Chapter 16: GUI Programming (e.g., using AWT/Swing or JavaFX)

Introduction

Graphical User Interface (GUI) programming enables interaction between users and software through graphical elements like buttons, text fields, menus, and windows instead of textual input via command-line interfaces. In Java, GUI programming has evolved significantly — from Abstract Window Toolkit (AWT) to Swing and, more recently, JavaFX.

AWT was Java's original platform-dependent GUI toolkit. Swing improved upon AWT by being more flexible and offering a rich set of components, all while being platform-independent. JavaFX, introduced later, brings modern GUI development capabilities like CSS styling, FXML, animations, and media playback.

This chapter delves deep into the architecture, components, event handling, and GUI design using AWT, Swing, and JavaFX — empowering students to build real-world desktop applications.

16.1 Basics of GUI Programming

16.1.1 What is a GUI?

- GUI (Graphical User Interface) is a visual interface for interacting with software.
- Consists of elements like windows, menus, toolbars, buttons, icons, and labels.

16.1.2 GUI vs CLI

Aspect	GUI	CLI
Usability	User-friendly	Requires commands
Accessibility	High for non-tech users	Low
Learning Curve	Easy	Steep
Performance	Resource-intensive	Lightweight

16.2 AWT (Abstract Window Toolkit)

16.2.1 Overview

- Part of Java's original GUI toolkit (`java.awt`).
- Uses native platform components (heavyweight).

16.2.2 AWT Hierarchy

```
Object
 └── Component
      └── Container
           ├── Window
           │    ├── Frame
           │    └── Dialog
           └── Panel
```

16.2.3 Common AWT Components

Component	Class
Button	`java.awt.Button`
Label	`java.awt.Label`
TextField	`java.awt.TextField`
TextArea	`java.awt.TextArea`
Checkbox	`java.awt.Checkbox`
Choice (dropdown)	`java.awt.Choice`
List	`java.awt.List`

16.2.4 AWT Layout Managers

- **FlowLayout**
- **BorderLayout**
- **GridLayout**
- **CardLayout**

16.2.5 Event Handling in AWT

- Based on **Delegation Event Model**.
- Components generate events.
- Listeners handle them.

```
Button b = new Button("Click");
b.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        System.out.println("Button clicked!");
    }
});
```

16.3 Swing

16.3.1 Overview

- Part of `javax.swing` package.
- Lightweight and platform-independent.

- Uses MVC (Model-View-Controller) architecture.

16.3.2 Swing vs AWT

Feature	Swing	AWT
Component Type	Lightweight	Heavyweight
Look & Feel	Pluggable	Native
Extensibility	High	Limited

16.3.3 Common Swing Components

Component	Class
Button	`JButton`
Label	`JLabel`
Text field	`JTextField`
Password field	`JPasswordField`
Text area	`JTextArea`
Check box	`JCheckBox`
Radio button	`JRadioButton`
Combo box	`JComboBox`
Menu bar	`JMenuBar`, `JMenu`, `JMenuItem`
Table	`JTable`
Tree	`JTree`

16.3.4 Creating a Simple Swing App

```
import javax.swing.*;

public class MySwingApp {
    public static void main(String[] args) {
        JFrame frame = new JFrame("My Swing App");
        JButton button = new JButton("Click Me");

        button.addActionListener(e -> JOptionPane.showMessageDialog(null,
"Hello!"));

        frame.add(button);
        frame.setSize(300, 200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

16.3.5 Layout Managers in Swing

- Same as AWT (Swing extends AWT containers).

- Additional: **GroupLayout**, **SpringLayout**, etc.

16.4 Event Handling in GUI

16.4.1 Event Sources and Listeners

- **Source**: Component that generates an event (e.g., Button).
- **Listener**: Interface that processes the event (e.g., `ActionListener`, `MouseListener`).

16.4.2 Common Event Listener Interfaces

Event	Interface
Button click	`ActionListener`
Mouse events	`MouseListener`, `MouseMotionListener`
Key press	`KeyListener`
Window open/close	`WindowListener`

16.4.3 Lambda in Event Handling (Java 8+)

```
button.addActionListener(e -> System.out.println("Clicked!"));
```

16.5 JavaFX

16.5.1 Overview

- Modern UI toolkit introduced in Java 8.
- Supports 2D/3D graphics, CSS, FXML (markup), and multimedia.
- Replaces Swing for complex UIs.

16.5.2 JavaFX Architecture

- **Stage**: Main window.
- **Scene**: Content inside the window.
- **Nodes**: UI components (buttons, labels, etc.).

16.5.3 JavaFX Application Structure

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

public class JavaFXApp extends Application {
    public void start(Stage primaryStage) {
        Button btn = new Button("Say Hello");
        btn.setOnAction(e -> System.out.println("Hello from JavaFX"));
```

```
        StackPane root = new StackPane(btn);
        Scene scene = new Scene(root, 300, 200);

        primaryStage.setTitle("JavaFX App");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

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

16.5.4 FXML and Scene Builder

- Declarative way to define UI in XML.
- **Scene Builder**: Drag-and-drop GUI tool.

```
<Button text="Click Me" onAction="#handleClick"/>
```

16.5.5 CSS Styling in JavaFX

```
.button {
    -fx-background-color: #00ff00;
    -fx-font-size: 14pt;
}
```

16.6 GUI Design Principles

- **Consistency**: UI components should behave similarly.
- **Feedback**: Visual/auditory cues for user actions.
- **Simplicity**: Avoid clutter; focus on key tasks.
- **Accessibility**: Ensure usability for all users.

16.7 Comparison of GUI Frameworks

Feature	AWT	Swing	JavaFX
Platform-dependency	Yes	No	No
Look & Feel	Native	Pluggable	Modern (CSS-based)
Multimedia support	Limited	Limited	Rich (audio/video/3D)
Declarative UI	No	No	Yes (FXML)
CSS Support	No	Limited	Full

Summary

In this chapter, we explored GUI programming in Java using AWT, Swing, and JavaFX. Starting from basic GUI principles and components to event-driven programming models and UI styling, the chapter laid a strong foundation for building rich client-side desktop applications. While AWT and Swing remain relevant for legacy systems, JavaFX provides a robust platform for modern GUI applications with superior styling, media, and 3D support.

For real-world application development, understanding these tools and frameworks empowers developers to deliver responsive, interactive, and visually appealing software solutions.
