Chapter 12: JavaFX and GUI Programming

Introduction

Graphical User Interfaces (GUIs) are a vital part of modern desktop and enterprise applications. While Swing was Java's original toolkit for GUI development, **JavaFX** has emerged as the modern alternative, offering a rich set of controls, CSS styling, FXML support, and hardware-accelerated graphics.

This chapter introduces **JavaFX**, explains how to design user-friendly interfaces, and covers core components like **layouts, controls, events, media integration, FXML**, and **CSS styling**. You'll learn how to build and deploy GUI-based applications with a responsive design and a clean separation of concerns using MVC (Model-View-Controller) patterns.

12.1 What is JavaFX?

JavaFX is a platform for creating and delivering desktop applications, as well as rich internet applications (RIAs). It replaces Swing and AWT with a modern, feature-rich GUI toolkit.

Features of JavaFX

- Hardware-accelerated graphics
- Scene graph-based rendering
- Built-in controls (buttons, sliders, etc.)
- CSS-like styling support
- FXML for declarative UI design
- Media and Web integration
- MVC support

12.2 JavaFX Architecture

JavaFX is based on a **Scene Graph** architecture.

🔷 Key Components:

- **Stage**: The top-level container (a window).
- **Scene**: Holds all UI elements.
- **Nodes**: Elements in the scene graph (buttons, panes, etc.).

```
nginxCopy codeStage → Scene → Parent Node → Child Nodes
```

🧩 Node Types:

- Controls (e.g., Button, TextField)
- Layouts (e.g., VBox, HBox, GridPane)
- Shapes (e.g., Circle, Rectangle)
- Containers (e.g., AnchorPane, BorderPane)

📓 12.3 Setting Up JavaFX

To start using JavaFX:

🔧 Installation:

- Use **JDK 11+** with OpenJFX SDK
- Set environment variables
- Use **IDE support** (e.g., IntelliJ, Eclipse with e(fx)clipse plugin)

✅ Hello World Example

```
javaCopy codeimport 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.setScene(scene);
        primaryStage.setTitle("JavaFX Demo");
        primaryStage.show();
    }
```

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

📝 12.4 JavaFX UI Controls

JavaFX provides a wide range of UI components:

Control	Description
`Button`	Triggers an action
`Label`	Displays text
`TextField`	Single-line input
`TextArea`	Multi-line input
`CheckBox`	Boolean toggle option
`RadioButton`	Mutually exclusive option
`ComboBox`	Dropdown list
`Slider`	Graphical value slider
`ListView`	Scrollable list
`TableView`	Displays tabular data

🔘 Example: Button and Event Handling

```
javaCopy codeButton btn = new Button("Click Me");
btn.setOnAction(e -> System.out.println("Button clicked!"));
```

📝 12.5 Layouts in JavaFX

Layouts help organize components within the scene.

🔲 Common Layout Managers:

Layout	Description
`HBox`	Horizontal row
`VBox`	Vertical column
`BorderPane`	Top, bottom, center, left, right layout

Layout	Description
`GridPane`	Grid-style layout
`StackPane`	Stacks components on top of each other
`AnchorPane`	Absolute positioning

📦 VBox Example:

```
javaCopy codeVBox root = new VBox(10); // 10 px spacing
root.getChildren().addAll(new Label("Name"), new TextField());
```

📝 12.6 Event Handling in JavaFX

JavaFX uses **event-driven programming**. Events are generated by UI interactions.

🔄 Event Types:

- **ActionEvent** (e.g., button clicks)
- **KeyEvent** (keyboard input)
- **MouseEvent** (mouse actions)

🛠️ Example:

```
javaCopy codebtn.setOnAction(event -> {
    System.out.println("Handled in lambda!");
});
```

You can also use named handler classes implementing `EventHandler<T>`.

📝 12.7 JavaFX CSS Styling

JavaFX allows UI styling using CSS, similar to web development.

🎨 Example CSS:

```
cssCopy code.button {
    -fx-font-size: 16px;
    -fx-background-color: #3498db;
}
```

📎 Applying CSS:

```
javaCopy codescene.getStylesheets().add("style.css");
```

12.8 FXML: Declarative UI

FXML is an XML-based language for designing UI separate from logic.

Sample FXML:

```
xmlCopy code<?xml version="1.0" encoding="UTF-8"?>
<VBox xmlns="http://javafx.com/javafx" xmlns:fx="http://javafx.com/fxml"
      fx:controller="myapp.MyController">
    <Label text="Enter Name"/>
    <TextField fx:id="nameField"/>
    <Button text="Submit" onAction="#handleSubmit"/>
</VBox>
```

Controller:

```
javaCopy codepublic class MyController {
    @FXML private TextField nameField;

    @FXML
    private void handleSubmit() {
        System.out.println("Hello, " + nameField.getText());
    }
}
```

Loading FXML:

```
javaCopy codeParent root = FXMLLoader.load(getClass().getResource("layout.fxm
l"));
Scene scene = new Scene(root);
```

12.9 JavaFX Media Integration

JavaFX supports **audio and video playback**.

Playing Audio:

```
javaCopy codeMedia sound = new Media("file:///path/to/audio.mp3");
MediaPlayer mediaPlayer = new MediaPlayer(sound);
mediaPlayer.play();
```

Playing Video:

```
javaCopy codeMedia media = new Media("file:///path/to/video.mp4");
MediaPlayer player = new MediaPlayer(media);
MediaView mediaView = new MediaView(player);
```

📓 12.10 Advanced GUI: Charts, Effects, and Animation

📊 Charts:

- **LineChart**
- **BarChart**
- **PieChart**

💡 Effects:

- DropShadow
- Glow
- Reflection

🎞️ Animation:

Use `Timeline` or `TranslateTransition` for animation.

```
javaCopy codeTranslateTransition transition = new TranslateTransition(Duratio
n.seconds(2), button);
transition.setToX(100);
transition.play();
```

📓 12.11 MVC Pattern in JavaFX

JavaFX encourages **Model-View-Controller (MVC)** architecture:

- **Model** – Data layer
- **View** – FXML/CSS UI
- **Controller** – Handles logic/events

This separation makes applications modular and testable.

📓 12.12 Deployment of JavaFX Applications

JavaFX apps can be deployed as:

- **Executable JARs**

- **Native installers (.exe, .dmg)**
- **Java modules (JMODs)**

Use **jlink** and **jpackage** tools for packaging in JDK 14+.

Summary

In this chapter, you explored **JavaFX** — Java's modern GUI toolkit — as a powerful way to build interactive and professional user interfaces. You learned how to work with **UI controls**, **layouts**, **event handling**, **FXML**, **CSS styling**, and **media integration**. Using MVC principles, JavaFX promotes a clean separation between logic and interface. As desktop development remains relevant in enterprise and educational applications, JavaFX is an essential tool for any Java developer.
