Introduction to JavaFX

JavaFX is a modern GUI (Graphical User Interface) toolkit for Java that allows developers to create rich, interactive desktop applications. It is the successor to the older Swing library and provides a more flexible and visually appealing platform for developing client applications.

Key Features of JavaFX

1. Scene Graph Architecture
   - JavaFX uses a scene graph where all UI elements (nodes) like buttons, shapes, and controls are organized hierarchically in a tree structure.
2. Rich UI Controls
   - Offers a wide range of pre-built UI controls such as Button, Label, TextField, TableView, ListView, etc.
3. Layouts (Panes)
   - Provides layout managers like VBox, HBox, BorderPane, GridPane, StackPane to arrange UI elements efficiently.
4. Shapes and Graphics
   - Supports 2D shapes (Rectangle, Circle, Line, Polygon) and allows custom drawing and animations.
5. CSS and Styling Support
   - UI elements can be styled using CSS, making applications visually attractive.
6. Animation and Multimedia
   - Built-in support for animations, transitions, and multimedia (audio/video) makes JavaFX ideal for modern, dynamic applications.
7. Event Handling
   - JavaFX provides a robust event-handling mechanism for user interactions like mouse clicks, key presses, and gestures.

Why Use JavaFX?

- Cross-platform: Works on Windows, Mac, Linux.
- Modern look and feel compared to Swing.
- Simplifies desktop GUI development with built-in support for graphics, media, and animation.

- Integrates seamlessly with Java and existing libraries.

Simple Example of JavaFX:

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

public class HelloJavaFX extends Application {
  @Override
  public void start(Stage stage) {
    Label label = new Label("Hello, JavaFX!");
    Scene scene = new Scene(label, 300, 200);
    stage.setScene(scene);
    stage.setTitle("JavaFX Example");
    stage.show();
  }

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

Explanation:

- Application: Base class for JavaFX programs.
- Stage: Main window.
- Scene: Container for UI elements.
- Label: Simple text element.

Core JavaFX Lecture Notes

1. Introduction to JavaFX

- **JavaFX** is a platform for creating rich client applications using **Java**.
- It provides **GUI (Graphical User Interface)** capabilities with modern UI controls, graphics, and animations.
- JavaFX is part of **JDK 8 and later**, but from JDK 11 onward, it needs to be added as an external library.

Key Features:

- Scene graph-based architecture.
- CSS styling support.
- Multimedia, animation, and 3D graphics support.

2. Basic Structure of a JavaFX Program

All JavaFX programs extend the **Application** class and override the start() method.

Template:

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

public class HelloJavaFX extends Application {
  @Override
  public void start(Stage primaryStage) {
    // Create UI elements
    Label label = new Label("Hello, JavaFX!");

    // Create scene and add UI elements
    Scene scene = new Scene(label, 400, 200);

    // Set stage properties
    primaryStage.setTitle("Basic JavaFX Example");
    primaryStage.setScene(scene);
    primaryStage.show();
  }

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

Explanation:

- **Application**: Base class for JavaFX applications.
- **start(Stage primaryStage)**: Main entry point to set up UI.
- **Scene**: Container for all UI elements.
- **Stage**: Window for the scene.

3. Panes in JavaFX

Panes are layout containers used to arrange UI elements in a scene.
Common Panes:

Pane	Description
Pane	Base class, manual positioning using setLayoutX and setLayoutY.
StackPane	Places children on top of each other.
VBox	Arranges children vertically.
HBox	Arranges children horizontally.
BorderPane	Divides layout into top, bottom, left, right, center.
GridPane	Arranges children in a grid of rows and columns.

Example with VBox and HBox:

```
import javafx.application.Application;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.layout.HBox;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;


public class PaneExample extends Application {

    @Override

    public void start(Stage stage) {

        Button b1 = new Button("Button 1");

        Button b2 = new Button("Button 2");

        Button b3 = new Button("Button 3");


        HBox hbox = new HBox(10, b1, b2); // 10px spacing
        VBox vbox = new VBox(15, hbox, b3); // 15px spacing vertically


        Scene scene = new Scene(vbox, 300, 150);
        stage.setScene(scene);
        stage.setTitle("Pane Example");
        stage.show();
    }


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

4. UI Controls in JavaFX

UI controls are ready-made interactive elements, e.g., buttons, labels, text fields, checkboxes, combo boxes, etc.

Common Controls:

- **Label**: Displays text.
- **Button**: Clickable button.
- **TextField**: Single-line text input.
- **TextArea**: Multi-line text input.
- **CheckBox**: Selection box.
- **RadioButton**: Selection among options.
- **ComboBox**: Drop-down list.

Example with Controls:

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class UIControlExample extends Application {
    @Override
    public void start(Stage stage) {
        Label label = new Label("Enter your name:");
        TextField tf = new TextField();
        Button btn = new Button("Submit");

        btn.setOnAction(e -> label.setText("Hello, " + tf.getText() + "!"));
        VBox vbox = new VBox(10, label, tf, btn);
        Scene scene = new Scene(vbox, 300, 150);


        stage.setScene(scene);
        stage.setTitle("UI Controls Example");
        stage.show();
    }

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

5. Shapes in JavaFX

JavaFX provides several built-in **shapes** for drawing graphics:

- **Rectangle**: Rectangle rect = new Rectangle(width, height);
- **Circle**: Circle circle = new Circle(radius);
- **Ellipse**: Ellipse ellipse = new Ellipse(rx, ry);
- **Line**: Line line = new Line(startX, startY, endX, endY);
- **Polygon / Polyline**: Polygon poly = new Polygon(x1, y1, x2, y2, ...);

Example with Shapes:

```
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.scene.shape.Rectangle;
import javafx.stage.Stage;

public class ShapeExample extends Application {
    @Override
    public void start(Stage stage) {
        Rectangle rect = new Rectangle(50, 50, 100, 70);
        rect.setFill(Color.BLUE);

        Circle circle = new Circle(200, 100, 50);
        circle.setFill(Color.RED);

        Pane pane = new Pane(rect, circle);

        Scene scene = new Scene(pane, 400, 200);
        stage.setScene(scene);
        stage.setTitle("Shapes Example");
        stage.show();
    }

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

---

6. Summary

- JavaFX is used to create **GUI applications** in Java.
- **Basic structure**: Application → Stage → Scene → UI Controls / Shapes.
- **Panes** control layout and positioning.
- **UI Controls** provide interactivity.
- **Shapes** allow drawing custom graphics.