Welcome

Presentation

CSE-2101

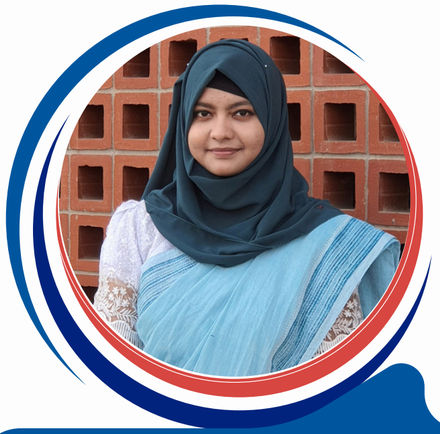
Object Oriented Design and Design Patterns

By

Abdullah Atif
241311051
34th { B }

MD. SAIBUR RAHMAN
241311053
34th { B }

NISHAT TABASSUM PROMI
241311059
34th { B }



MST. FARHANA JAHAN
241311060
34th { B }

Today's Topic

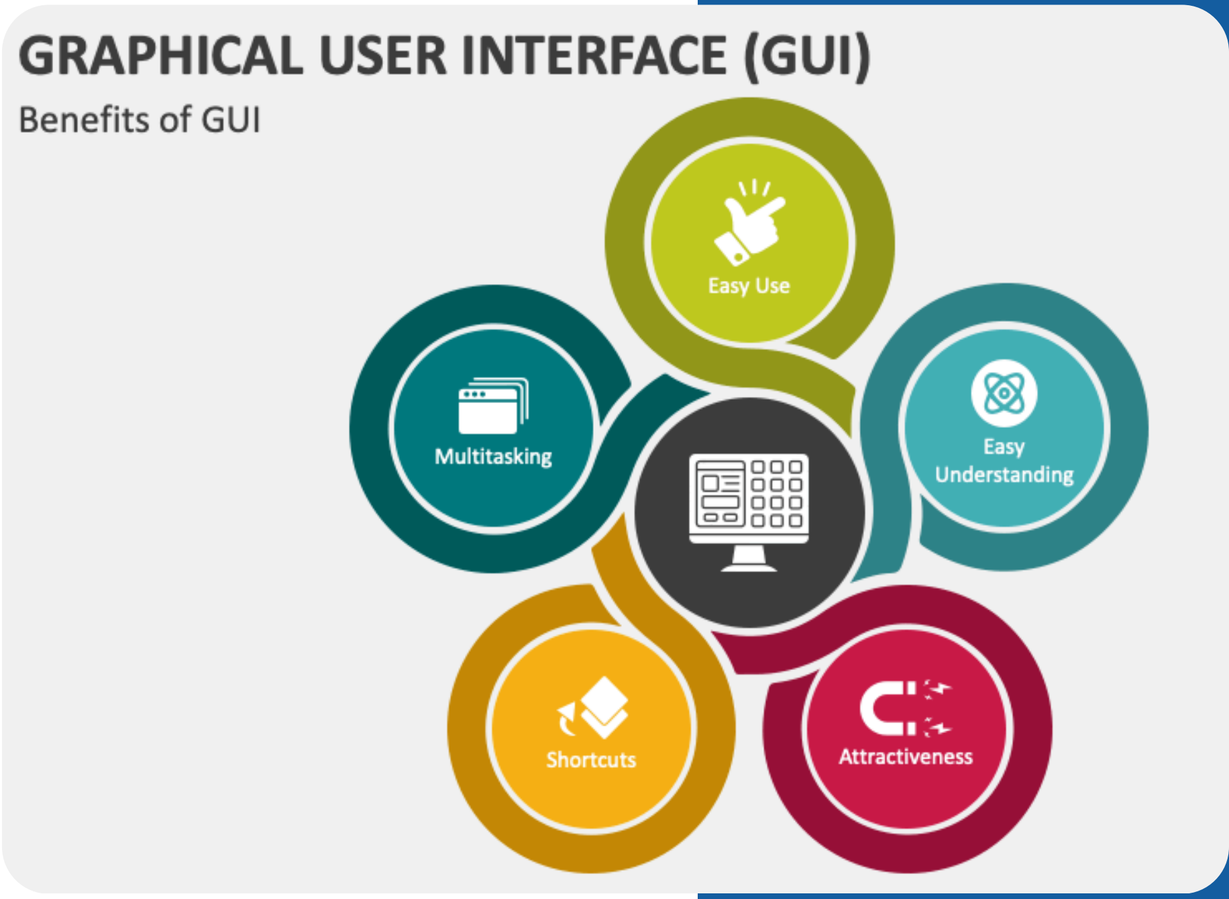
GUI

A basic login and registration form with a nice user interface that includes buttons and windows.

What can you see in the picture?

What is a GUI?

- A GUI (pronounced “GOO-ee”) stands for **Graphical User Interface** gives an application a distinctive “look and feel.”
- A type of UI that uses graphical elements like windows, icons, and buttons for interaction instead of text-based commands.
- A graphical user interface is a visual interface to a program. GUIs are built from GUI components (buttons ,menus, labels etc). A GUI component is an object with which the user interacts via the mouse or keyboard.



Then, what is a Java GUI?

Java GUI

Java GUI is a set of graphical components provided by Java (via libraries like AWT and Swing) that enable developers to create interactive, user-friendly applications with windows, buttons, menus, and other visual elements. The classes that are used to create GUI components are part of the java.awt or javax.swing package. Both these packages provide rich set of user interface components.

Java GUI Toolkits

Java GUI toolkits are libraries and frameworks that provide developers with the tools to create graphical user interfaces (GUIs) for Java applications. These toolkits offer pre-built components such as buttons, text fields, and windows, allowing for the design of interactive and visually appealing applications.

Popular Java GUI toolkits...

- AWT
- Swing
- JavaFX

AWT

01

AWT stands for Abstract Window Toolkit contains original but not pure GUI components that came with the first release of JDK.

02

Introduced in Java 1.0, AWT was the first GUI framework in Java, allowing platform-independent GUI development using native OS components.

03

AWT is Java's original GUI toolkit that provides basic components like buttons, labels, and text fields for building graphical user interfaces.

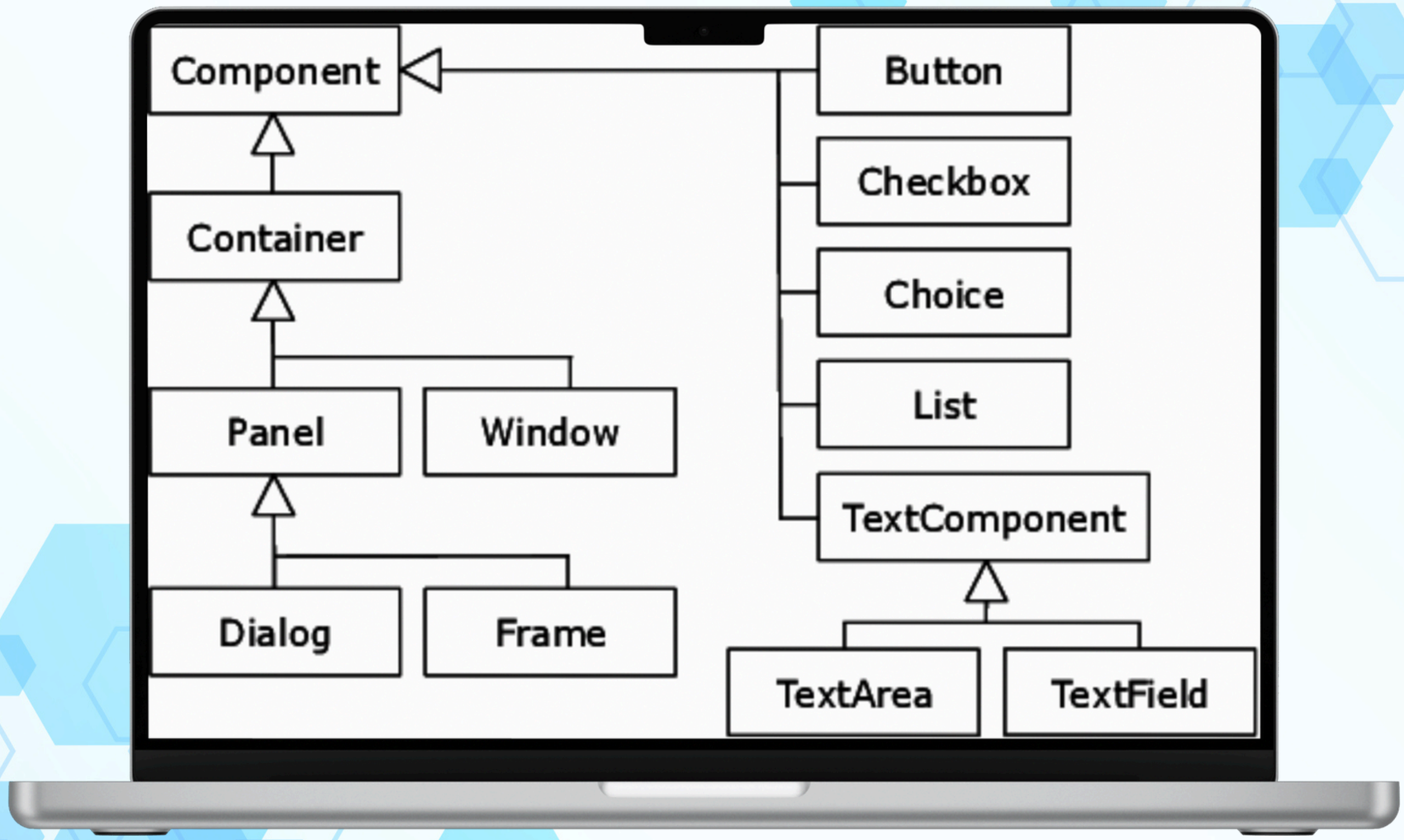
04

AWT components are often called Heavy Weight Components (HWC) as they rely on the local platform‘s windowing system.

05

AWT component it creates a corresponding process on the operating system.

AWT Components



AWT Components

Basic UI Components

Label

Displays a single line of read-only text.

Button

Triggers an action when clicked.

TextField

Allows the user to enter a single line of text.

TextArea

Allows multi-line text input or display.

Choice and Selection Components

Checkbox

Allows multiple selections (true/false).

Choice

A drop-down list for selecting one item.

List

Displays a scrollable list of items; supports single or multiple selections.

Scrollbar

Used to scroll through content vertically or horizontally.

Swing

01 *Swing is a GUI toolkit in Java that provides a richer set of components than AWT and supports a pluggable look-and-feel.*

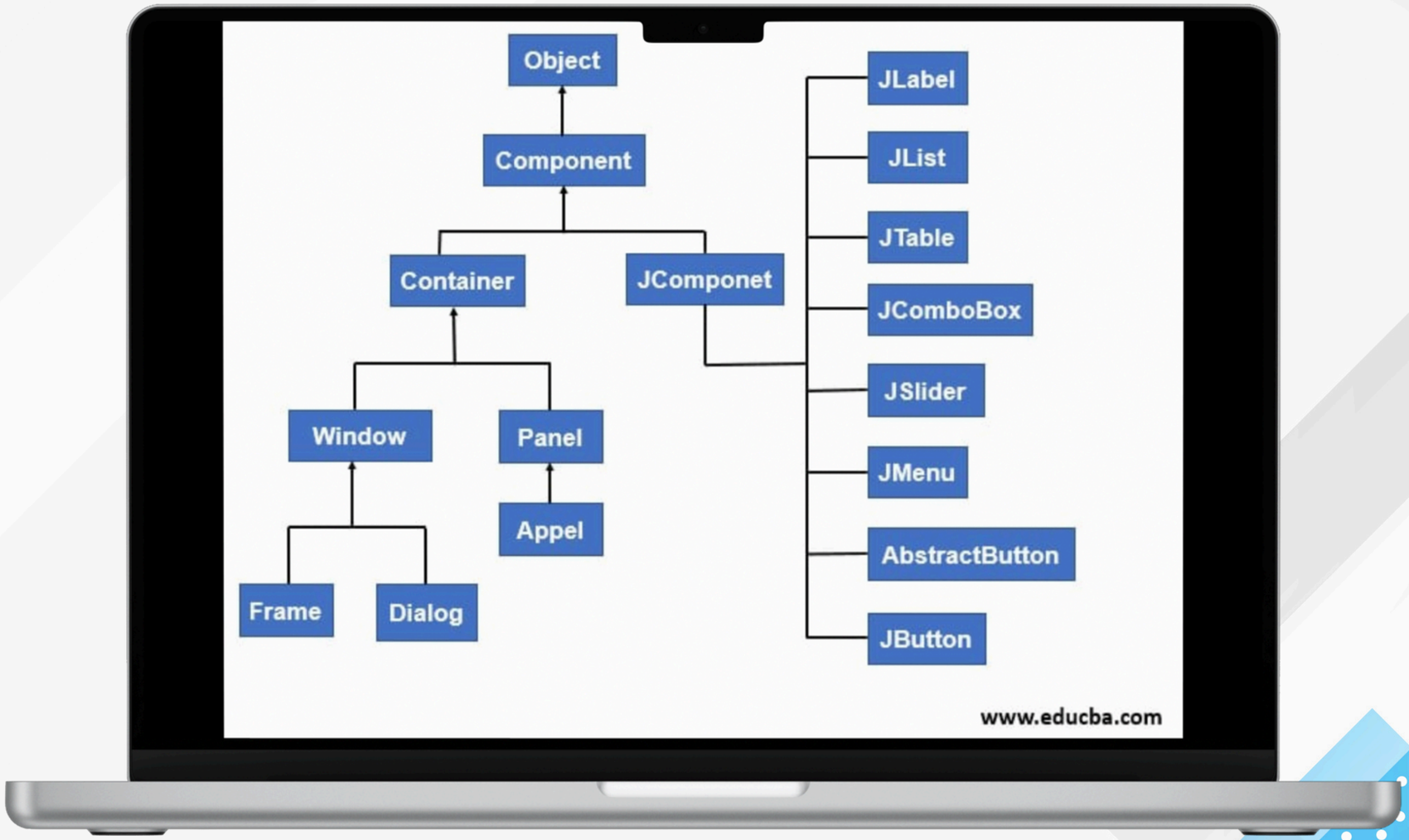
02 *It is part of Javax.swing package and was introduced as a part of Java Foundation Classes (JFC).*

03 *Swing is lightweight, meaning it doesn’t rely on the operating system's native GUI.*

04 *It supports pluggable look-and-feel, allowing the UI to look consistent across platforms or mimic native styles.*

05 *Swing follows the MVC (Model-View-Controller) architecture, separating data, UI, and event handling.*

Swing Components



Swing Components

Basic Components

- **JLabel** — Displays text/images
- **JButton** — A clickable button
- **JTextField** — Single-line input
- **JTextArea** — Multi-line input

Selection & Advanced Components

- **JCheckBox** — Multiple options (checkable)
- **JRadioButton** — Single selection from a group
- **JComboBox** — Drop-down menu for one selection
- **JTable** — Displays data in tabular form
- **JTree** — Shows hierarchical data (like folders)
- **JTabbedPane** — Multiple tabs in one window
- **JScrollPane** — Adds scrollbars to large components
- **JTable** — Displays data in tabular form

Layout Manager

Definition

A Layout Manager in Java is an object that controls the position and size of components within a container.

🎯 Purpose of Layout Manager

To automatically arrange GUI components in a structured and responsive way, so developers don’t have to manually set coordinates.

Common Layout Managers (used in both AWT and Swing)

Layout Manager	Description
FlowLayout	Arranges components left to right, wrapping to next line
BorderLayout	Divides the container into 5 regions: North, South, East, West, Center
GridLayout	Places components in a grid with equal-sized cells
GridBagLayout	Flexible grid with varying cell sizes and alignments
CardLayout	Manages multiple components (like cards), showing one at a time
BoxLayout	Aligns components either vertically or horizontally (more common in Swing)

📝 Note:

- Swing offers more flexibility and better customization with the same layout managers.
- BoxLayout is often preferred in Swing for advanced UI design.

Why we prefer Swing over AWT

Feature	AWT (Heavyweight)	Swing (Lightweight)	Explanation
Platform Dependence	Uses native OS components (e.g., Windows buttons, Mac menus)	Uses pure Java components	AWT relies on the system's windowing toolkit, making it platform-dependent. Swing creates UI elements in Java itself, so it's platform-independent.
Look & Feel	OS-dependent — looks like the host system	Pluggable and customizable — can mimic any OS or use custom themes	AWT components always follow the OS style. Swing allows developers to change the look and feel using built-in or custom styles.
Component Set	Limited — basic components like Button, Label	Extensive — includes advanced components like JTable, JTree, JTabbedPane, etc.	Swing offers a richer set of components, suitable for complex UIs. AWT supports only basic UI elements.
Performance	Generally faster, but less flexible	Slightly slower, but more powerful and flexible	AWT can be faster due to using native code, but lacks customizability. Swing is a bit heavier but supports more features and better UI control.

Event Handling in AWT & Swing

Definition

Event handling in Java refers to the process of responding to user interactions (like button clicks, mouse movements, key presses) by using a mechanism called the event delegation model, where events are generated by components and handled by listener interfaces.

🎯 Purpose of Event Handling

- **Respond to User Actions**

To perform specific actions when a user interacts with GUI elements (e.g., clicking a button).

- **Separate Logic from UI**

Helps keep the business logic separate from the user interface code using listener interfaces.

- **Control Application Behavior**

Allows the program to behave dynamically based on different user inputs or system events.

Building a Simple GUI Application

Swing.java

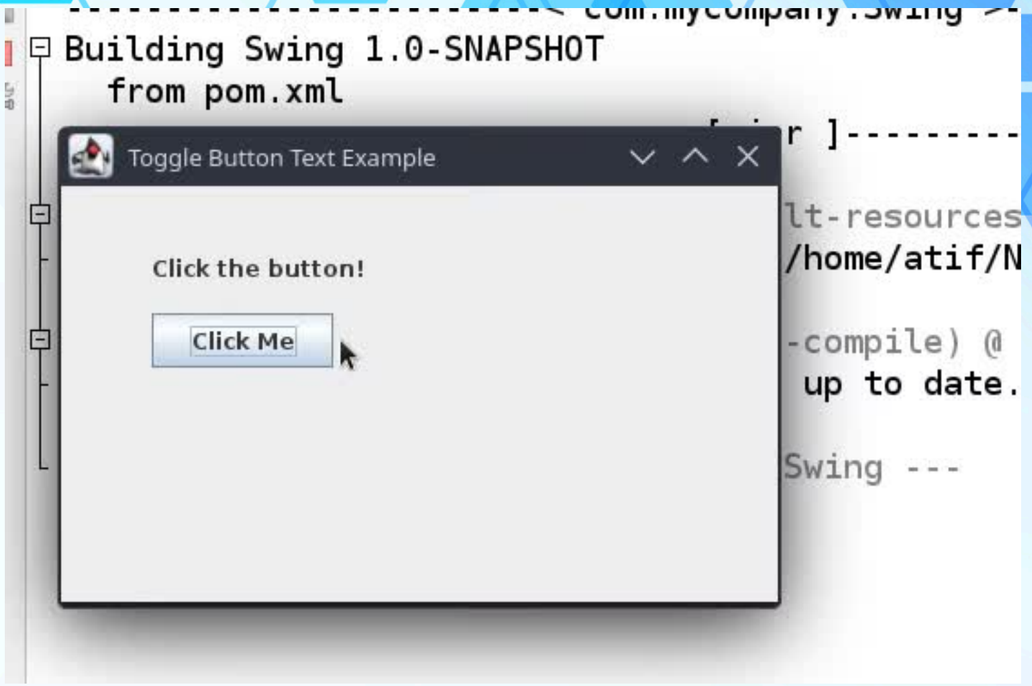
```java
package in.swing;
import javax.swing.*;

public class Swing extends JFrame {
  private boolean clicked = false; // Track state
  public Swing() {
      setTitle("Toggle Button Text Example");
      setLayout(null);

      // Create label and button
      JLabel label = new JLabel("Click the button!");
      label.setBounds(50, 30, 150, 30);
      JButton button = new JButton("Click Me");
      button.setBounds(50, 70, 100, 30);

      // Add action listener with toggle logic
      button.addActionListener(e -> {
          if (clicked) {
              label.setText("Click the button!");
          } else {
              label.setText("Button Clicked!");
          }
          clicked = !clicked; // Toggle the state
      });
      // Add components
      add(label);
      add(button);

      // Frame settings
      setSize(300, 200);
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      setVisible(true);
  }
  public static void main(String[] args) {
      Swing swing = new Swing(); // Create and show the GUI
  }
}
```



Ouput

Summary and Conclusion

✅ Key Takeaways

- AWT: Java's original GUI toolkit; uses heavyweight components tied to OS.
- Swing: Improved GUI toolkit; provides lightweight, platform-independent components.
- Both use Event Delegation Model for handling events.

Thank You!