



Royal University of Bhutan



Unit VII

Working with Graphics User Interface (Swing Components)

Tutor: Pema Galey

Learning Outcomes

In this session, you will learn about:

- Introduction to Swing
- Exploring Swing
- Introducing Swing Menus

Swing

- Swing did not exist in the early days of Java.
- In AWT, it translates its various visual components into their corresponding, platform-specific equivalents, or *peers*.
- The look and feel of an AWT component is defined by the platform, not by Java.
- The AWT components use native code resources, they are referred to as ***heavyweight***.
- Swing *does not* replace AWT. Instead, Swing is built on the foundation of the AWT. The basic understanding of the AWT and of event handling is required to use Swing.

Swing

- Swing was created to address the limitations present in the AWT. It does this through two key features: **lightweight components and a pluggable look and feel.**
- Swing components are *lightweight*. It has written entirely in Java and do not map directly to platform-specific peers. Thus, lightweight components are more efficient and more flexible.
- Swing supports a *pluggable look and feel* (PLAF). Because each Swing component is rendered by Java code rather than by native peers.

Components and Containers of Swing

Components :

- Swing components are derived from the **JComponent** class.
- **JComponent** provides the functionality that is common to all components. **JComponent** inherits the AWT classes **Container** and **Component**.
- Thus, a Swing component is built on and compatible with an AWT component.
- All of Swing's components are represented by classes defined within the package **javax.swing**.

Swing Components

JApplet (Deprecated)	JButton	JCheckBox	JCheckBoxMenuItem
JColorChooser	JComboBox	JComponent	JDesktopPane
JDialog	JEditorPane	JFileChooser	JFormattedTextField
JFrame	JInternalFrame	JLabel	JLayer
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		

The Swing Containers

Swing defines two types of containers.:

1. The first are Top-Level containers:

- **JFrame**, **JWindow**, and **JDialog**. These containers do not inherit **JComponent**. They do, however, inherit the AWT classes **Component** and **Container**.
- Unlike Swing's other components, the top-level containers are heavyweight.
- A top-level container is not contained within any other container.

The Swing Containers

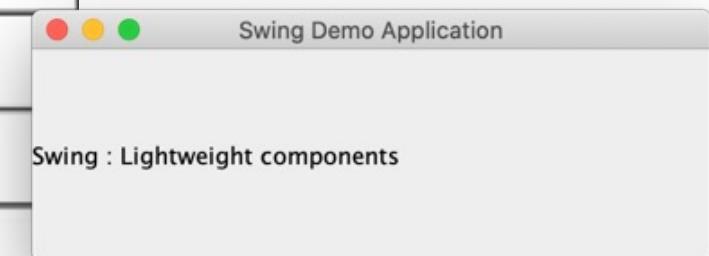
Swing defines two types of containers.:

2. The second are Lightweight containers:

- Lightweight containers *do* inherit **JComponent**. Like **JPanel**, which 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.
- **JPanel** to create subgroups of related controls that are contained within an outer container.

Swing Packages

javax.swing	javax.swing.plaf.basic	javax.swing.text
javax.swing.border	javax.swing.plaf.metal	javax.swing.text.html
javax.swing.colorchooser	javax.swing.plaf.multi	javax.swing.text.html.parser
javax.swing.event	javax.swing.plaf.nimbus	javax.swing.text.rtf
javax.swing.filechooser	javax.swing.plaf.synth	javax.swing.tree
javax.swing.plaf	javax.swing.table	javax.swing.undo



- The main package is **javax.swing**
- The **add()** method is inherited by **JFrame** from the AWT class **Container**. The content pane can be directly invoke which it was obtained by calling **getContentPane()** on a **JFrame** instance.

Swing Components

- The **Swing components** provide rich functionality and allow a high level of customization.

JButton	JList
JTable	JCheckBox
JRadioButton	JTextField
JComboBox	JScrollPane
JToggleButton	JLabel
JTabbedPane	JTree
JPanel	JTextArea
JTable	

Swing Component

JLabel :

- **JLabel** is Swing's easiest-to-use component.
- **JLabel** can be used to display text and/or an icon.
- **JLabel** defines several constructors:

`JLabel(Icon icon)`

`JLabel(String str)`

`JLabel(String str, Icon icon, int align)`

Swing Component

JLabel :

- **ImageIcon** can be passed as an argument to the **Icon** parameter of **JLabel**'s constructor.
- The icon and text associated with a label can be set by these methods:

`void setIcon(Icon icon)`

`void setText(String str)`

Swing Component

JTextField :

- **JTextField** is the simplest Swing text component.
- **JTextField** allows you to edit one line of text.
- The constructors are :

JTextField(int cols)

JTextField(String str, int cols)

JTextField(String str)

Swing Component

JButton:

- Swing defines four types of buttons: **JButton**, **JToggleButton**, **JCheckBox**, and **JRadioButton**.
- All are subclasses of the **AbstractButton** class, which extends **JComponent**.
- The following methods sets icons:

```
void setDisabledIcon(Icon di)
void setPressedIcon(Icon pi)
void setSelectedIcon(Icon si)
void setRolloverIcon(Icon ri)
```

Swing Component

JButton:

- The **JButton** class provides the functionality of a push button.
- **JButton** allows an icon, a string, or both to be associated with the push button.
- Three of its constructors are shown here:

`JButton(Icon icon)`

`JButton(String str)`

`JButton(String str, Icon icon)`

Swing Component

JCheckBox :

- The **JCheckBox** class provides the functionality of a check box.
- **JCheckBox** defines several constructors.

JCheckBox (String str)

Swing Component

JRadioButton :

- Radio buttons are a group of mutually exclusive buttons, in which only one button can be selected at any one time.
- **JRadioButton** provides several constructors;

JRadioButton (String str)

Swing Component

JComboBox:

- Swing provides a *combo box* (a combination of a text field and a drop-down list) through the **JComboBox** class.
- A combo box normally displays one entry, but it will also display a drop-down list that allows a user to select a different entry.

`class JComboBox<E>`

- Use following method to add list:

`void addItem(E obj)`

Swing Component

JTabbedPane:

- JTabbedPane encapsulates a tabbed pane. It manages a set of components by linking them with tabs.
- Selecting a causes the component associated with that tab come to the forefront.
- JTabbedPane defines three constructors:.

JTabbedPane ()

- Use following method to add panel:
void addTab(String name, Component comp)
- Often, the component added to a tab is a **JPanel** that contains a group of related components. It can hold a set of components.

Swing Menus

- The Swing menu system is supported by a group of related classes.

Class	Description
JMenuBar	An object that holds the top-level menu for the application.
JMenu	A standard menu. A menu consists of one or more JMenuItem s.
JMenuItem	An object that populates menus.
JCheckBoxMenuItem	A check box menu item.
JRadioButtonMenuItem	A radio button menu item
JSeparator	The visual separator between menu items.
JPopupMenu	A menu that is typically activated by right-clicking the mouse.

Demo

- Demo files will be uploaded in VLE

Lab Work

1. Re-work on Exercise #6 using Swing (Designing Forms)
2. Re-design the file menus using Swing Menus.
3. Design Multiple tabs like **Home, About, Contact, Vacancy, Link**
 - Design different page views for each tab.

Thank you!