Chapter 7 : User Interface Components with Swing, Introduction to JavaFX

What is Swing?

Swing in Java is a lightweight GUI toolkit which has a wide variety of widgets for building optimized window based applications. It is a part of the JFC(Java Foundation Classes).

It is build on top of the AWT API and entirely written in java. It is platform independent unlike AWT and has lightweight components.

It becomes easier to build applications since we already have GUI components like button, checkbox etc.

This is helpful because we do not have to start from the scratch.

The MVC(Model View Controller) Architecture and Swing

MVC(Model View Controller) Architecture :

Swing API architecture follows loosely based MVC architecture in the following manner.

- Model - Model represents component's data.
- View represents visual representation of the component's data.
- Controller takes the input from the user on the view and reflects the changes in Component's data.
- Swing component has Model as a seperate element, while the View and Controller part are clubbed in the User Interface elements. Because of which, Swing has a pluggable look and feel architecture.

Swing Architecture :

The design of the Swing component classes is based on the Model-View-Controller architecture, or MVC.

1. The model stores the data.
2. The view creates the visual representation from the data in the model.
3. The controller deals with user interaction and modifies the model and/or the view.

Usage of Layout Managers

The job of a layout manager is to arrange components on a container.

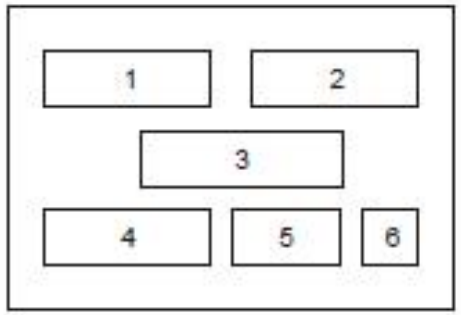
Each container has a layout manager associated with it. To change the layout manager for a container, use the setLayout() method.

Syntax :

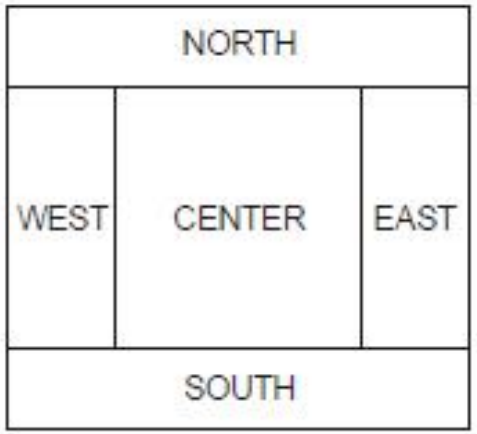
setLayout(LayoutManager obj)

The predefined managers are listed below :

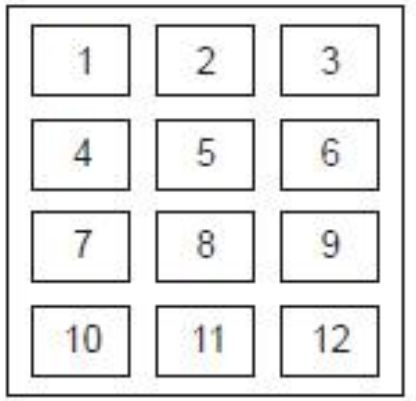
1. FlowLayout
2. BorderLayout
3. GridLayout
4. BoxLayout
5. CardLayout
6 .GridBagLayout



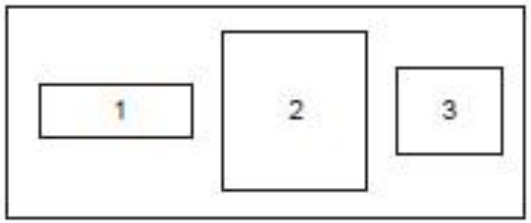
FlowLayout

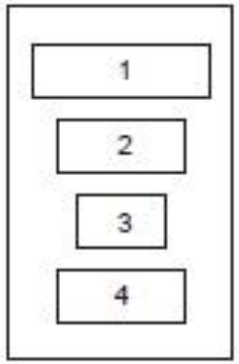


BorderLayout

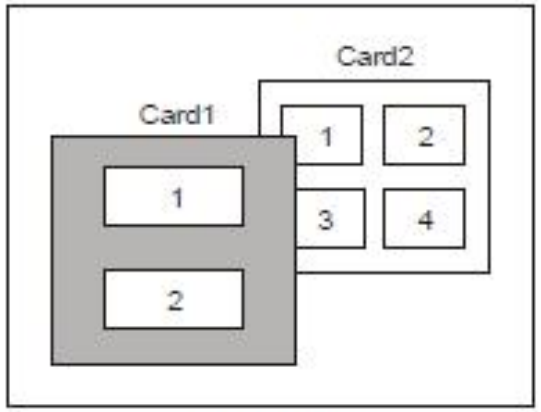


GridLayout

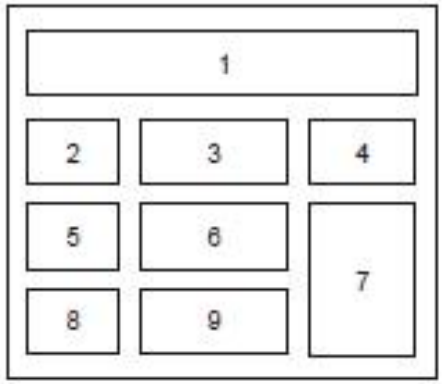




BoxLayout



CardLayout



GridBagLayout

Examples :

```
JPanel p1 = new JPanel()
p1.setLayout(new FlowLayout());
p1.setLayout(new BorderLayout());
p1.setLayout(new GridLayout(3,4));
```

Swing Components

Components	Description
JButton	Selectable component that supports text/image display
JLabel	For displaying text/images
JTextField	For editing and display of single-attributed textual content on a single line
JTextArea	For editing and display of single-attributed textual content
JCheckBox	Selectable component that displays state to user
JRadioButton	Selectable component that displays state to user; included in ButtonGroup to ensure that only one button is selected
JList	For selecting from a scrollable list of choices

Components	Description
JComboBox	For selecting from a drop-down list of choices
JMenu	Selectable component for holding menu items; supports text/image display
JPopupMenu	For holding menu items and popping up over components
JMenuItem	Selectable component that supports text/image display
JCheckBoxMenuItem	Selectable component for a menu; displays state to user
JRadioButtonMenuItem	Selectable component for menus; displays state to user; included in ButtonGroup to ensure that only one button is selected
JScrollBar	For control of a scrollable area

Dialogs (Message, confirmation, input), JFileChooser, JColorChooser

Types :

1. **Modal** : wont let the user interact with the remaining windows of application until first deals with it. Ex- when user wants to read a file, user must specify file name before prg. can begin read operation.

2. **Modeless dialog box** : Lets the user enters information in both, the dialog box & remainder of application ex- toolbar.

Swing has a JOptionPane class, that lets you put a simple dialog box.

Methods in JOption Class

1. static void showMessageDialog()- Shows a message with ok button.

2. static int showConfirmDialog()- shows a message & gets users options from set of options.

3. static int showOptionDialog- shows a message & get users options from set of options.

4. String showInputDialog()- shows a message with one line of user input.

JFileChooser

The object of JFileChooser class represents a dialog window from which the user can select file. It inherits JComponent class.

JFileChooser class declaration :

public class JFileChooser extends JComponent implements Accessible

Constructor	Description
JFileChooser()	Constructs a JFileChooser pointing to the user's default directory.
JFileChooser(File currentDirectory)	Constructs a JFileChooser using the given File as the path.
JFileChooser(String currentDirectoryPath)	Constructs a JFileChooser using the given path.

JColorChooser

The JColorChooser class is used to create a color chooser dialog box so that user can select any color. It inherits JComponent class.

JColorChooser class declaration :

public class JColorChooser extends JComponent implements Accessible

Constructor	Description
JColorChooser()	It is used to create a color chooser panel with white color initially.
JColorChooser(color initialcolor)	It is used to create a color chooser panel with the specified color initially.

Event Handling

Event handling is an important part of GUI based applications. Events are generated by event sources. A mouse click, Window closed, key typed etc. are examples of events.

All java events are sub-classes of java.awt.AWTEvent class.

Java has two types of events :

1. **Low-Level Events** : Low-level events represent direct communication from user. A low level event is a key press or a key release, a mouse click, drag, move or release, and so on.

Following are low level events.

Event	Description
ComponentEvent	Indicates that a component object (e.g. Button, List, TextField) is moved, resized, rendered invisible or made visible again.
FocusEvent	Indicates that a component has gained or lost the input focus.
KeyEvent	Generated by a component object (such as TextField) when a key is pressed, released or typed.
MouseEvent	Indicates that a mouse action occurred in a component. E.g. mouse is pressed, releases, clicked (pressed and released), moved or dragged.
ContainerEvent	Indicates that a container's contents are changed because a component was added or removed.
WindowEvent	Indicates that a window has changed its status. This low level event is generated by a Window object when it is opened, closed, activated, deactivated, iconified, deiconified or when focus is transferred into or out of the Window.

2. **High-Level Events** : High-level (also called as semantic events) events encapsulate the meaning of a user interface component. These include following events.

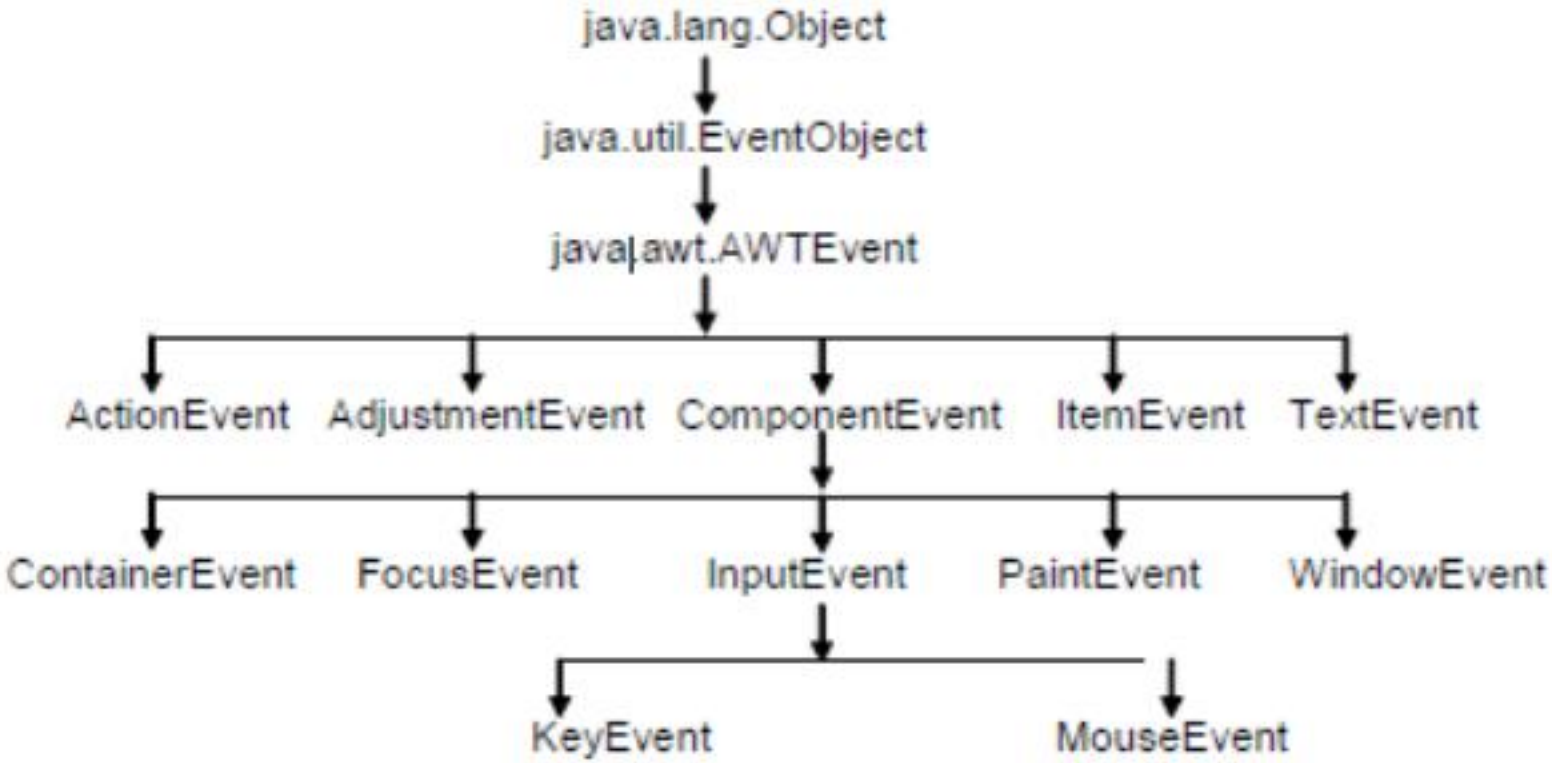
Event	Description
ActionEvent	Indicates that a component-defined action occurred. This high-level event is generated by a component (such as Button) when the component-specific action occurs (such as being pressed).
AdjustmentEvent	The adjustment event is emitted by Adjustable objects like scrollbars.
ItemEvent	Indicates that an item was selected or deselected. This high-level event is generated by an ItemSelectable object (such as a List) when an item is selected or deselected by the user.
TextEvent	Indicates that an object's text changed. This high-level event is generated by an object (such as TextComponent) when its text changes.

The following table lists the events, their corresponding listeners and the method to add the listener to the component.

Event	Event Source	Event Listener	Method to add listener to event source
ComponentEvent	Component	ComponentListener	addComponentListener()
FocusEvent	Component	FocusListener	addFocusListener()
KeyEvent	Component	KeyListener	addKeyListener()
MouseEvent	Component	MouseListener MouseMotionListener	addMouseListener() addMouseMotionListener()
ContainerEvent	Container	ContainerListener	addContainerListener()
WindowEvent	Window	WindowListener	addWindowListener()

Event	Event Source	Event Listener	Method to add listener to event source
ActionEvent	Button List MenuItem TextField	ActionListener	addActionListener()
ItemEvent	Choice CheckBox CheckBoxMenuItem List	ItemListener	addItemListener()
AdjustmentEvent	Scrollbar	AdjustmentListener	addAdjustmentListener()
TextEvent	TextField TextArea	TextListener	addTextLIstener()

Event class hierarchy



Adapters

There are some event listeners that have multiple methods to implement i.e. some of the listener interfaces contain more than one method.

For instance, the MouseListener interface contains five methods such a mouseClicked , mousePressed, mouseReleased etc. If you want to use only one method out of these then also you will have to implement all of them. Thus, the methods which you do not want to care about can have empty bodies. To avoid such thing, we have adapter class.
An Adapter class provides an empty implementation of all methods in an event listener interface.

Commonly used listener interfaces implemented by Adapter classes

Adapters Class	Listener Interfaces
ComponenetAdapter	ComponenetListener
ContainerAdapter	ContainerListener
FocusAdapter	FocusListener
KeyAdapter	KeyListener
MouseAdapter	MouseListener
MouseMotionAdapter	MouseMotionListener
WindowAdapter	WindowListener

What Is JavaFX?

JavaFX is a Java library used to develop Desktop applications as well as Rich Internet Applications (RIA).

The applications built in JavaFX, can run on multiple platforms including Web, Mobile and Desktops.

JavaFX is intended to replace swing in Java applications as a GUI framework. However, It provides more functionalities than swing.

Like Swing, JavaFX also provides its own components and doesn't depend upon the operating system.

It is lightweight and hardware accelerated. It supports various operating systems including Windows, Linux and Mac OS.