UNIT- 5 : Introduction to JavaFX

1. Introduction to JavaFX

JavaFX is a **powerful framework** for developing **rich Internet applications (RIA)** and **modern desktop GUIs** using Java.

Definition

JavaFX is a **set of graphics and media packages** that enables developers to design, create, test, debug, and deploy rich client applications that operate consistently across multiple platforms.

Key Features

- Rich GUI (Graphical User Interface)
- Built-in support for 2D and 3D graphics, audio, and video
- CSS-based styling
- FXML for UI layout (XML-based)
- Property binding (easy data synchronization)
- Integration with Swing and AWT
- Cross-platform support (Windows, macOS, Linux)

JavaFX Architecture

JavaFX Applications
JavaFX Scene Graph (Nodes, UI)
JavaFX API (UI Controls, Media)
Prism (Graphics Engine)
Glass Windowing Toolkit
Operating System Layer

2. Structure of a JavaFX Application

A JavaFX program must **extend the javafx.application.Application** class.

Lifecycle Methods

Method	Description
init()	Called before the application starts (used for initialization).
start(Stage primaryStage)	Called to set up the GUI. Every application must override this.
stop()	Called when the application is about to close.

Basic Example: JavaFX Hello World

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

public class HelloFX extends Application {
  @Override
  public void start(Stage primaryStage) {
    Label label = new Label("Hello, JavaFX!");
    Scene scene = new Scene(label, 300, 200);
    primaryStage.setTitle("My First JavaFX Program");
    primaryStage.setScene(scene);
    primaryStage.show();
  }

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

Explanation:

- Stage → Top-level container (window)
- Scene → Holds all UI elements (a container for nodes)
- Node → Any visual element in JavaFX (e.g., Label, Button, TextField)

3. JavaFX Scene Graph

Definition

A **Scene Graph** is a hierarchical tree of nodes used to build a JavaFX application's user interface.

Hierarchy

Stage → Scene → Nodes

Types of Nodes:

1. **Root Node** – Base of the Scene graph.
2. **Branch Node** – Containers like Pane, HBox, VBox, etc.
3. **Leaf Node** – UI controls like Button, Label, TextField.

4. JavaFX Containers (Layout Panes)

Layout	Description	Example
HBox	Horizontal arrangement	HBox hbox = new HBox(10);
VBox	Vertical arrangement	VBox vbox = new VBox(10);
BorderPane	Divides into top, bottom, left, right, center	BorderPane bp = new BorderPane();
GridPane	Arranges in grid form	grid.add(new Button("OK"), 0, 0);
FlowPane	Flow layout (left to right, top to bottom)	FlowPane fp = new FlowPane();
StackPane	Stacks elements on top of each other	StackPane sp = new StackPane();

5. JavaFX UI Controls

Common Controls

Control	Description	Example
Label	Displays text	new Label("Welcome");
Button	Clickable button	new Button("Submit");
TextField	Single-line text input	new TextField();
PasswordField	Hides text with dots	new PasswordField();
TextArea	Multi-line input	new TextArea();
CheckBox	Toggle option	new CheckBox("Accept");
RadioButton	One option among many	new RadioButton("Male");
ComboBox	Drop-down list	new ComboBox<>();
ListView	List of items	new ListView<>();

Example: Button Action Event

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

public class ButtonExample extends Application {
  @Override
  public void start(Stage primaryStage) {
    Label label = new Label("Click the Button!");
    Button btn = new Button("Click Me");

    btn.setOnAction(e -> label.setText("Button Clicked!"));

    VBox vbox = new VBox(10, label, btn);
    Scene scene = new Scene(vbox, 300, 150);
    primaryStage.setScene(scene);
    primaryStage.setTitle("JavaFX Button Example");
    primaryStage.show();
  }
```

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

6. JavaFX Event Handling

Definition

JavaFX uses an **Event-driven model** — each user action (mouse click, key press) triggers an event.

Types of Events

- ActionEvent – Button click
- KeyEvent – Keyboard press/release
- MouseEvent – Mouse actions (click, drag)

Example: Handling an Action Event

```
btn.setOnAction(e -> System.out.println("Button pressed!"));
```

7. JavaFX Property Binding

Definition

Binding synchronizes two properties, so when one changes, the other updates automatically.

Example:

```
DoubleProperty x = new SimpleDoubleProperty(10);
DoubleProperty y = new SimpleDoubleProperty(20);

y.bind(x.multiply(2));
System.out.println(y.get()); // Output: 20
x.set(15);
System.out.println(y.get()); // Output: 30
```

8. JavaFX FXML

Definition

FXML is an XML-based language to **design UI separately** from Java logic (like HTML for UI).

Advantages

- Clear separation of design and logic.
- Easier for designers and developers to collaborate.
- Reduces Java code size.

Example:
sample.fxml

```
<VBox xmlns:fx="http://javafx.com/fxml" fx:controller="MyController">
   <Label text="Hello FXML!" />
   <Button text="Click Me" onAction="#handleButtonClick" />
</VBox>
```

MyController.java

```
public class MyController {
   public void handleButtonClick() {
      System.out.println("FXML Button Clicked!");
   }
}
```

9. JavaFX CSS Styling

Definition

JavaFX allows applying **CSS (Cascading Style Sheets)** to style UI components.

Example:
style.css

```
.button {
   -fx-background-color: #0096C7;
   -fx-text-fill: white;
   -fx-font-weight: bold;
}
```

Applying in code:

```
scene.getStylesheets().add("style.css");
```

10. JavaFX Charts and Media (Overview)

Charts

JavaFX includes built-in chart classes:

- LineChart, BarChart, PieChart, AreaChart, etc.

Example:

```
PieChart pieChart = new PieChart();
pieChart.getData().add(new PieChart.Data("Java", 50));
pieChart.getData().add(new PieChart.Data("Python", 30));
pieChart.getData().add(new PieChart.Data("C++", 20));
```

Media

Supports **audio and video** playback using Media and MediaPlayer classes.

Summary

Concept	Description
JavaFX	Modern GUI toolkit
Scene Graph	Tree structure for UI
Stage	Top-level window
Scene	Container for nodes
Layout Panes	Arranges controls
UI Controls	Buttons, Labels, etc.
Event Handling	Respond to user actions
Property Binding	Sync data automatically
FXML	Declarative UI
CSS	Styling for UI