CS205 Object Oriented Programming in Java

Module 5 - **Graphical User Interface and Database support of Java**

(Part 2)

Prepared by

Renetha J.B.

AP

Dept.of CSE,

Lourdes Matha College of Science and Technology

Topics

- ☑ **Swings**
  - ☑ MVC
  - ☑ Swing Controls
  - ☑ Components and Containers

MVC

- A visual component is a composite of three distinct aspects:
  - The **state information** associated with the component **MODEL**
  - The way that the **component looks** when rendered on the screen **VIEW**
  - The way that the **component reacts** to **CONTROLLER**

- MVC (Model View Controller) architecture is successful because **each piece of the design corresponds to an aspect of a component.**

MVC(contd.)

- In **MVC** terminology,
  - the ***Model*** corresponds to the **state information associated with the component**.
    - For example, in the case of a check box, the **model** contains a field that indicates if the box is checked or unchecked.
  - The ***View*** determines **how the component is displayed on the screen**, including any aspects of the view that are affected by the current state of the model.
  - The ***Controller*** determines **how the component reacts to the user**.
    - For example, when the user clicks a check box, the controller reacts by changing the model to reflect the user's choice (checked or unchecked). So view changes.

MVC(contd.)

- By separating a component into a model, a view, and a controller, the *specific implementation of each can be changed* **without affecting the other two**.
- Swing uses a **modified version of MVC** that **combines the view and the controller into a single logical entity** called the ***UI delegate***.
  - So, Swing's approach is called either the Model-Delegate architecture or the Separable Model architecture.
- Swing's pluggable look and feel is made possible by its Model-Delegate architecture.
- Because the view (look) and controller (feel) are ***separate*** from the model, the look and feel can be changed without affecting how the component is used within a program.

MVC(contd.)

- To support the **Model-Delegate** architecture, most Swing components contain two objects.
  - The first represents the **model**.
    - Models are defined by interfaces
  - The second represents the **UI delegate**.
    - UI delegates are classes that inherit **ComponentUI.**

Swing Controls

- Swing controls plays major role in swing applications, every application will have some components and their corresponding event listener.
- Swing controls will inherit the properties from following classes.
  - Component
  - Container
  - JComponent.

Swing Controls

- Some swing controls are
  - Container Control(Parent control)
    - It hold other controls(child controls or components)
      - JFrame, JPanel
  - Child Control –These controls are inside container control
    - They can only exist inside container control. They can be components or containers.
      - JTextArea,JLabel, JButton
- When container control is deleted then its child controls are also deleted.

Components and Containers

- A Swing GUI consists of two key items: ***components*** and ***containers***
- A container holds a group of components.
  - So a **container** is a **special type of component** that is designed to hold other components.
- To display a component, it must be held within a container.
- So all Swing GUIs will have at least one container.
- Because containers are components, a *container can also hold other containers*.

Components

- Swing components are derived from the **JComponent class.**
- **JComponent** provides the ***functionality*** that is common to all components.
  - E.g **JComponent** supports the pluggable look and feel.
- **JComponent** inherits the AWT classes **Container** and **Component**..
  - So, a Swing component is **built on and compatible** with an AWT component.

Components(contd.)

- All of Swing's **components** are represented by **classes** defined within the package **javax.swing.**
- The class names for Swing components are

JApplet	JButton	JCheckBox	JCheckBoxMenuItem
JColorChooser	JComboBox	JComponent	JDesktopPane
JDialog	JEditorPane	JFileChooser	JFormattedTextField
JFrame	JInternalFrame	JLabel	JLayeredPane
JList	JMenu	JMenuBar	JMenuItem
JOptionPane	JPanel	JPasswordField	JPopupMenu
JProgressBar	JRadioButton	JRadioButtonMenuItem	JRootPane
JScrollBar	JScrollPane	JSeparator	JSlider
JSpinner	JSplitPane	JTabbedPane	JTable
JTextArea	JTextField	JTextPane	JTogglebutton
JToolBar	JToolTip	JTree	JViewport
JWindow			

** All component classes begin with the letter **J.***

Containers

- Swing defines **two types** of containers.
  - The first are **top-level containers(heavyweight)**
  - The second type of containers supported by Swing are **lightweight containers**.

Containers(contd.)

- The first are **top-level containers**
  - **JFrame, JApplet, JWindow, and JDialog.**
  - These containers **do not** inherit **JComponent**
  - They inherit the AWT classes **Component and Container**
  - The top-level containers are **heavyweight.**
  - A top-level container must be at the top of a containment hierarchy
  - A top-level container is not contained within any other container.
  - Every containment hierarchy must begin with a top-level container.
  - The one most commonly used for applications is **JFrame.**
  - The one used for applets **is JApplet.**

Containers (contd.)

- The second type of containers supported by Swing are **lightweight containers**.
  - Lightweight containers *do* ***inherit JComponent.***
    - E.g. **JPanel** is a general-purpose container.
  - Lightweight containers are often used to organize and manage groups of related components because a lightweight container can be contained within another container.
  - We can use lightweight containers such as **JPanel** to create subgroups of related controls that are contained within an outer container.

Top-level containers

- Top level containers **do not** inherit **Jcomponent.**
- **Heavyweight**
- A top-level container is not contained within any other container.

- E.g. **JFrame, JApplet, JWindow, and JDialog**

Lightweight containers.

- Lightweight containers *do inherit **JComponent.***
- **Lightweight**
- A lightweight container can be contained within another container.

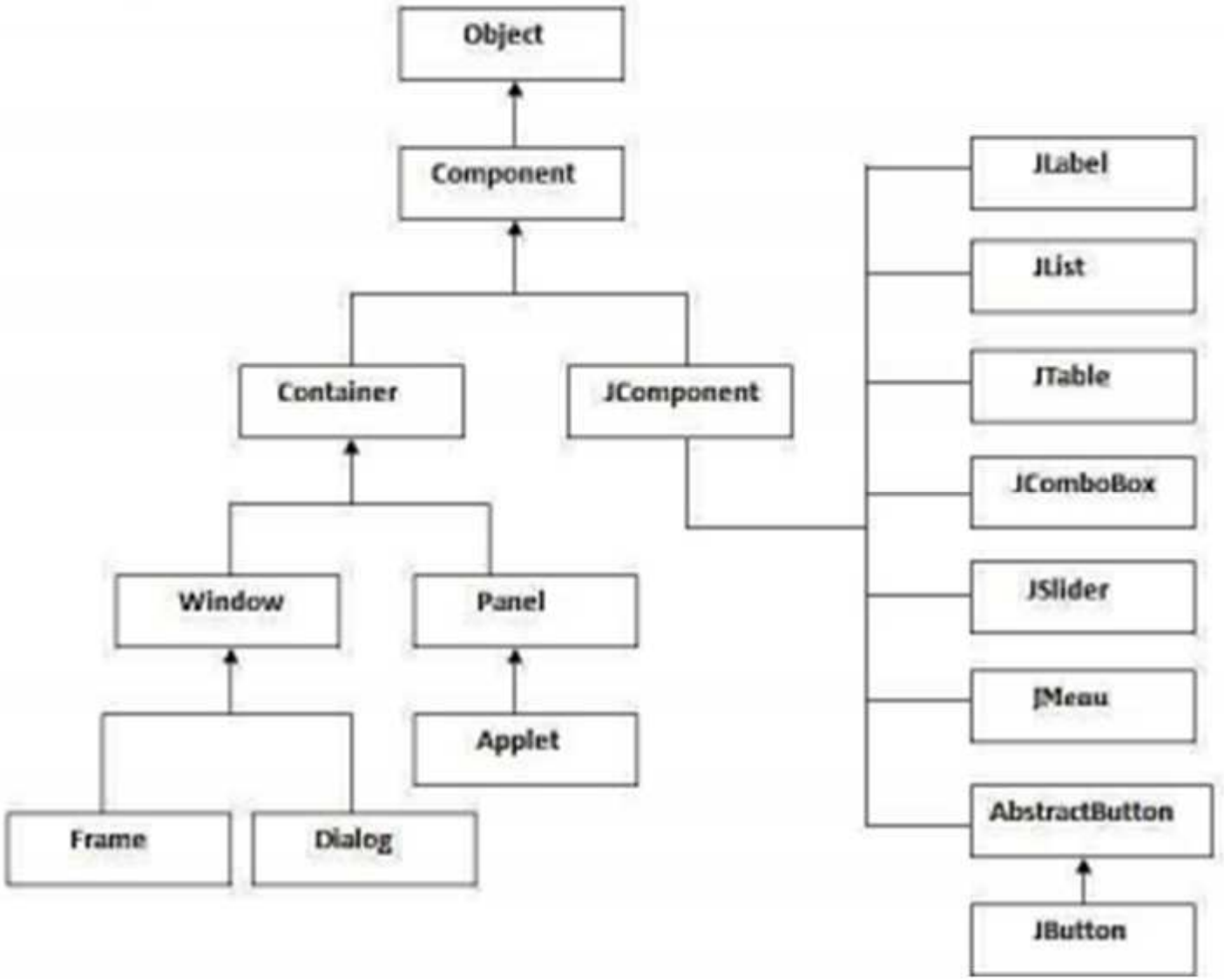
- E.g. **JPanel**

The Top-Level Container Panes

- The **Top-Level Container Panes**
  - Each top-level container defines a set of panes.
  - At the **top** of the hierarchy is an instance of **JRootPane**.
    - JRootPane is a lightweight container whose purpose is to manage the other panes.
    - It also helps manage the optional menu bar.
    - The panes that comprise the root pane are called
      - the **glass pane,**
      - **the content pane,**
      - **the layered pane**.

The Top-Level Container Panes

- Glass pane :
  - The glass pane is the **top-level pane.**
  - It sits above and completely covers all other panes.
  - By default, it is a **transparent** instance of JPanel.
- Layered pane :
  - The layered pane is an instance of JLayeredPane.
  - The layered pane allows components to be **given a depth value.**
  - This value determines which component overlays another.
- Content pane:
  - The pane with which your application will **interact the most** is the content pane
  - When we add a component, such as a button, to a top- level container, **we will add it to the content pane**.
  - By default, the content pane is an **opaque** instance of JPanel.


Object
Component
Container
JComponent
Window
Panel
Applet
Frame
Dialog
JLabel
JList
JTable
JComboBox
JSlider
JMenu
AbstractButton
JButton

Reference

- Herbert Schildt, Java: **The Complete Reference, 8/e, Tata McGraw Hill, 2011**.