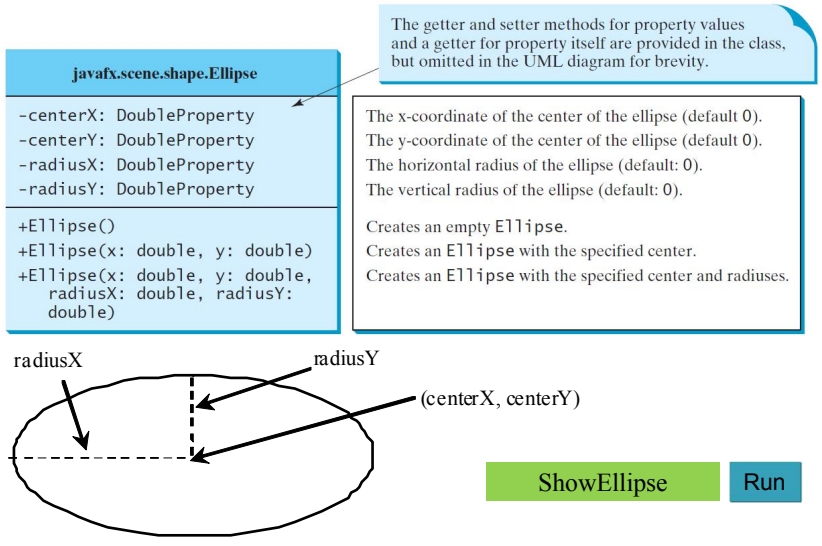
Line




```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.stage.Stage;
import javafx.scene.shape.Line;

public class ShowLine extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // Create a scene and place it in the stage
    Scene scene = new Scene(new LinePane(), 200, 200);
    primaryStage.setTitle("ShowLine"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }
}
```

```
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);
  }
}
```


Ellipse



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

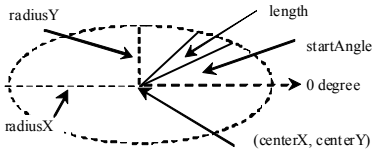
```
-centerX: DoubleProperty
-centerY: DoubleProperty
-radiusX: DoubleProperty
-radiusY: DoubleProperty
-startAngle: DoubleProperty
-length: DoubleProperty
-type: ObjectProperty<ArcType>
```

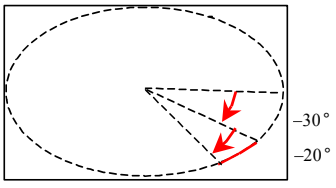
```
+Arc()
+Arc(x: double, y: double,
   radiusX: double, radiusY:
   double, startAngle: double,
   length: double)
```

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`).

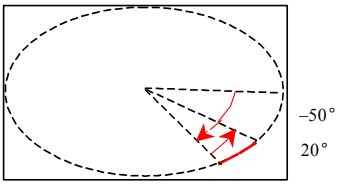
Creates an empty `Arc`.
Creates an `Arc` with the specified arguments.

Arc Examples





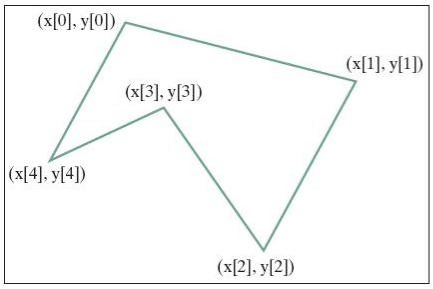
(a) Negative starting angle –30° and negative spanning angle –20°



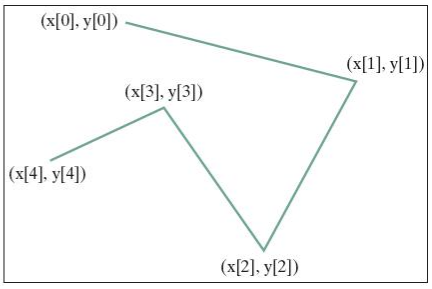
(b) Negative starting angle –50° and positive spanning angle 20°

ShowArc Run

Polygon and Polyline



(a) Polygon



(b) Polyline

Polygon

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

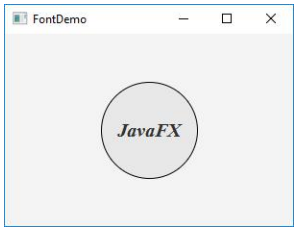
The Font Class

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

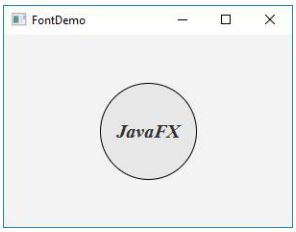
javafx.scene.text.Font	
-size: double	The size of this font.
-name: String	The name of this font.
-family: String	The family of this font.
+Font(size: double)	Creates a Font with the specified size.
+Font(name: String, size: double)	Creates a Font with the specified full font name and size.
+font(name: String, size: double)	Creates a Font with the specified name and size.
+font(name: String, w: FontWeight, size: double)	Creates a Font with the specified name, weight, and size.
+font(name: String, w: FontWeight, p: FontPosture, size: double)	Creates a Font with the specified name, weight, posture, and size.
+getFamilies(): List<String>	Returns a list of font family names.
+getFontNames(): List<String>	Returns a list of full font names including family and weight.

FontDemo Run

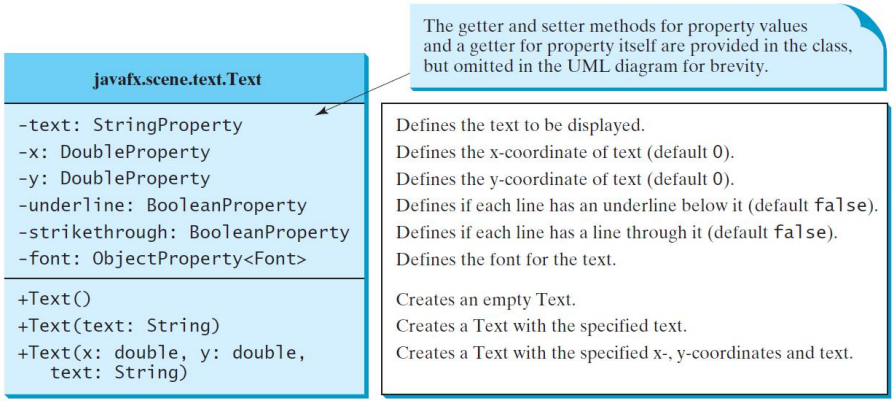
```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.*;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.scene.text.*;
import javafx.scene.control.*;
import javafx.stage.Stage;

public class FontDemo extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // Create a pane to hold the circle
    Pane pane = new StackPane();

    // Create a circle and set its properties
    Circle circle = new Circle();
    circle.setRadius(50);
    circle.setStroke(Color.BLACK);
    circle.setFill(new Color(0.5, 0.5, 0.5, 0.1));
    pane.getChildren().add(circle); // Add circle to the pane
```



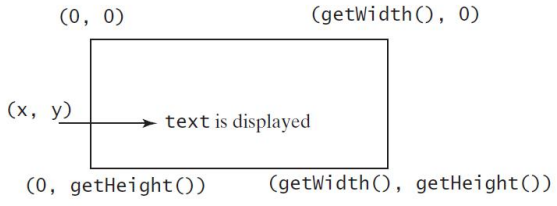

```
    // Create a label and set its properties
      Label label = new Label("JavaFX");
      label.setFont(Font.font("Times New Roman",
        FontWeight.BOLD, FontPosture.ITALIC, 20));
      pane.getChildren().add(label);

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

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



Text



Text Example



(a) Text(x, y, text)

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

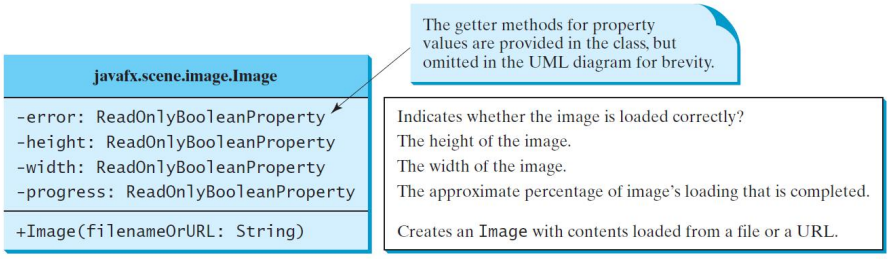
ShowText Run

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.geometry.Insets;
import javafx.stage.Stage;
import javafx.scene.text.Text;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.scene.text.FontPosture;

public class ShowText extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // Create a pane to hold the texts
    Pane pane = new Pane();
    pane.setPadding(new Insets(5, 5, 5, 5));
    Text text1 = new Text(20, 20, "Programming is fun");
    text1.setFont(Font.font("Courier", FontWeight.BOLD,
      FontPosture.ITALIC, 15));
    pane.getChildren().add(text1);
```

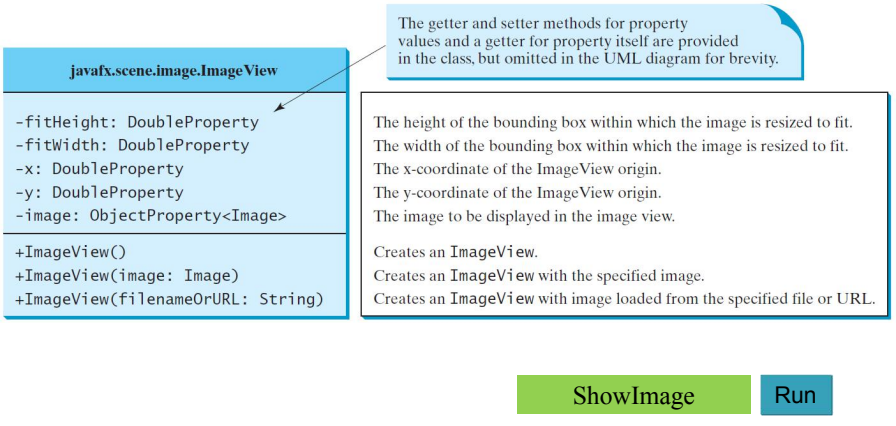


```
    Text text2 = new Text(60, 60, "Programming is fun\nDisplay text");
      pane.getChildren().add(text2);

      Text text3 = new Text(10, 100, "Programming is fun\nDisplay text");
      text3.setFill(Color.RED);
      text3.setUnderline(true);
      text3.setStrikethrough(true);
      pane.getChildren().add(text3);

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

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

The Image Class

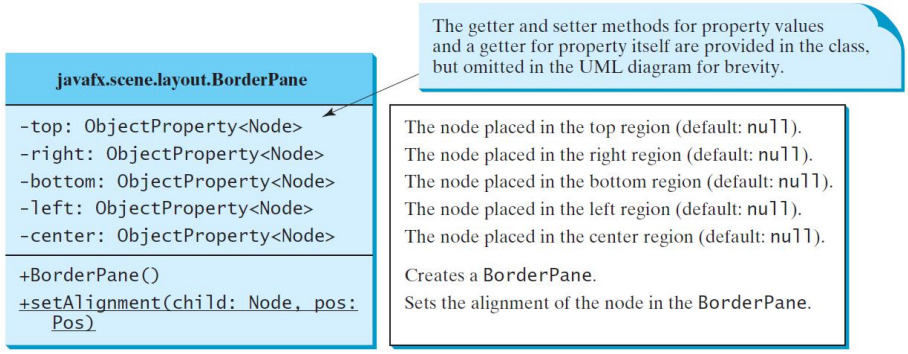


The ImageView Class



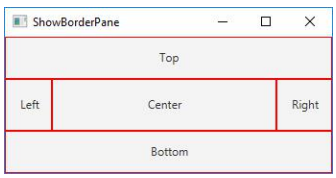
ShowImage Run

BorderPane



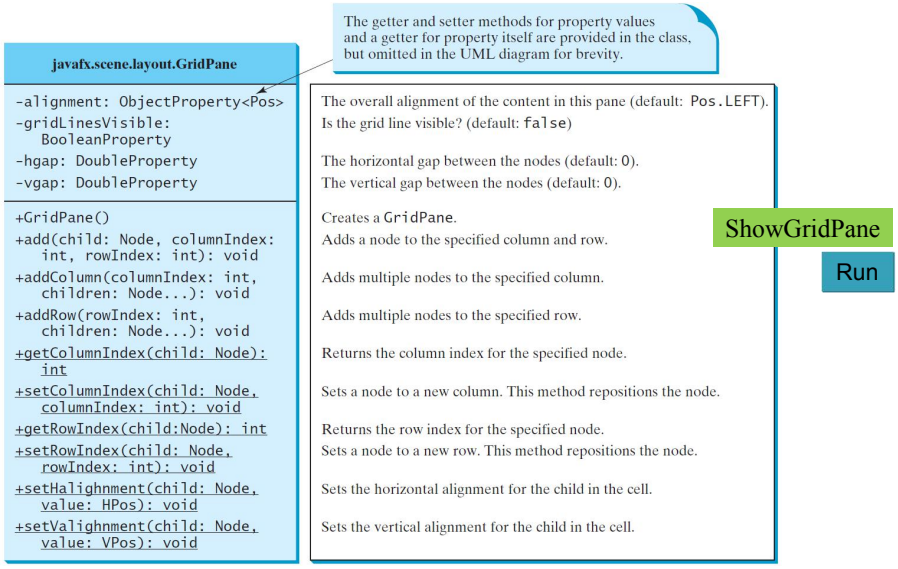
ShowBorderPane Run


```
public class ShowBorderPane extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // Create a border pane
    BorderPane pane = new BorderPane();

    // Place nodes in the pane
    pane.setTop(new CustomPane("Top"));
    pane.setRight(new CustomPane("Right"));
    pane.setBottom(new CustomPane("Bottom"));
    pane.setLeft(new CustomPane("Left"));
    pane.setCenter(new CustomPane("Center"));
// Create a scene and place it in the stage
    Scene scene = new Scene(pane);
    primaryStage.setTitle("ShowBorderPane"); // Set the
    primaryStage.setScene(scene);
   primaryStage.show();
}
   public static void main(String[] args) {
    launch(args);
  }
}
```

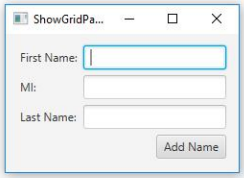



```
//Define a custom pane to hold a label in the center of the
pane
class CustomPane extends StackPane {
  public CustomPane(String title) {
    getChildren().add(new Label(title));
    setStyle("-fx-border-color: red");
setPadding(new Insets(11.5, 12.5, 13.5, 14.5));
  }


}
```

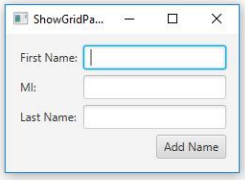
GridPane




```
import javafx.application.Application;
import javafx.geometry.HPos;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

public class ShowGridPane extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // Create a pane and set its properties
    GridPane pane = new GridPane();
    pane.setAlignment(Pos.CENTER);
    pane.setPadding(new Insets(11.5, 12.5, 13.5, 14.5));
    pane.setHgap(5.5);
    pane.setVgap(5.5);
```



```
// Place nodes in the pane
    pane.add(new Label("First Name:"), 0, 0);
    pane.add(new TextField(), 1, 0);
    pane.add(new Label("MI:"), 0, 1);
    pane.add(new TextField(), 1, 1);
    pane.add(new Label("Last Name:"), 0, 2);
    pane.add(new TextField(), 1, 2);
    Button btAdd = new Button("Add Name");
    pane.add(btAdd, 1, 3);
    GridPane.setHalignment(btAdd, HPos.RIGHT);

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

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



FlowPane

javafx.scene.layout.FlowPane

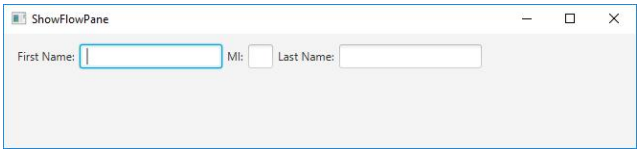
```
-alignment: ObjectProperty<Pos>
-orientation:
   ObjectProperty<Orientation>
-hgap: DoubleProperty
-vgap: DoubleProperty
```

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

The 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.

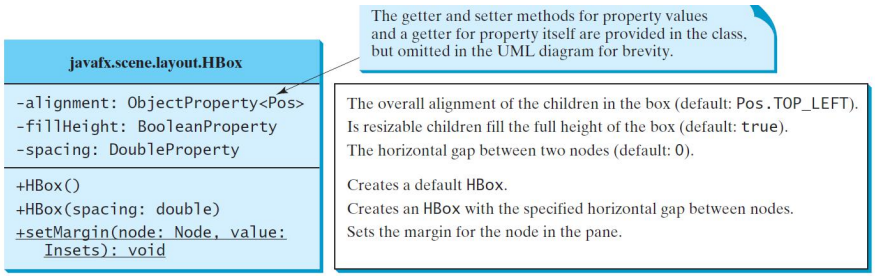
-alignment	The overall alignment of the content in this pane (default: `Pos.LEFT`).
-orientation	The orientation in this pane (default: `Orientation.HORIZONTAL`).
-hgap	The horizontal gap between the nodes (default: 0).
-vgap	The vertical gap between the nodes (default: 0).
+FlowPane()	Creates a default `FlowPane`.
+FlowPane(hgap: double, vgap: double)	Creates a `FlowPane` with a specified horizontal and vertical gap.
+FlowPane(orientation: ObjectProperty<Orientation>)	Creates a `FlowPane` with a specified orientation.
+FlowPane(orientation: ObjectProperty<Orientation>, hgap: double, vgap: double	Creates a `FlowPane` with a specified orientation, horizontal gap and vertical gap.

MultipleStageDemo Run

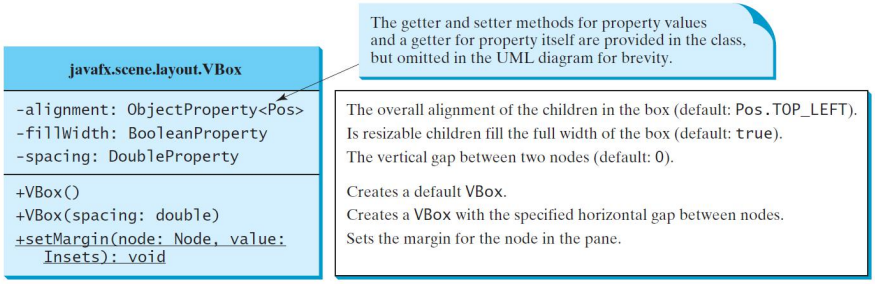



```
FlowPane pane = new FlowPane();
  pane.setPadding(new Insets(11, 12, 13, 14));
  pane.setHgap(5);
  pane.setVgap(5);

  // Place nodes in the pane
  pane.getChildren().addAll(new Label("First Name:"),
    new TextField(), new Label("MI:"));
  TextField tfMi = new TextField();
  tfMi.setPrefColumnCount(1);
  pane.getChildren().addAll(tfMi, new Label("Last Name:"),
    new TextField());

  // Create a scene and place it in the stage
  Scene scene = new Scene(pane, 200, 250);
  primaryStage.setTitle("ShowFlowPane"); // Set the stage title
  primaryStage.setScene(scene); // Place the scene in the stage
  primaryStage.show(); // Display the stage
```


HBox



VBox



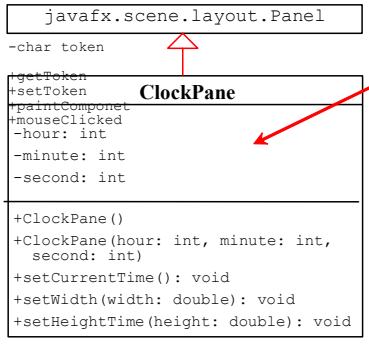
ShowHBoxVBox Run


```
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();
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;
  }
    // Place nodes in the pane
    pane.setTop(getHBox());
    pane.setLeft(getVBox());
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 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;
 }

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


Case Study: The ClockPane Class

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



The `getter` and `setter` methods for these data fields are provided in the class, but omitted in the UML diagram for brevity.

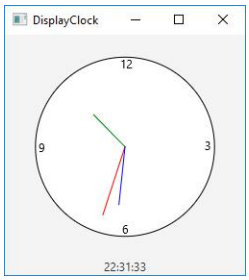
The hour in the clock.
The minute in the clock.
The second in the clock.

Constructs a default clock for the current time.
Constructs a clock with the specified time.

Sets hour, minute, and second for current time.
Sets clock pane's width and repaint the clock,
Sets clock pane's height and repaint the clock,

ClockPane

Use the ClockPane Class



DisplayClock Run

There is a problem with the default installation of Eclipse and getting JavaFX to run. Here's the fix

When you create a JavaFX project, right-click on the project (under the Package Explorer in Eclipse), and select "Properties"

In the window that appears, click on "Java Build Path" on the left side, then "Add External JARs" on the right from _ Libraries tab

Navigate to: C:\Program Files (x86)\Java\jre1.8.0_111\lib\ext
Double-click on jfxrt.jar, which should now appear at the top of your build path
Click on the "Order and Export" tab

Click on jfxrt.jar to highlight it
Then click the button labeled “Top” to move it to the top of the list
Click “OK”

For more help:

https://www.youtube.com/watch?v=5CLmTqQaUws