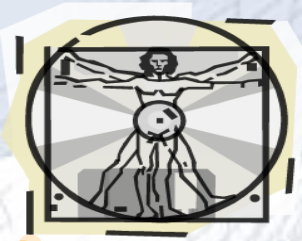
Programing Languages - II
Visual Programming-JavaFX

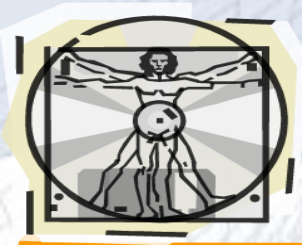
Özgür Koray ŞAHİNGÖZ
Prof.Dr.

Biruni University
Computer Engineering Department



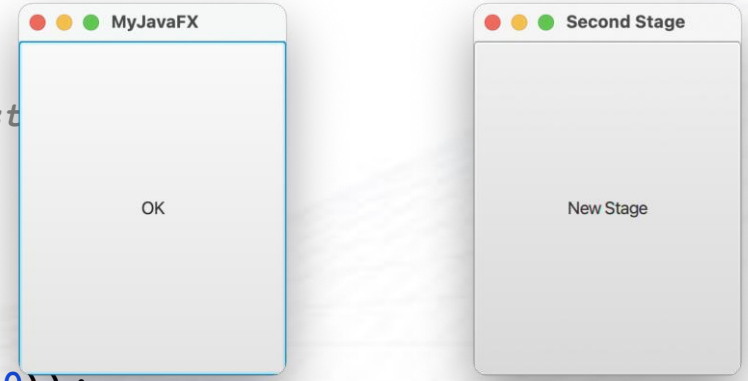
A Simple Program

```
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);
    }
}
```



Multiple Stages

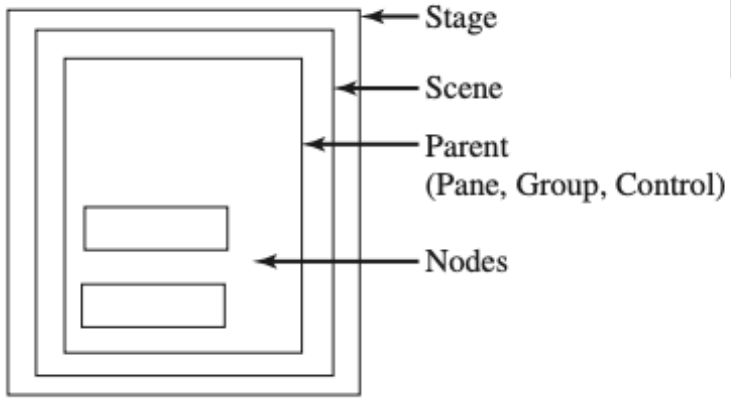
```
public class HelloApplication extends Application {
    @Override
    public void start(Stage primaryStage) throws IOException {

        // Create a scene and place a button in the scene
            Scene scene = new Scene(new Button("OK"), 200, 250);
            primaryStage.setTitle("MyJavaFX"); // Set the stage title
            primaryStage.setScene(scene); // Place the scene in the st
            primaryStage.show(); // Display the stage

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



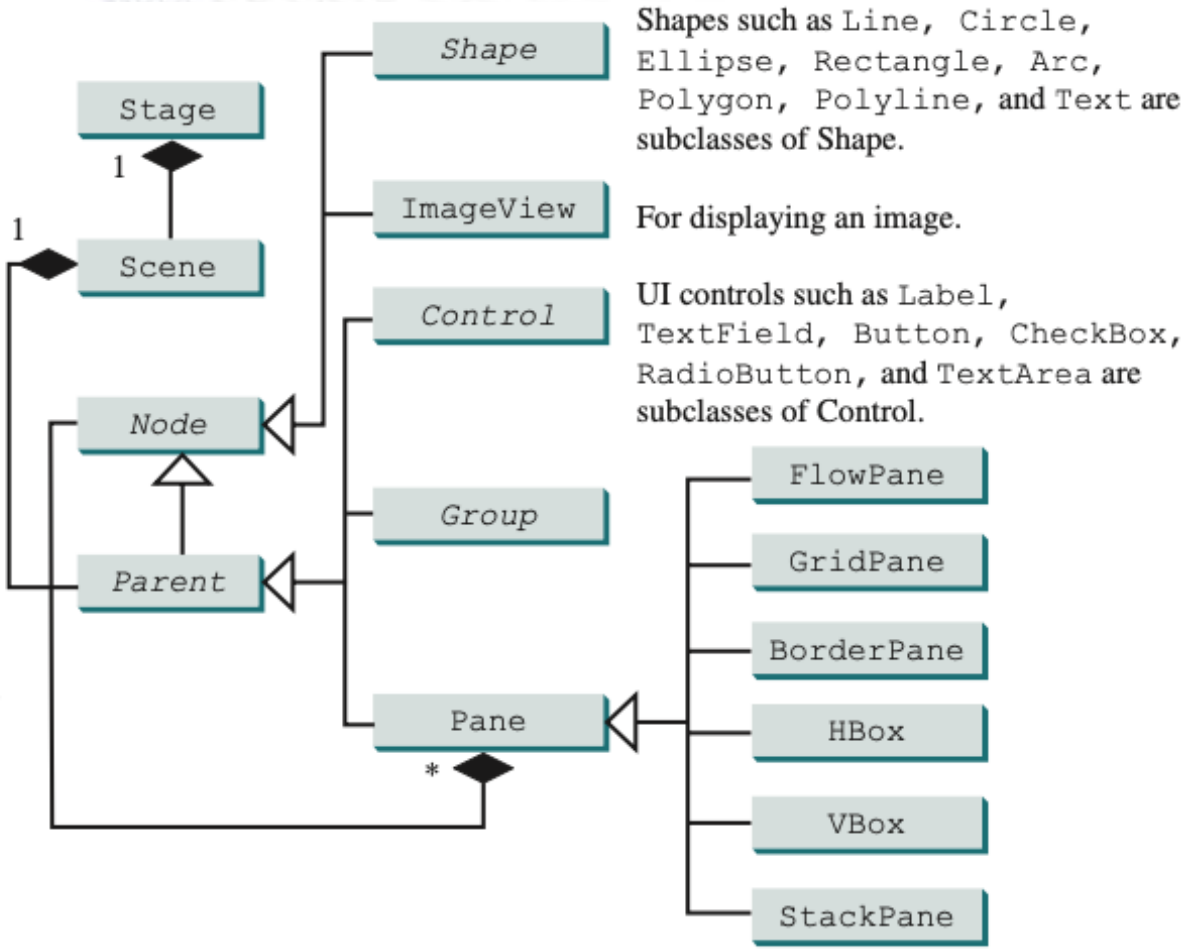
MultipleStageDemo

Pane, Group, UI Control, Shape

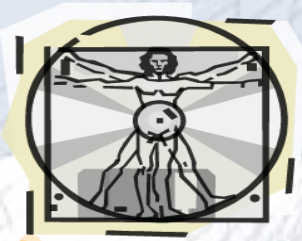
- Relations



(a)

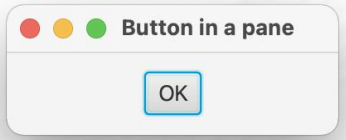


(b)

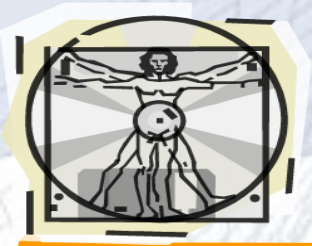


Use of more than one Stage

```
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.stage.Stage;

import java.io.IOException;

public class HelloApplication extends Application {
    @Override
    public void start(Stage primaryStage) throws IOException {

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

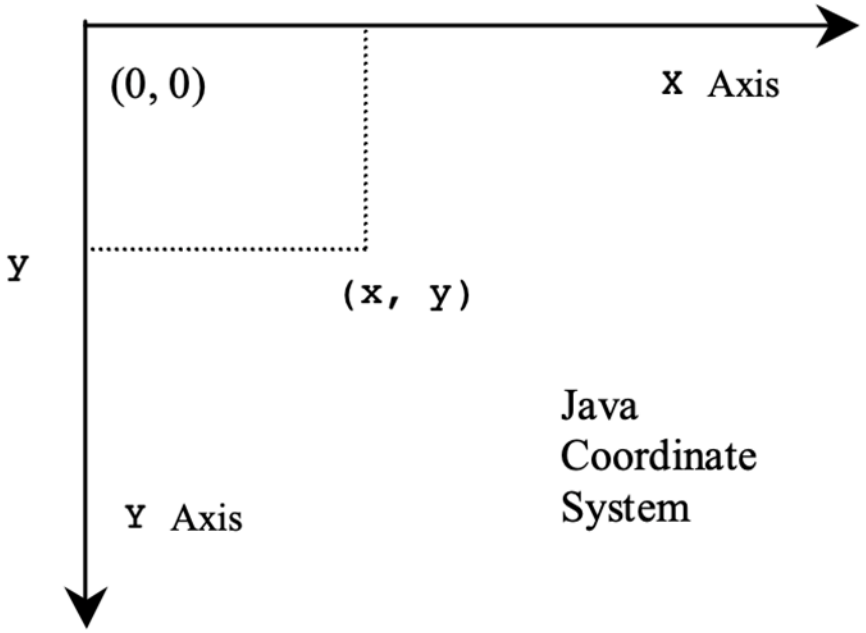


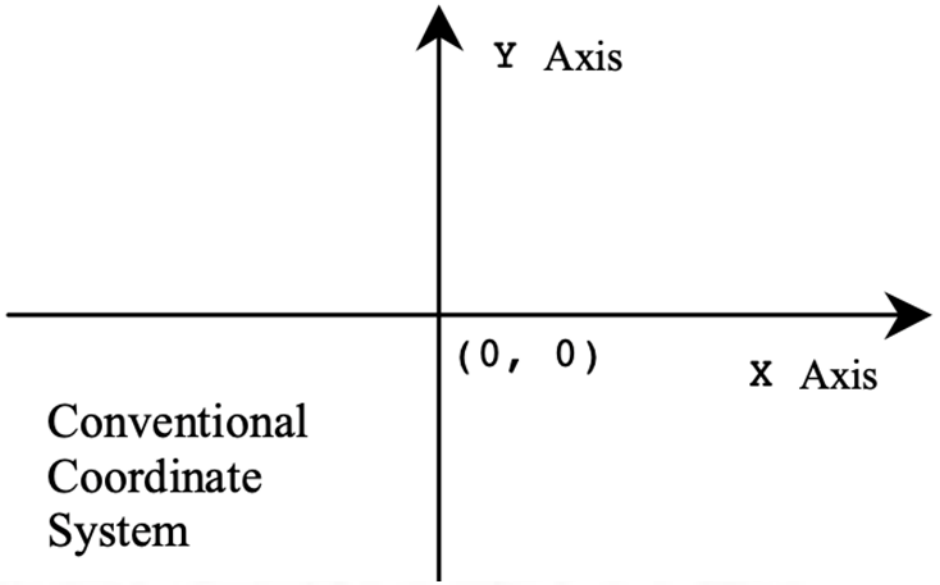
ButtonInPane

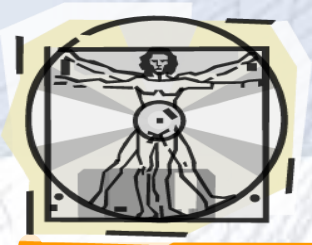


Showing Shapes

- Coordinates in Java

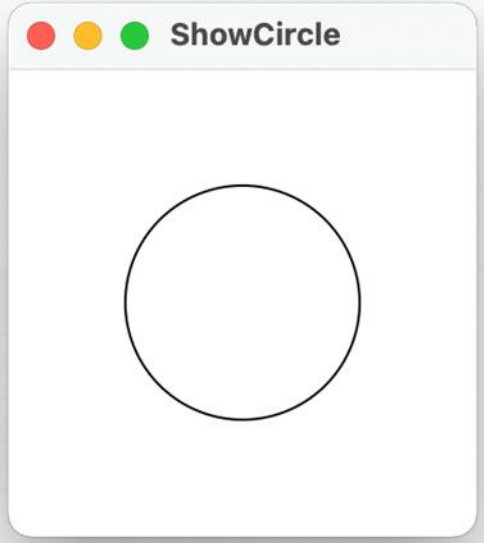




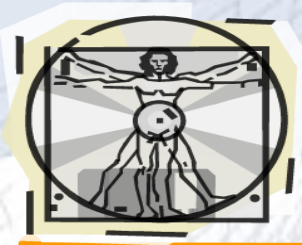


Showing a Shape

```
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(100);
    circle.setCenterY(100);
    circle.setRadius(50);
    circle.setStroke(Color.BLACK); // Set circle stroke color
    circle.setFill(Color.WHITE);

    // 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
  }

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



ShowCircle



Binding

When you enlarge the visual window in the previous program, you will notice that the location of the circle has changed, it cannot stay in the center.If we want the shape to change when the previous screen gets bigger, we can take advantage of this feature.

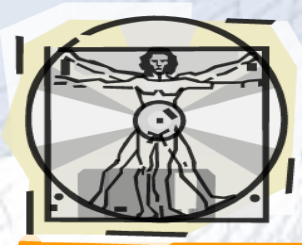
Target.bind(source)

```
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

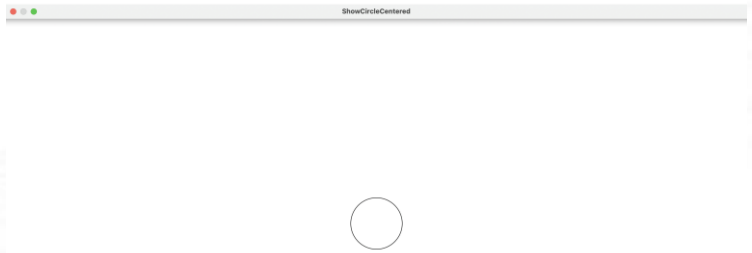
```
public class Circle {

  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

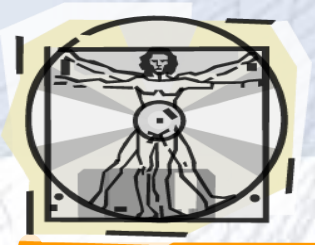


Binding

```
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;
import java.io.IOException;

public class HelloApplication extends Application {
    @Override
    public void start(Stage primaryStage) throws IOException {
        // 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
    }

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

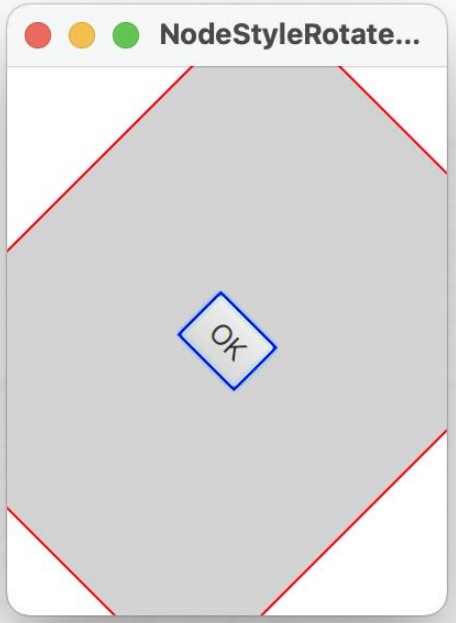


ShowCircleCentered



Nodes

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

import java.io.IOException;

public class HelloApplication extends Application {
    @Override
    public void start(Stage primaryStage) throws IOException {
        // 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); // Rotate pane 45 degrees
        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

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



NodeStyleRotateDemo

Color Class

- Color Class and Methods

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

javafx.scene.paint.Color

```
-red: double
-green: double
-blue: double
-opacity: double
```

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

The red value of this Color (between `0.0` and `1.0`).

The green value of this Color (between `0.0` and `1.0`).

The blue value of this Color (between `0.0` and `1.0`).

The opacity of this Color (between `0.0` and `1.0`).

Creates a `Color` with the specified red, green, blue, and opacity values.

Creates a `Color` that is a brighter version of this `Color`.

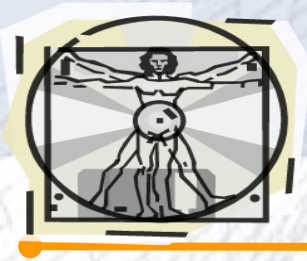
Creates a `Color` that is a darker version of this `Color`.

Creates an opaque `Color` with the specified red, green, and blue values.

Creates a `Color` with the specified red, green, blue, and opacity values.

Creates a `Color` with the specified red, green, and blue values in the range from 0 to 255.

Creates a `Color` with the specified red, green, and blue values in the range from 0 to 255 and a given opacity.



Font Class

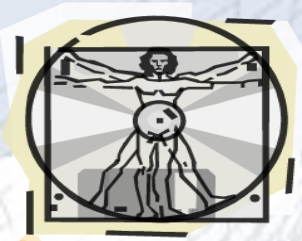
The getter methods for property values are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.text.Font

```
-size: double
-name: String
-family: String
```

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

The size of this font.	
The name of this font.	
The family of this font.	
Creates a `Font` with the specified size.	
Creates a `Font` with the specified full font name and size.	
Creates a `Font` with the specified name and size.	
Creates a `Font` with the specified name, weight, and size.	
Creates a `Font` with the specified name, weight, posture, and size.	
Returns a list of font family names.	
Returns a list of full font names including family and weight.	



Font

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

import java.io.IOException;

public class HelloApplication extends Application {
    @Override
    public void start(Stage primaryStage) throws IOException {
        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);
        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);
    }
}
```

FontDemo

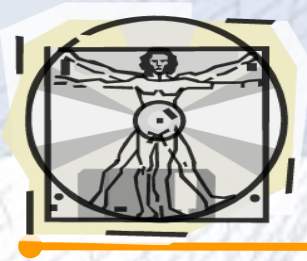


Image Class

javafx.scene.image.Image

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

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

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

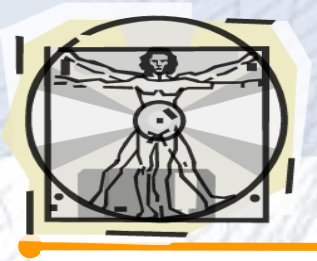
Indicates whether the image is loaded correctly?

The height of the image.

The width of the image.

The approximate percentage of image's loading that is completed.

Creates an `Image` with contents loaded from a file or a URL.



ImageView Class

javafx.scene.image.ImageView

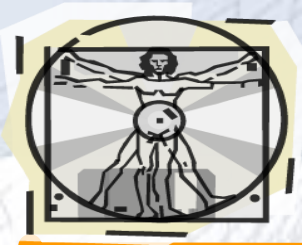
```
-fitHeight: DoubleProperty
-fitWidth: DoubleProperty
-x: DoubleProperty
-y: DoubleProperty
-image: ObjectProperty<Image>
```

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

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

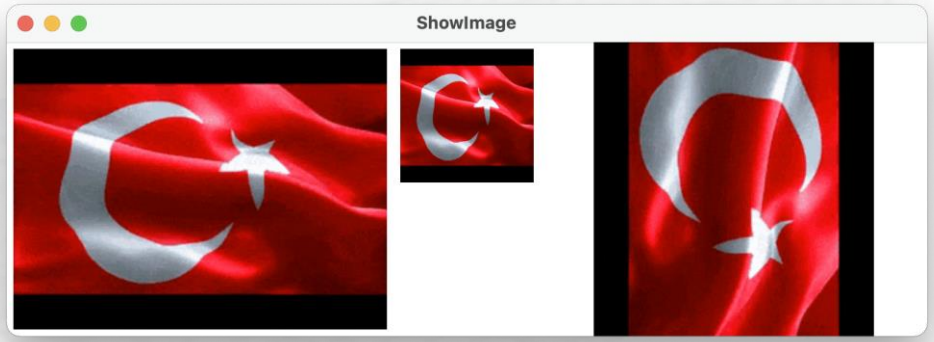
The height of the bounding box within which the image is resized to fit.
The width of the bounding box within which the image is resized to fit.
The x-coordinate of the ImageView origin.
The y-coordinate of the ImageView origin.
The image to be displayed in the image view.

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

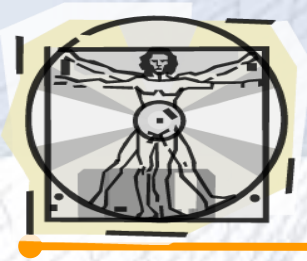


ImageView

```
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Pane;
import javafx.stage.Stage;

import java.io.IOException;

public class HelloApplication extends Application {
    @Override
    public void start(Stage primaryStage) throws IOException {
        // 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("turk.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);

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

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



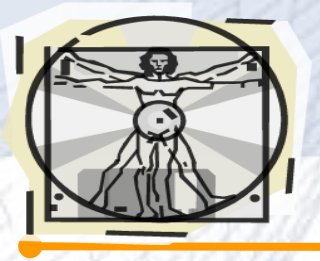
ShowImage



Panes

- JavaFX makes available various pane objects that hold many nodes within the 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 Class

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.

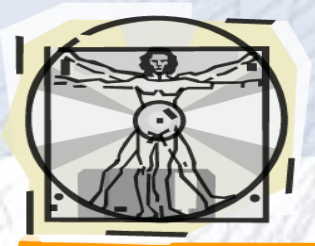
The overall alignment of the content in this pane (default: `Pos.LEFT`).
The orientation in this pane (default: `Orientation.HORIZONTAL`).

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.

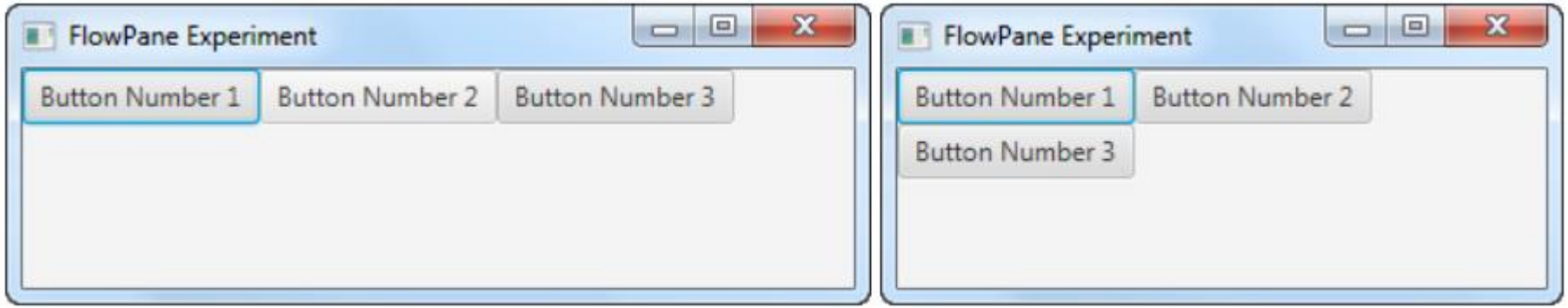
Creates a `FlowPane` with a specified orientation.

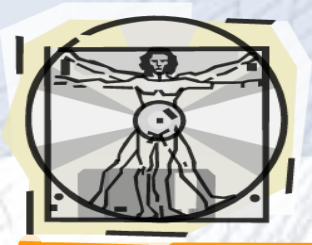
Creates a `FlowPane` with a specified orientation, horizontal gap and vertical gap.



- A JavaFX FlowPane is a layout component which lays out its child components either vertically or horizontally, and which can wrap the components onto the next row or column if there is not enough space in one row. The JavaFX FlowPane layout component is represented by the class javafx.scene.layout.FlowPane

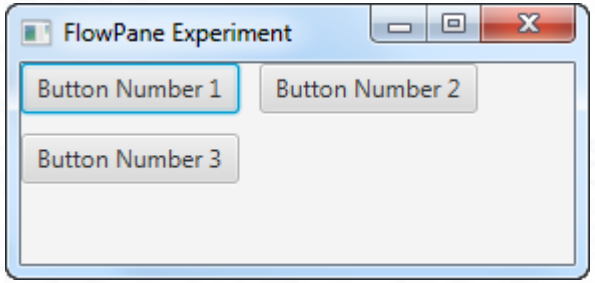
```
Button button1 = new Button("Button Number 1");
Button button2 = new Button("Button Number 2");
Button button3 = new Button("Button Number 3");
FlowPane flowpane = new FlowPane();
flowpane.getChildren().add(button1);
flowpane.getChildren().add(button2);
flowpane.getChildren().add(button3);
```

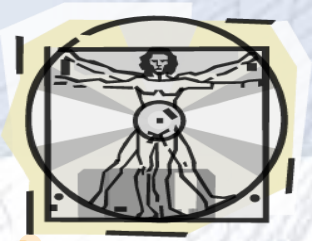




Horizontal and Vertical Spacing

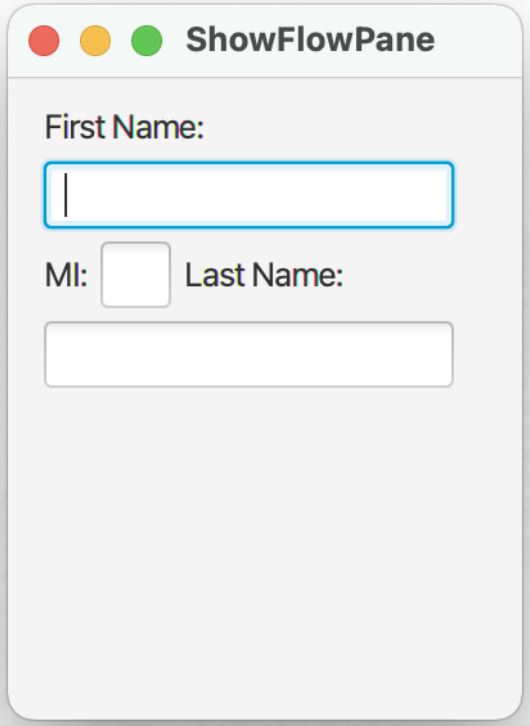
- You can set the horizontal and vertical spacing between the components shown inside a JavaFX FlowPane using its setHGap() and setVGap() methods. Here is an example that shows how to set the horizontal and vertical gap between components in a FlowPane :
- flowpane.setHgap(10);
- flowpane.setVgap(10);



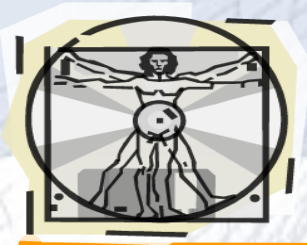


FlowPane

```
package com.example.deneme5;

import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.FlowPane;
import javafx.stage.Stage;

import java.io.IOException;

public class HelloApplication extends Application {
    @Override
    public void start(Stage primaryStage) throws IOException {
        FlowPane pane = new FlowPane();
        pane.setPadding(new Insets(11, 12, 13, 14));
        pane.setHgap(5); // Set hGap to 5px
        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
    }

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



ShowFlowPane



GridPane 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.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.

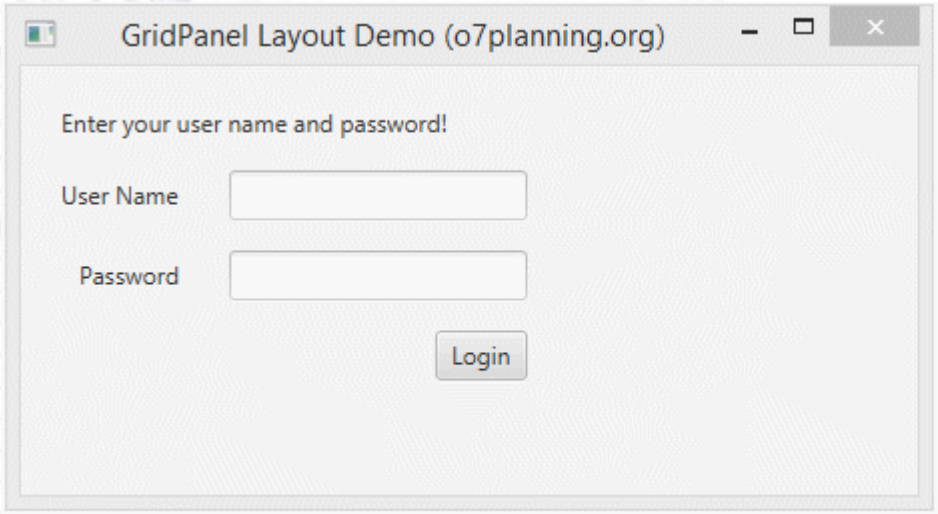
Returns the row index for the specified node.
Sets a node to a new row. This method repositions the node.

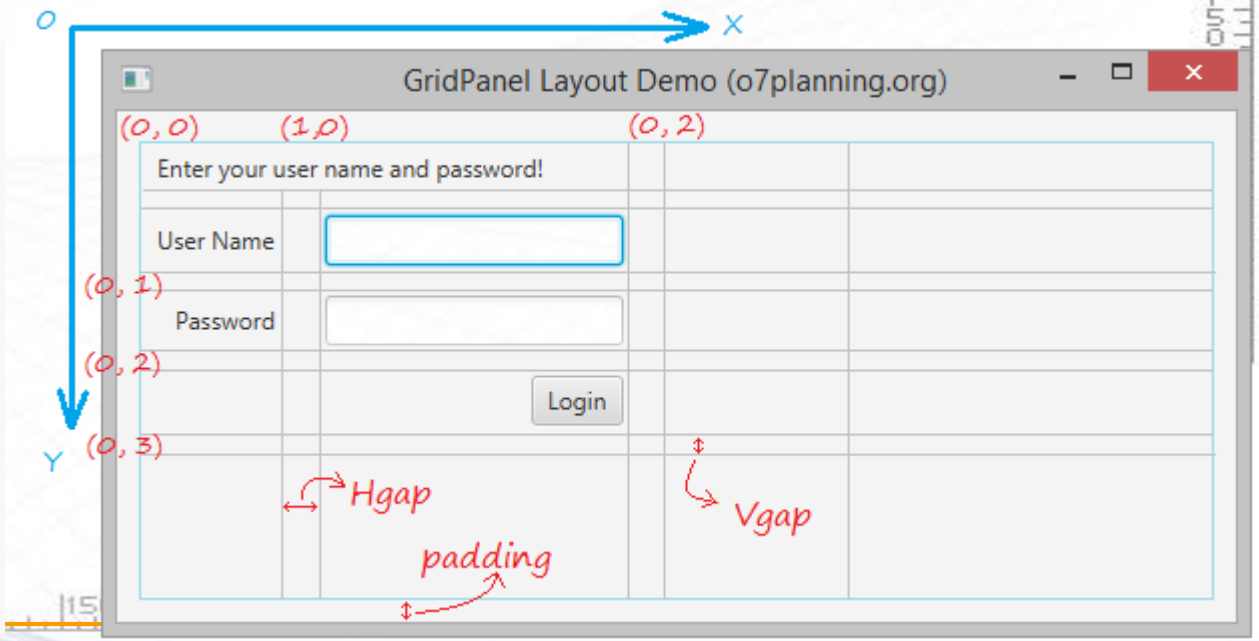
Sets the horizontal alignment for the child in the cell.

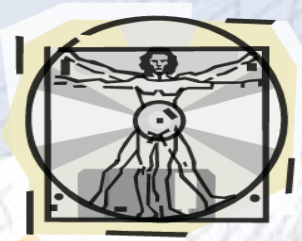
Sets the vertical alignment for the child in the cell.

ShowG

- A JavaFX GridPane is a layout component which lays out its child components in a grid. The size of the cells in the grid depends on the components displayed in the GridPane, but there are some rules. All cells in the same row will have the same height, and all cells in the same column will have the same width. Different rows can have different heights and different columns can have different widths.

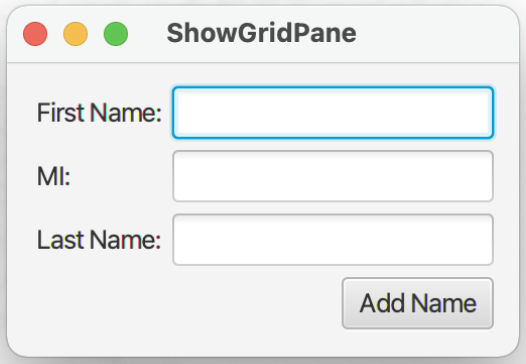




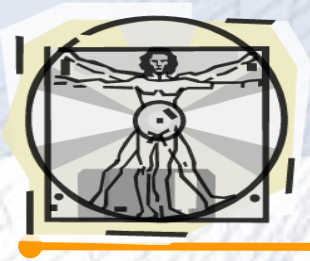


GridPane

```
package com.example.deneme5;

import javafx.application.Application;
import javafx.geometry.*;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

import java.io.IOException;

public class HelloApplication extends Application {
    @Override
    public void start(Stage primaryStage) throws IOException {
        // Create a pane and set its properties
        GridPane pane = new GridPane();
        pane.setAlignment(Pos.CENTER); // Set center alignment
        pane.setPadding(new Insets(11.5, 12.5, 13.5, 14.5));
        pane.setHgap(5.5);
        pane.setVgap(5.5); // Set vGap to 5.5px

        // 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);
    }
}
```



ShowGridPane



BorderPane Class

javafx.scene.layout.BorderPane

```
-top: ObjectProperty<Node>
-right: ObjectProperty<Node>
-bottom: ObjectProperty<Node>
-left: ObjectProperty<Node>
-center: ObjectProperty<Node>
```

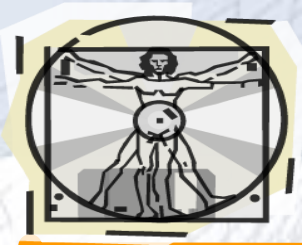
```
+BorderPane()
+setAlignment(child: Node, pos: Pos)
```

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`.

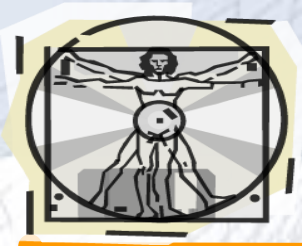
Sets the alignment of the node in the `BorderPane`.



BorderPane

```
package com.example.deneme5;
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

public class HelloApplication 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
stage title
        primaryStage.setScene(scene); // Place the scene in
the stage
        primaryStage.show(); // Display the stage
    }
}

// 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));
    }
}
```

ShowBorderPane



HBox Class

javafx.scene.layout.HBox

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

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

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

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

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

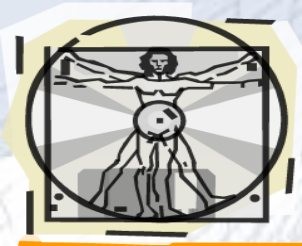
The horizontal gap between two nodes (default: `0`).

Creates a default `HBox`.

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

Sets the margin for the node in the pane.

HBox lays out its children in a single horizontal row. If the hbox has a border and/or padding set, then the contents will be layed out within those insets.



VBox 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.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`).

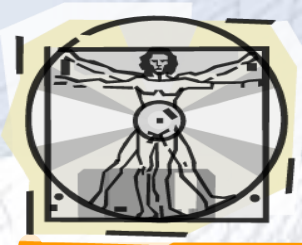
The vertical gap between two nodes (default: 0).

Creates a default VBox.

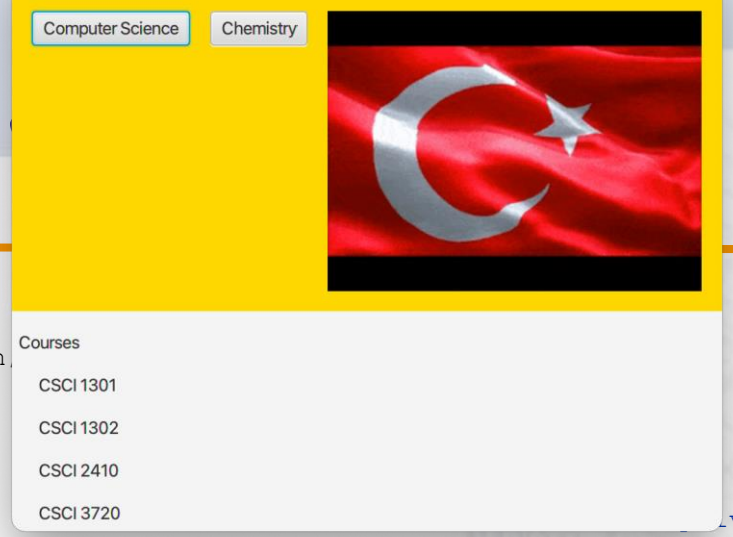
Creates a VBox with the specified horizontal gap between nodes.

Sets the margin for the node in the pane.

VBox lays out its children in a single vertical column. If the vbox has a border and/or padding set, then the contents will be layed out within those insets

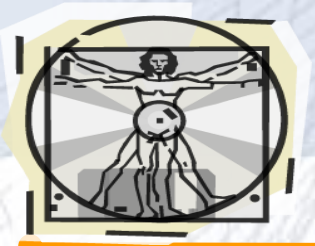


HBox,

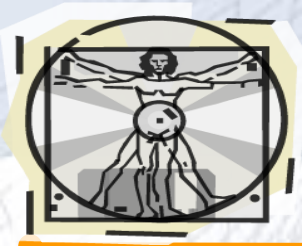



```
package com.example.deneme5;

import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.stage.Stage;
import javafx.scene.image.*;

public class HelloApplication 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); // Create an HBox with 15px spacing
        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("turk.gif"));
        hBox.getChildren().add(imageView);
        return hBox;
    }

    private VBox getVBox() {
        VBox vBox = new VBox(15); // Create a VBox with 15px spacing
        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;
    }

    /**
     * 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);
    }
}
```

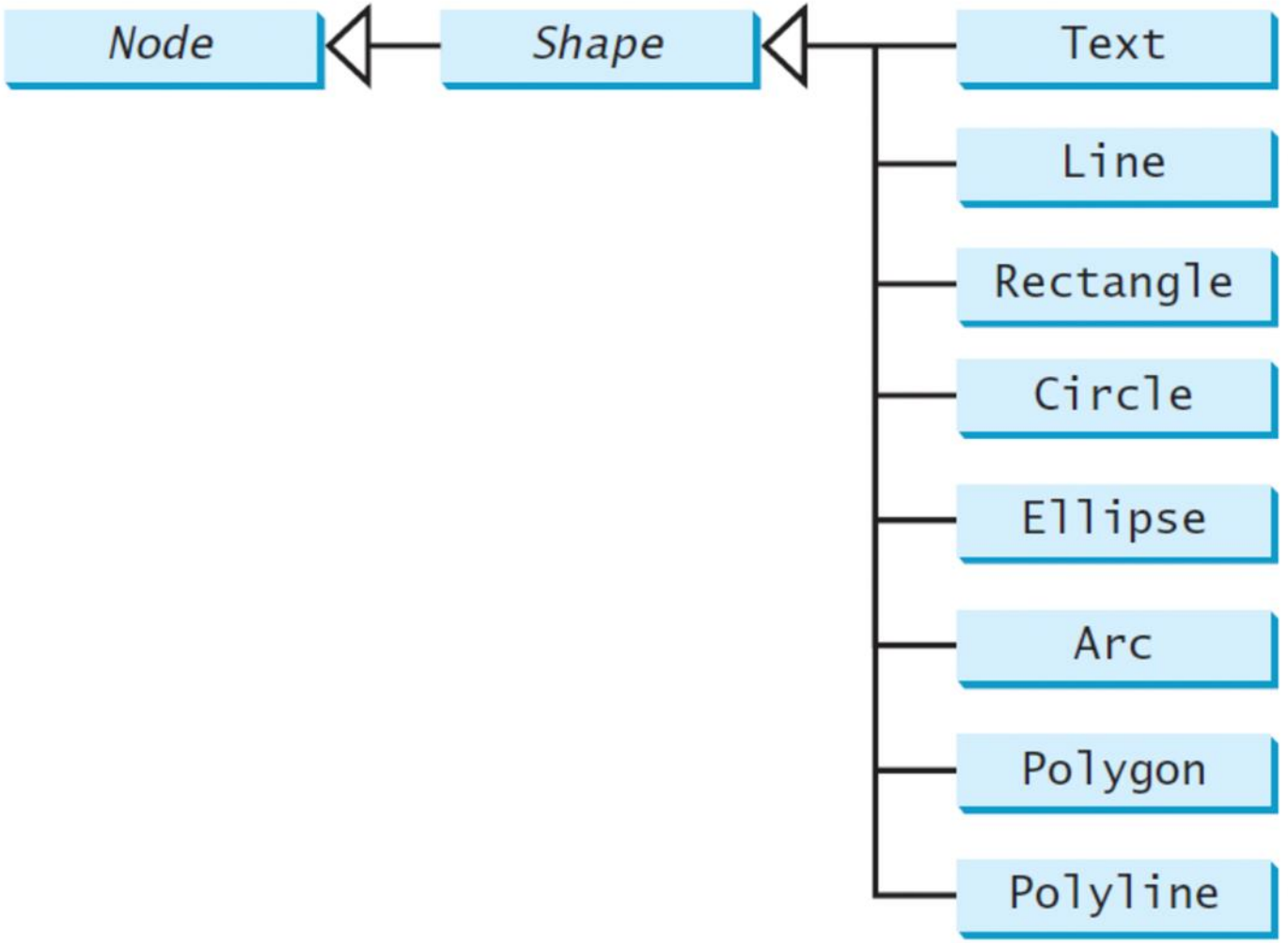
ShowHBoxVBox

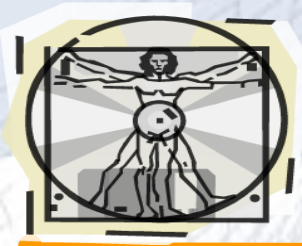


Shapes



Shapes Class





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`).

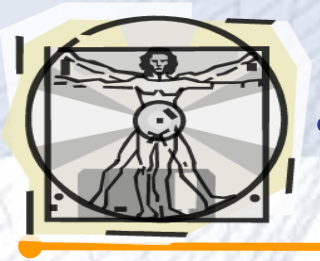
Defines if each line has a line through it (default `false`).

Defines the font for the text.

Creates an empty Text.

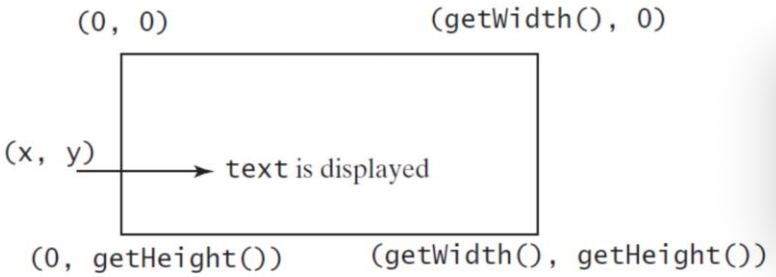
Creates a Text with the specified text.

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



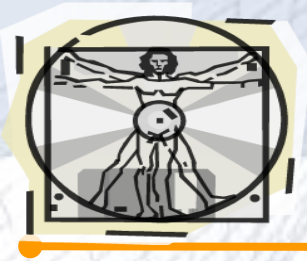
Text

```
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.*;

public class HelloApplication 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); // Underline for text3
        text3.setStrikethrough(true); // Strikethrough for text3
        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
    }

    /**
     * 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);
    }
}
```

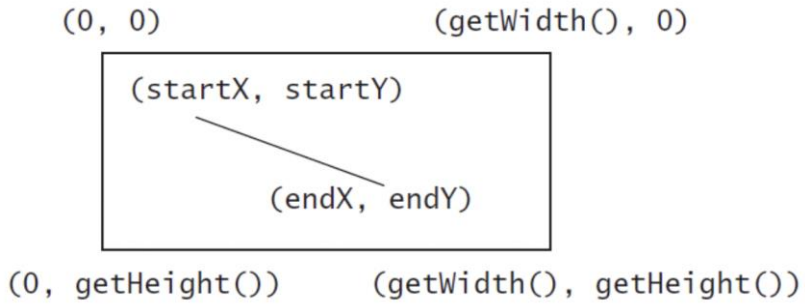


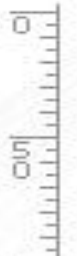
(a) Text(x, y, text)

ShowText



Line





javafx.scene.shape.Line

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

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

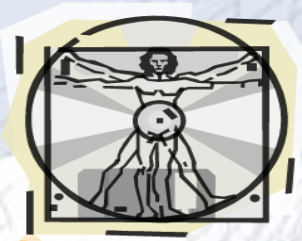
The y-coordinate of the start point.

The x-coordinate of the end point.

The y-coordinate of the end point.

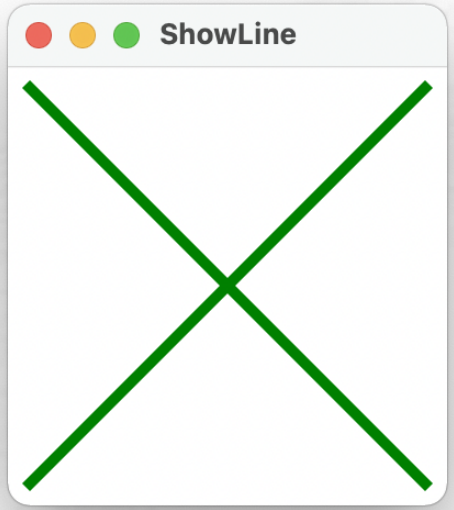
Creates an empty `Line`.

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

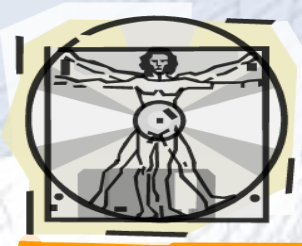


Line

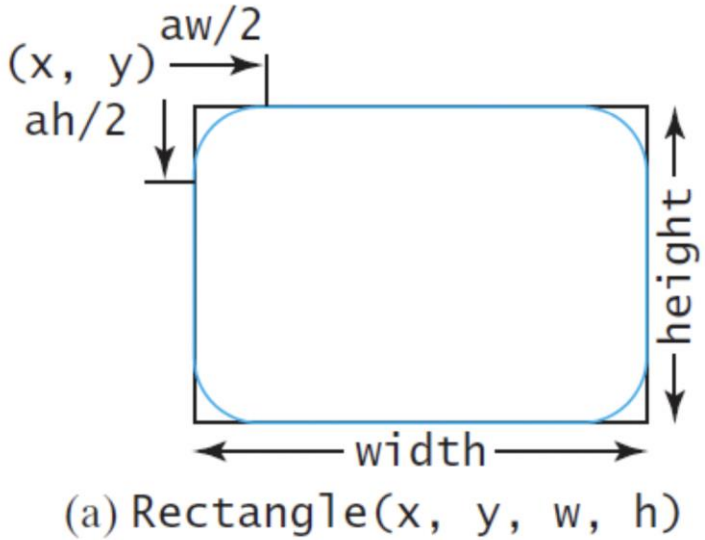
```
package com.example.deneme5;

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 HelloApplication 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);
    }
}
```



ShowLine



Rectangle



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

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 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 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).

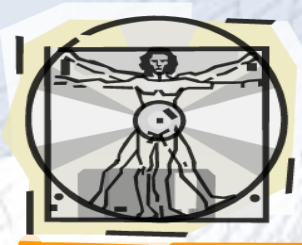
The height of the rectangle (default: 0).

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).

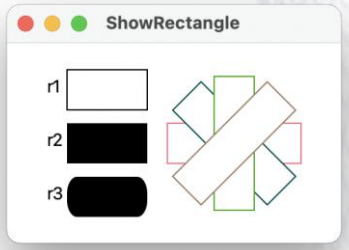
Creates an empty `Rectangle`.

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

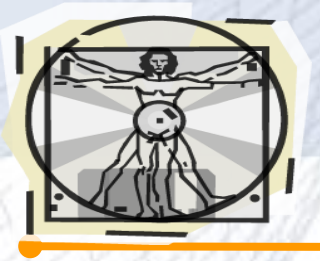


Rectangle

```
package com.example.deneme5;

import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Line;
import javafx.stage.Stage;
import javafx.scene.text.Text;
import javafx.scene.shape.Rectangle;

public class HelloApplication 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
    }

    /**
     * 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);
    }
}
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);
    }
}
```



ShowRectangle



Circle

javafx.scene.shape.Circle

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

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

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

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

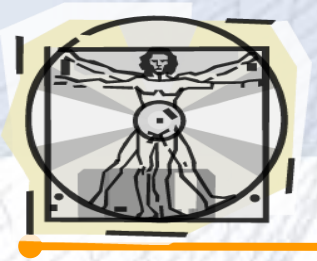
The y-coordinate of the center of the circle (default 0).

The radius of the circle (default: 0).

Creates an empty `Circle`.

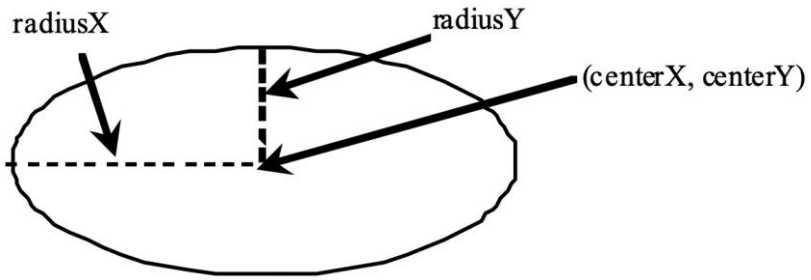
Creates a `Circle` with the specified center.

Creates a `Circle` with the specified center and radius.



Ellipse

- Ellipse, özelliklerini ve metotlarını inceleyelim.



javafx.scene.shape.Ellipse

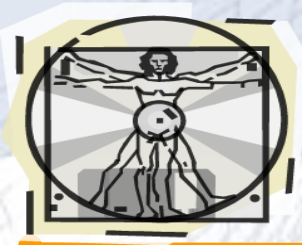
```
-centerX: DoubleProperty
-centerY: DoubleProperty
-radiusX: DoubleProperty
-radiusY: DoubleProperty
```

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

The 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).

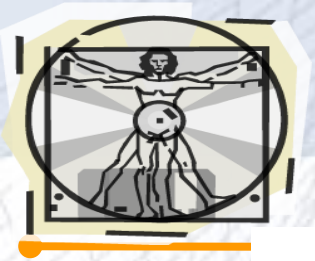
Creates an empty `Ellipse`.
Creates an `Ellipse` with the specified center.
Creates an `Ellipse` with the specified center and radiuses.



Ellipse

```
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.Ellipse;

public class HelloApplication 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 MyEllipse(), 300, 200);
        primaryStage.setTitle("ShowEllipse"); // 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);
    }
}

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);
        }
    }

    @Override
    public void setWidth(double width) {
        super.setWidth(width);
        paint();
    }

    @Override
    public void setHeight(double height) {
        super.setHeight(height);
        paint();
    }
}
```

ShowEllipse



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.

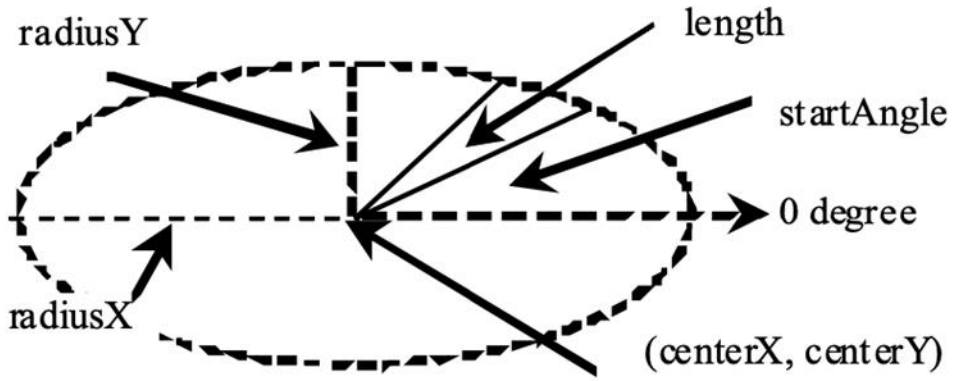
The angular extent of the arc in degrees.

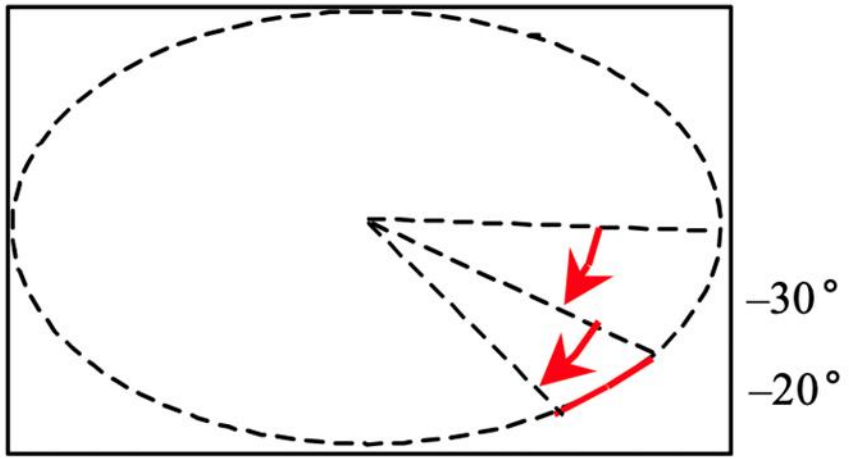
The closure type of the arc (`ArcType.OPEN`, `ArcType.CHORD`, `ArcType.ROUND`).

Creates an empty `Arc`.

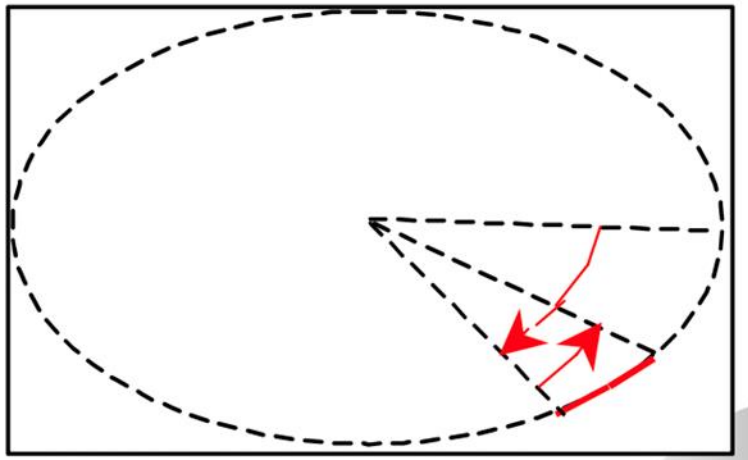
Creates an `Arc` with the specified arguments.

Arc

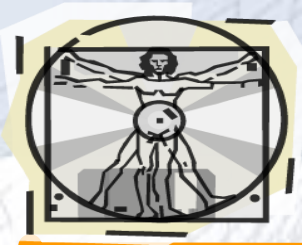




(a) Negative starting angle –30° and negative spanning angle –20°

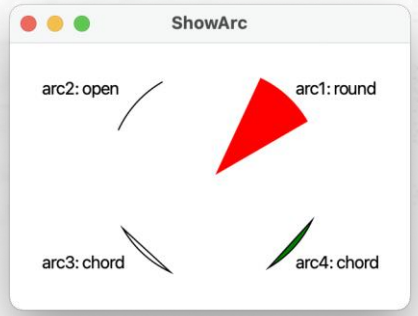


(b) Negative starting angle –50° and positive spanning angle 20°

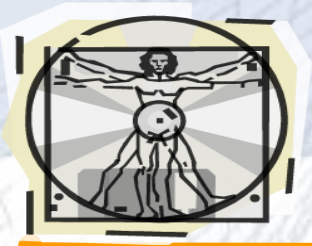


Arc

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.Group;
import javafx.scene.layout.BorderPane;
import javafx.scene.paint.Color;
import javafx.stage.Stage;
import javafx.scene.shape.Arc;
import javafx.scene.shape.ArcType;
import javafx.scene.text.Text;

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

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



ShowArc



Polygon

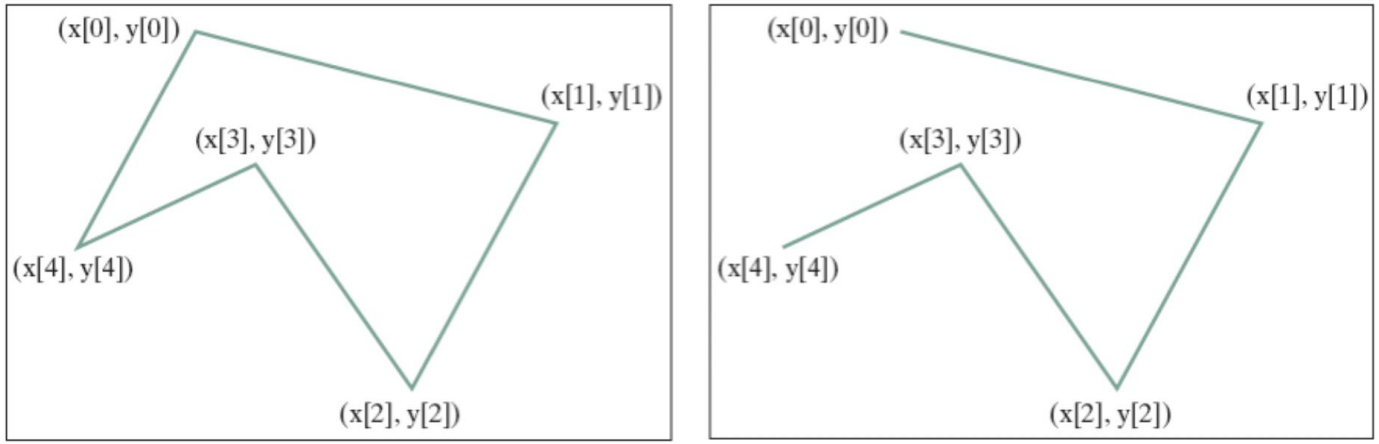
javafx.scene.shape.Polygon
+Polygon()
+Polygon(double... points)
+getPoints(): ObservableList<Double>

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.

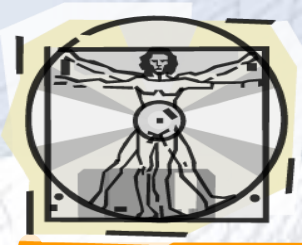
Creates an empty polygon.

Creates a polygon with the given points.

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

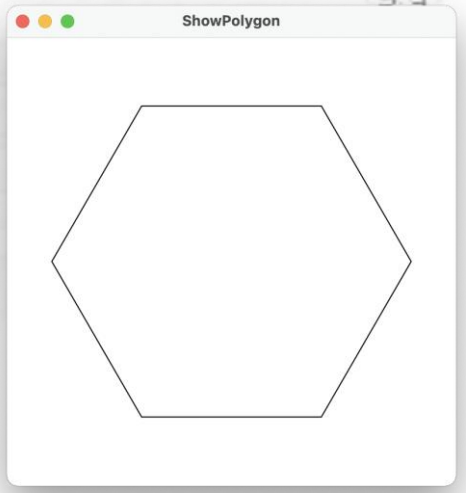


(a) Polygon (b) Polyline

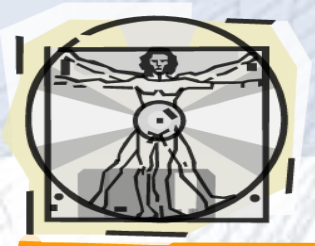


Polygon

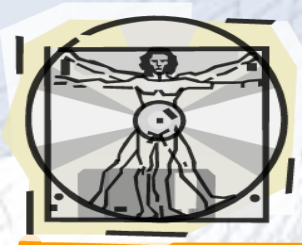
```
import javafx.application.Application;
import javafx.collections.ObservableList;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.stage.Stage;
import javafx.scene.shape.Polygon;

public class HelloApplication 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 MyPolygon(), 400, 400);
        primaryStage.setTitle("ShowPolygon"); // Set the title
        primaryStage.setScene(scene); // Place the scene in stage
        primaryStage.show(); // Display the stage
    }

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

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(); // Clear pane
        getChildren().add(polygon);
    }

    @Override
    public void setWidth(double width) {
        super.setWidth(width);
        paint();
    }

    @Override
    public void setHeight(double height) {
        super.setHeight(height);
        paint();
    }
}
```



ShowPolygon



JavaFX Scene Builder



JavaFX Scene Builder

javafx scene builder

javafx scene builder - Google'da Ara

Google javafx scene builder

Tümü Videolar Görseller Haberler Alışveriş Daha fazla Araçlar

Yaklaşık 275.000 sonuç bulundu (0,57 saniye)

İpucu: Yalnızca **Türkçe** sonuçları arayın. Arama dilinizi Tercihler sayfasında belirtebilirsiniz.

https://www.oracle.com › javase ▾ Bu sayfanın çevirisini yap

JavaFX Scene Builder Information - Oracle

JavaFX Scene Builder is a visual layout tool that lets users quickly design JavaFX application user interfaces, without coding. Users can drag and drop UI ...

https://www.oracle.com › java ▾ Bu sayfanın çevirisini yap

JavaFX Scene Builder 1.x Archive - Oracle

JavaFX Scene Builder Archive. Go to the main Oracle Java Archive Downloads page. WARNING: These versions of **JavaFX Scene Builder** may include components that ...

https://gluonhq.com › products ▾ Bu sayfanın çevirisini yap

Scene Builder - Gluon

Scene Builder works with the **JavaFX** ecosystem – official controls, community projects, and Gluon offerings including Gluon Mobile, Gluon Desktop, and Gluon ...

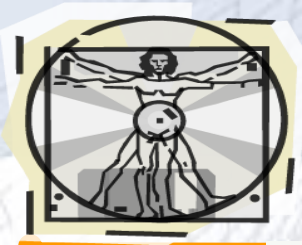
Scene Builder

Yazılım

JavaFX Scene Builder is **a visual layout tool that lets users quickly design JavaFX application user interfaces, without coding.**

https://www.oracle.com › java › technologies › javase ›

JavaFX Scene Builder Information -



JavaFX Scene Builder

gluonhq.com

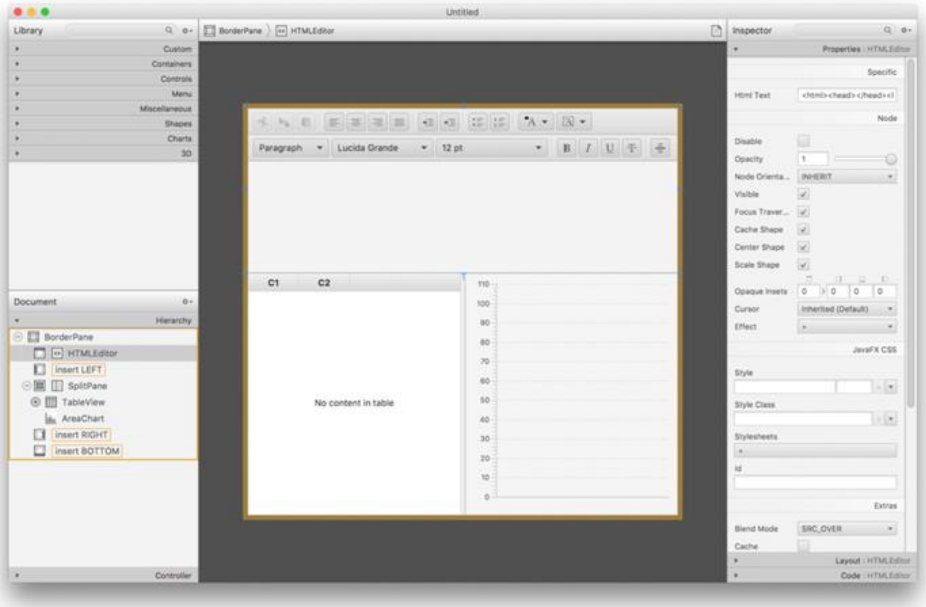
Getting Started with JavaFX | Scene Builder - Gluon | JavaFX - Gluon

GLUON

Products | Developers | Pricing | Services | Insights | Contact

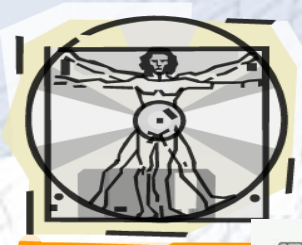
Home » Products » Scene Builder

Scene Builder



Drag & Drop,
Rapid Application
Development.

Download Now



JavaFX Scene Builder

gluonhq.com

Getting Started with JavaFX | Thanks - Gluon | JavaFX - Gluon

GLUON Products Developers Pricing Services Insights Contact

Home » Products » Scene Builder » Thanks

Thanks

While Scene Builder is downloading, let us introduce you to our JavaFX Long Term Support offering.

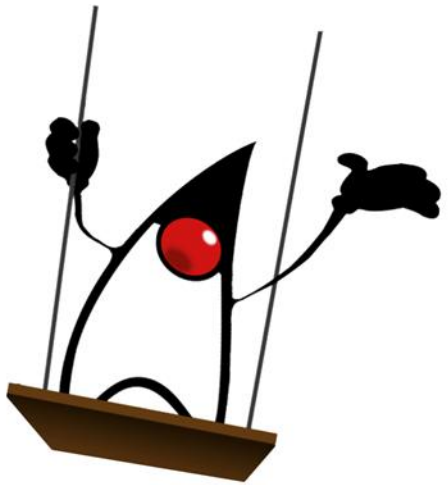
Gluon and JavaFX are inseparable – our team have contributed to, and led, some of the most critical open source libraries, written some of the primary books, maintained the most important tools, based our commercial software offerings on it, and even co-led the OpenJFX project itself. We are not ashamed to say that the team at Gluon are world experts in JavaFX.

We provide JavaFX 11 long term support, in which we provide regular builds of JavaFX 11 including security updates and critical patches.

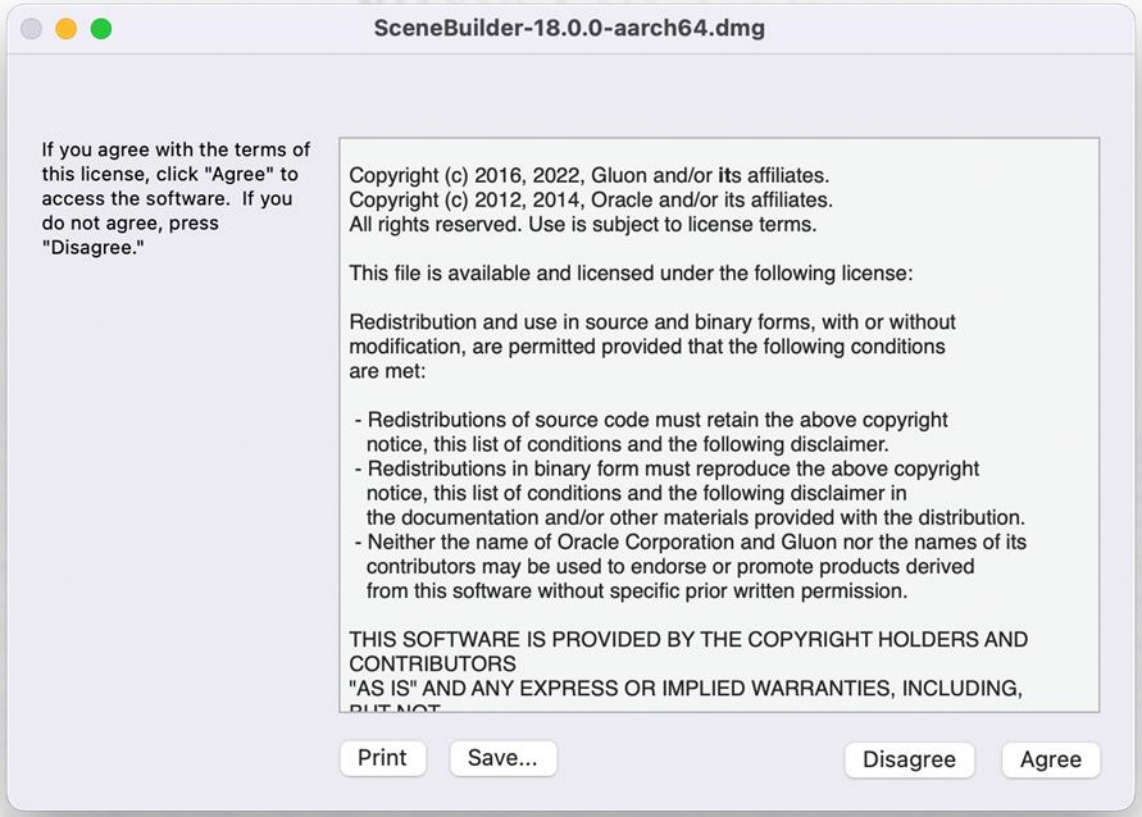
Request LTS pricing information

Request consulting services pricing

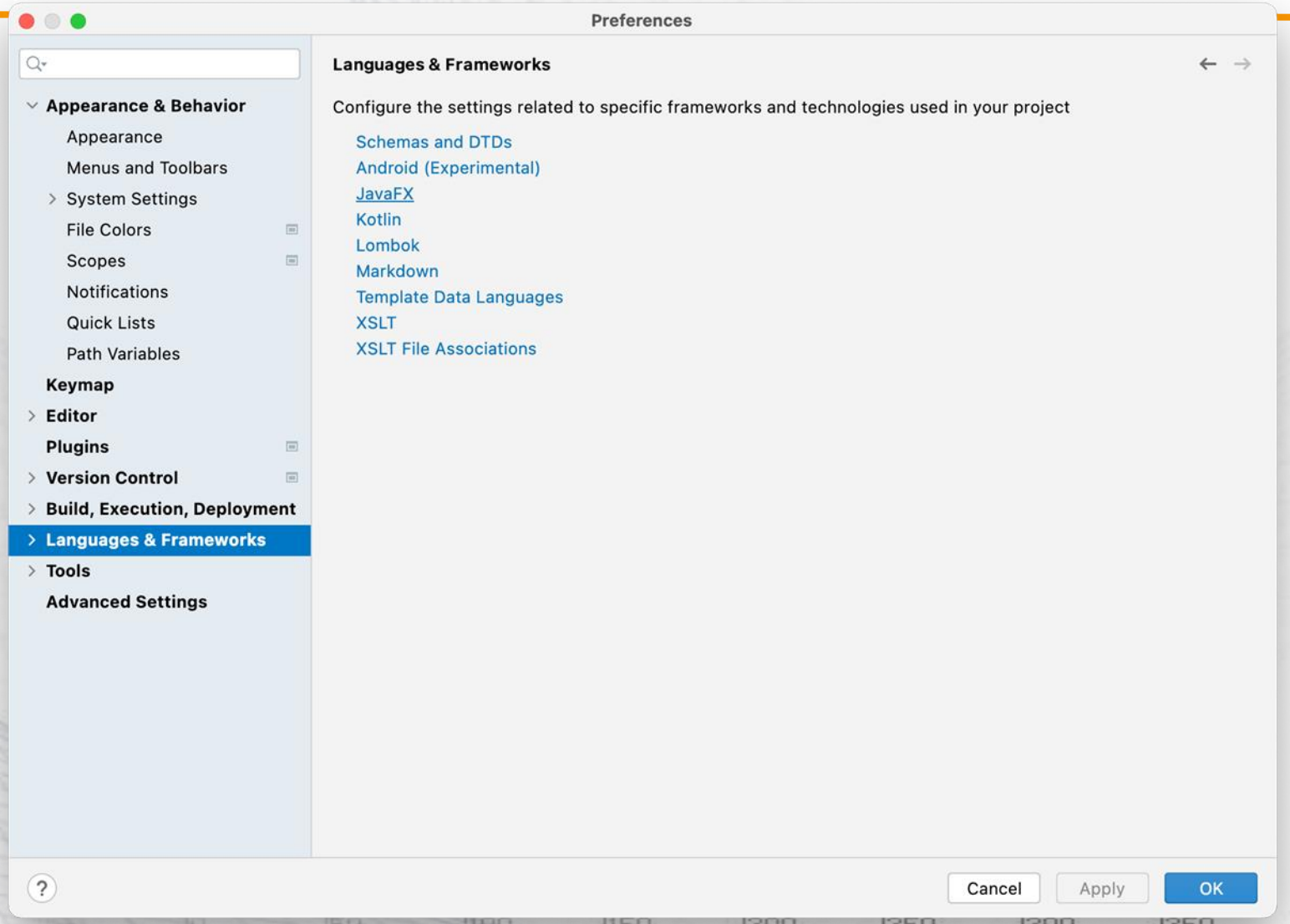
More information about JavaFX on mobile

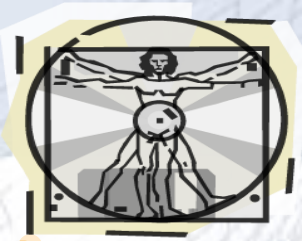


JavaFX Scene Builder

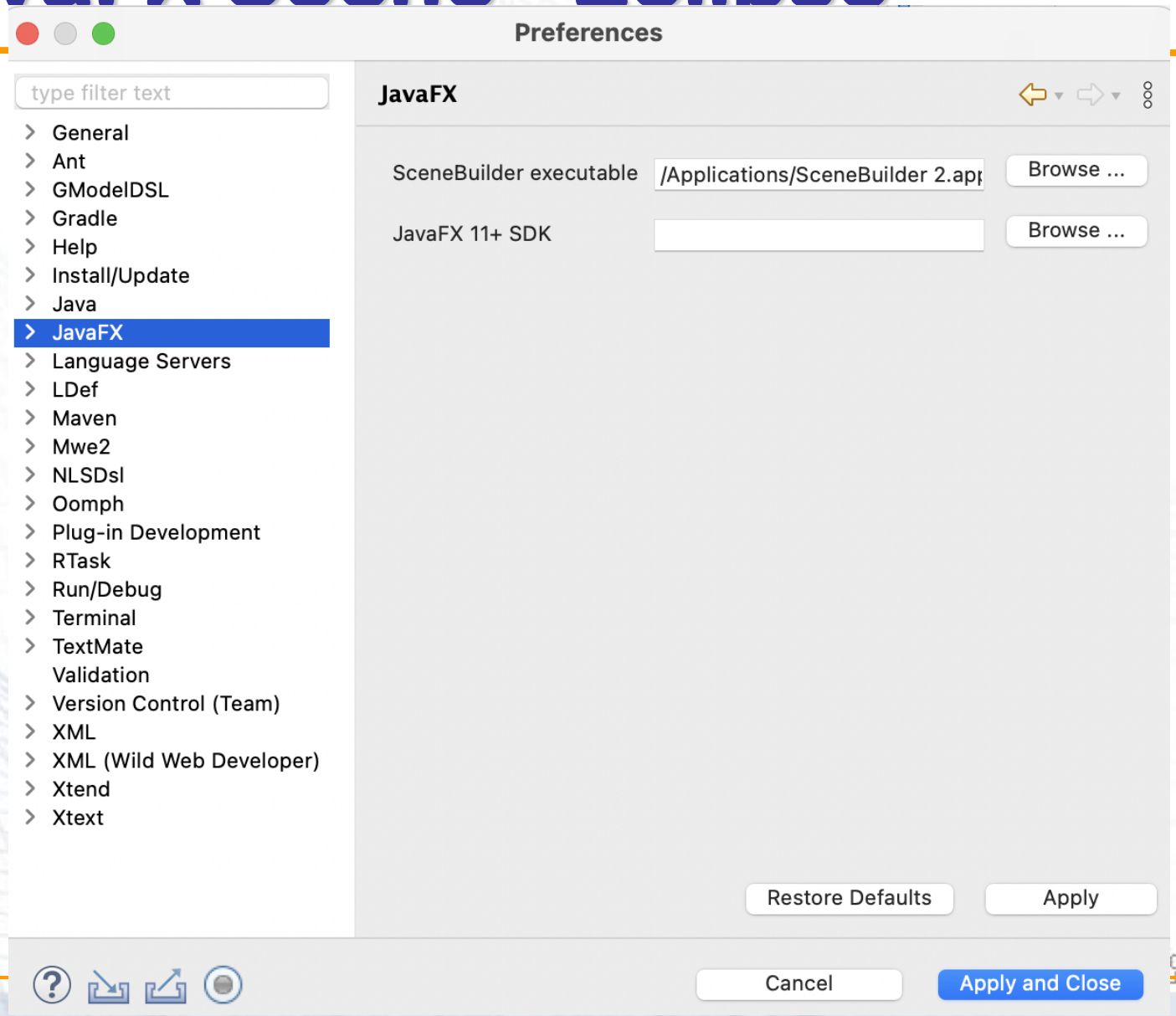


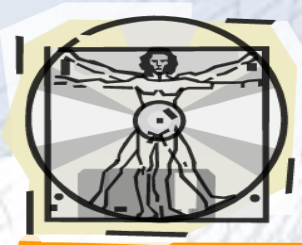
JavaFX Scene Builder - IntelliJ IDEA



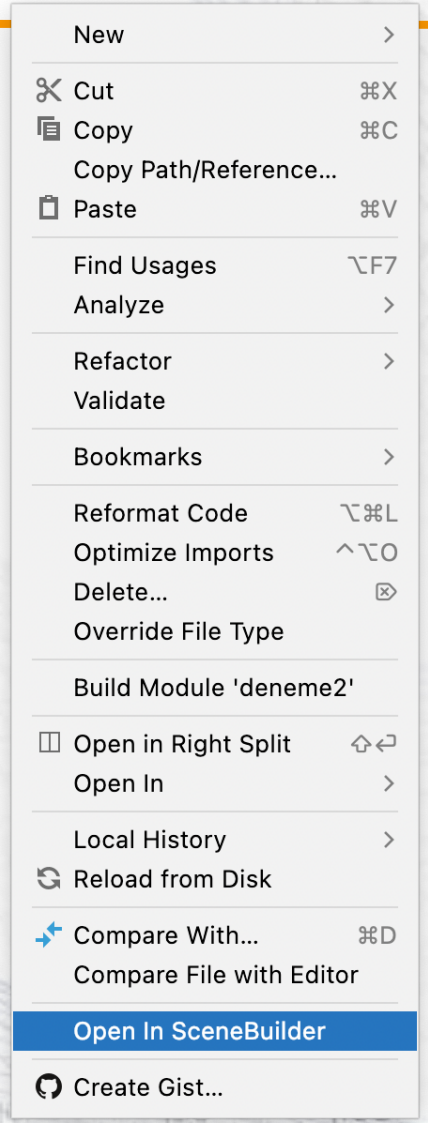


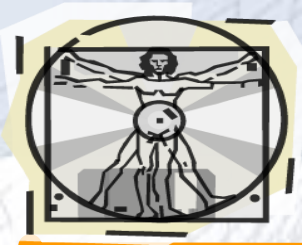
JavaFX Scene - Eclipse



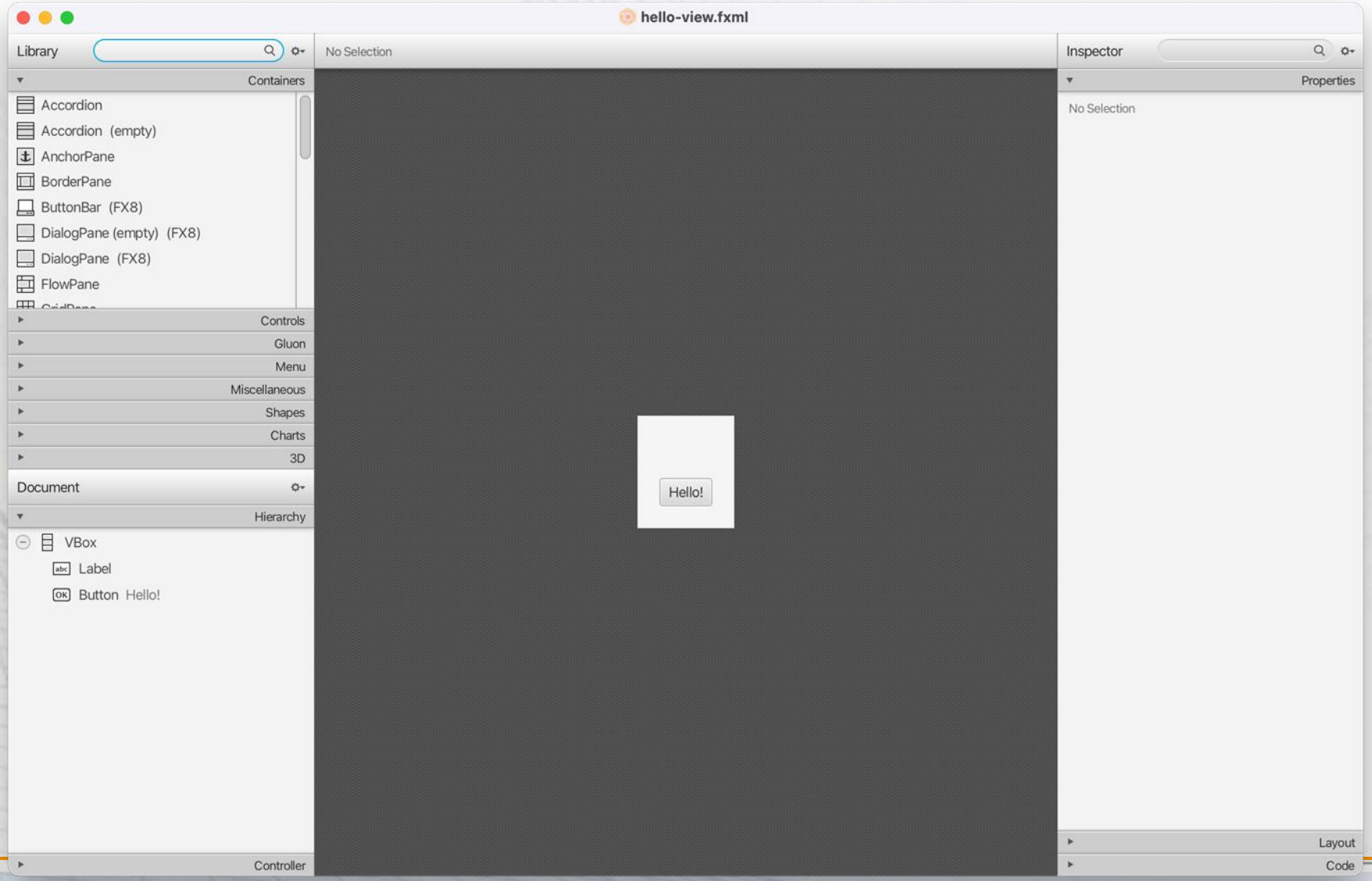


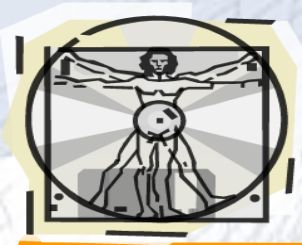
JavaFX Scene Builder



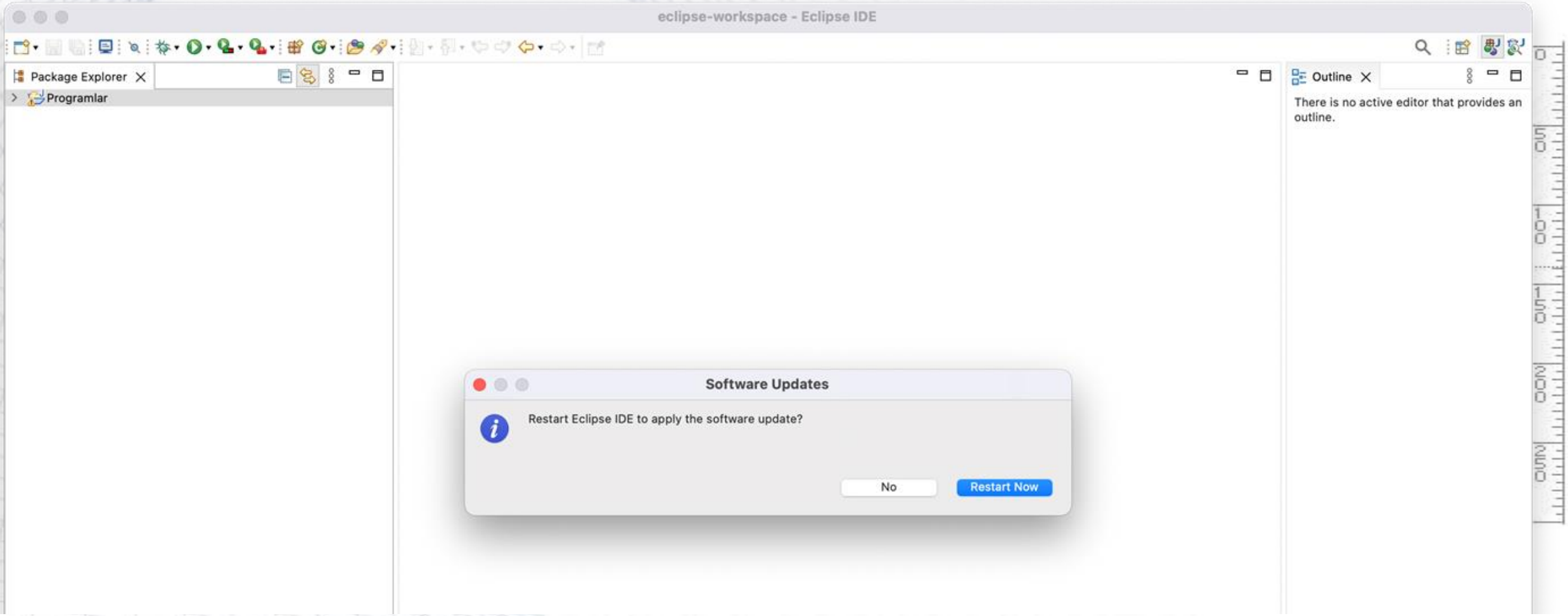


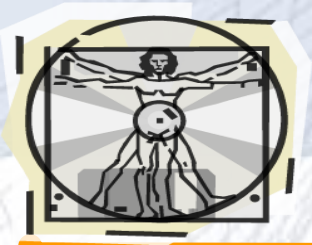
JavaFX Scene Builder





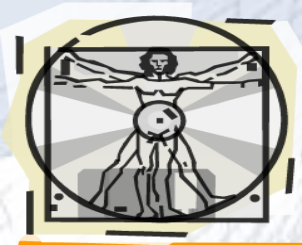
JavaFX Scene Builder



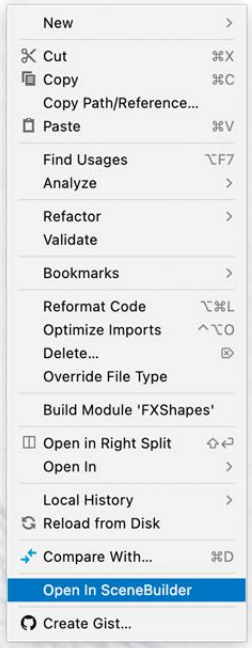


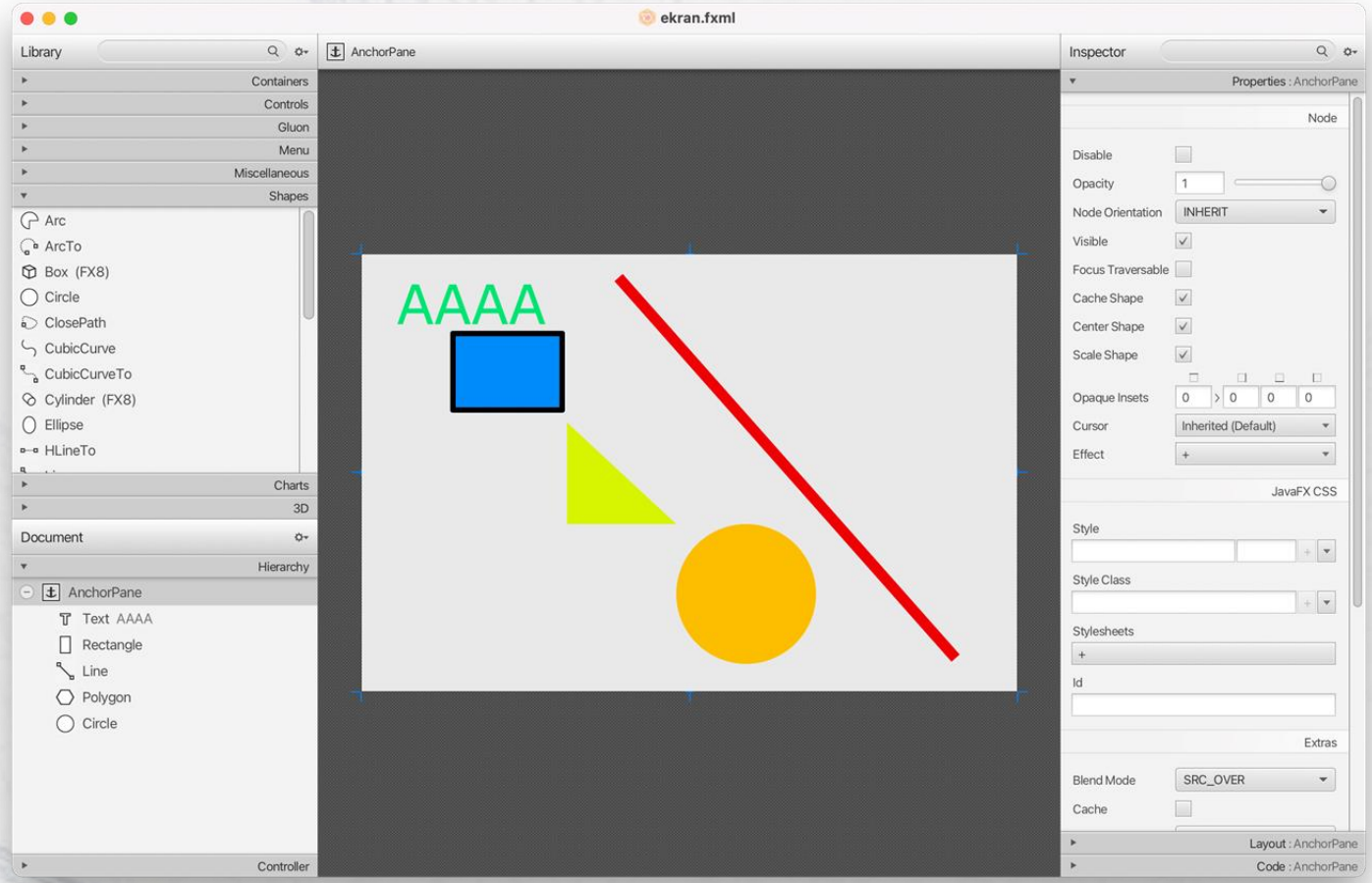
Example-1

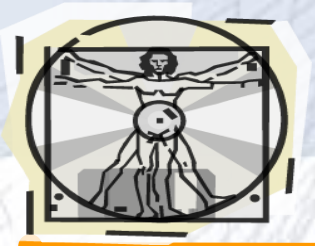
```
public class HelloApplication extends Application {
    @Override
    public void start(Stage stage) throws IOException {

        FXMLLoader fxmlLoader = new
FXMLLoader(HelloApplication.class.getResource("ekran.fxml"));

        Scene scene = new Scene(fxmlLoader.load(), Color.LIGHTSKYBLUE);
        stage.setTitle("Hello!");
        stage.setScene(scene);
        stage.show();
    }

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



Example


New
Cut ⌘X
Copy ⌘C
Copy Path/Reference...
Paste ⌘V
Find Usages ⌥F7
Analyze
Refactor
Validate
Bookmarks
Reformat Code ⌥⌘L
Optimize Imports ^⌥O
Delete...
Override File Type
Build Module 'FXShapes'
Open in Right Split ⇧↵
Open In
Local History
Reload from Disk
Compare With... ⌘D
Open In SceneBuilder
Create Gist...


ekran.fxml
Library
AnchorPane
Inspector
Containers
Controls
Gluon
Menu
Miscellaneous
Shapes
Arc
ArcTo
Box (FX8)
Circle
ClosePath
CubicCurve
CubicCurveTo
Cylinder (FX8)
Ellipse
HLineTo
Charts
3D
Document
Hierarchy
AnchorPane
Text AAAA
Rectangle
Line
Polygon
Circle
Controller
AAAA
Properties : AnchorPane
Node
Disable
Opacity
Node Orientation
INHERIT
Visible
Focus Traversable
Cache Shape
Center Shape
Scale Shape
Opaque Insets
Cursor
Inherited (Default)
Effect
JavaFX CSS
Style
Style Class
Stylesheets
Id
Extras
Blend Mode
SRC_OVER
Cache
Layout : AnchorPane
Code : AnchorPane

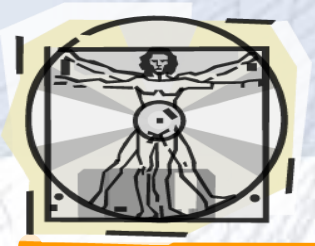


Example

- Bir şekli yukarı, aşağı, sağa, sola hareket ettiren uygulama geliştirelim.

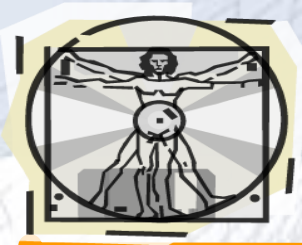
- Button'ların OnAction event'leri ile bağlantı kuracağız.
- Oluşturduğumuz .fxml dosyasını aşağıdaki kod ile çağıracağız.

```
    Parent root =
FXMLLoader.load(getClass().getResource("Main.fxml"));
```



Example

```
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

import java.io.IOException;

public class HelloApplication extends Application {
    @Override
    public void start(Stage stage) throws Exception
{
        try {
            Parent root =
FXMLLoader.load(getClass().getResource("Main.fxml"))
;
            Scene scene = new Scene(root);
            stage.setTitle("Dear Students!");
            stage.setScene(scene);
            stage.show();
        }  catch (Exception e){
            e.printStackTrace();}
    }


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



Example

```
package com.example.deneme4;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.shape.Circle;

public class Controller {
    @FXML
    private Circle daire;
    private double x;
    private double y;

    public void yukariya(ActionEvent e){
        daire.setCenterY(y=y-10);
        //System.out.println("Yukarıya");
    }
    public void asagiya(ActionEvent e){
        daire.setCenterY(y=y+10);
        //System.out.println("Aşağıya");
    }
    public void sola(ActionEvent e){
        daire.setCenterX(x=x-10);
        //System.out.println("Sola");
    }
    public void saga(ActionEvent e){
        daire.setCenterX((x=x+10));
        //System.out.println("Sağa");
    }
```