

GUI Programming in JAVA

Using Swing Control -I

Prepared By:

(Cs Basics Team)

Introduction

- A GUI (Graphical User Interface) is an interface that uses pictures and other graphical components along with text to interact with user in a familiar way.
- In JAVA, the GUI programming is done through Swing API (Application Programming Interface) enables look-and feel (L & F) environment.
- Swing controls are shipped with JAVA SE platform to build a GUI applications. It is part of Java Foundation Classes (JFC).
- JFC comprises five types of Libraries to assist Developer. These libraries are-
 1. Swing - Next Generation GUI Toolkit.
 2. AWT - Abstract Window Toolkit, basic toolkit of GUI. Swing internally borrows functionality from AWT.
 3. Accessibility: This package provides help to users who faces trouble using traditional user interface, ex. Visually impaired people.
 4. 2D API- This contains features for implementing various painting styles, fonts and shapes etc.
 5. Drag and Drop- This package allows Drag and Drop features in GUI applications.

Types of Swing Controls

□ Component:

A Swing component is a self-contained graphic entity that can be customized and inserted into applications. Ex. JLabel,

□ JTextField, JButton etc.

Container

A container is special type of component that can hold other components. Ex. JFrame, JPanel, JDialog etc.

Container controls are also divided into-

1. Top Level Container: which can be placed on the Desktop. Ex. JFrame

Various Swing Controls

Basic Controls	Simple components used to get input from the user	jButton, jCheckBox, jList, jMenu, jRadioButton, jSlider, jSpinner, jTextField, jPasswordField
Interactive Display	Displays the information in formatted way	jColorChooser, jEditorPane, jTextPane, jFileChooser, jTable, jTextArea, jTree
Un-editable Information Display	Display information to user which can't be edit	jLabel, jProgressBar, jSeparator, jToolTip
To-Level Containers	Contains other controls on the Desktop	jApplet, jDialog, jFrame
General Purpose Containers	Contains general purpose utility.	jPanel, jScrollPane, jSplitPane, jTabbedPane, jToolBar
Special Purpose Containers	Contains some predefined specific role.	jInternalFrame, jLyeredPane, RootPane

Layout Managers

- Layout managers enable you to control the way in which visual components are arranged in GUI forms by determining the size and position of components within containers.
- There are seven types of layout are available-
 - Flow Layout
 - Grid Layout
 - Card Layout
 - Spring Layout
 - Border Layout
 - GridBag Layout
 - Box Layout

1. Flow Layout

- It arranges components in a container like words on a page. It fills the top line from left to right and then top to bottom.
- Features:
 - Components are given their preferred size.
 - Components are arranged in order as they are attached i.e. first components appears at top left.
 - If container is not wide enough to display all the components, it is wrapped around the line.
 - Vertical and horizontal gap between components can be controlled.
 - Components can be left, center or right aligned.

2. Border Layout

- It arranges all the components along the edges or the middle of the container i.e. top, bottom, right and left edges of the area.
- Features:
 - Components added to the top or bottom gets its preferred height, but its width will be the width of container.
 - Components added to the left or right gets its preferred width, but its height will be the remaining height of container.
 - Components added to the center gets neither its preferred height or width. It covers the remaining area of the container.

3. Grid Layout

- It arranges all the components in a grid of equally sized cells, adding them from the left to right and top to bottom.
- Features:
 - Only one component can be placed in a cell.
 - Each region of the grid will have the same size. When container is resized, all cells are automatically resized.
 - Order of placing components in cell is determined as they were attached. No
 - components are their preferred height

or width i.e. all are assumed as same size.

4. GridBag Layout

- It is powerful layout which arranges all the components in a grid of cells and maintains the aspect ratio of the object whenever container is resized. In this layout cells may be different in size.

- Features:

- It assigns consistent horizontal and vertical gap among components. It

5. Box Layout

- It arranges multiple components in either vertically or horizontally, but not both. Components are arranged from left to right or top to bottom.
- Features:
 - Components are displayed either horizontally or vertically.
 - It do not wrap components like Flow layout.
 - If the components are aligned horizontally, the height of all components will be same, and equal to the largest sized components.
 - If the components are aligned vertically, the width of all components will be same, and equal to the largest width components.

6. Card Layout

- It arranges two or more components having the same size. Components are arranged in a deck, where all cards of the same size and only top card is visible at any time.
- Features:
 - It treats each components as a card. Only one card (top) is visible.
 - First component added in the container will be kept at the top of the deck.
 - The default gap at the left, right and top, bottom edged is zero. which are to be arranged The card Components are displayed either horizontally or vertically.

7. Spring Layout

- It is rarely used layout in which it arranges components they may have fixed spaces. NetBeans IDE 6.5.1 does not support this layout.
- Instead of Spring layout two other layouts are provided in NetBeans 6.5 version.

- Absolute Layout:

- It places the components where they are placed.

- Null layout:

- It is used to design a form without any layout manager at all.

By default, NetBeans uses Group Layout — also called Free Design. In which a container groups item as a single entity.
Ex. Java pannel

How to use Layout Manager

There are two ways, as you can use layout manager-

□ From GUI Builder-

- Right click on Frame/panel.
- Choose desired layout from sub menu of Set Layout of the context menu.
- The IDE applies the selected layout.

□ From Inspector Window

- Right click on container name in Inspector Window.
- Choose desired layout from sub menu of Set Layout of the context menu.
- The IDE applies the selected layout.

Controlling/Setting Layout Manager:

After adding Layout manager, you may customize the setting of the layout manager.

- To set up the properties, click on Layout Node in Inspector Window and get Layout Properties box.
- Set various properties to change the existing setting.

Events Handling in a GUI Application

An event is occurrence of some activities either initiated by user or by the system. In order to react, you need to implement some Event handling system in your Application. Three things are important in Even Handling-

- Event Source:

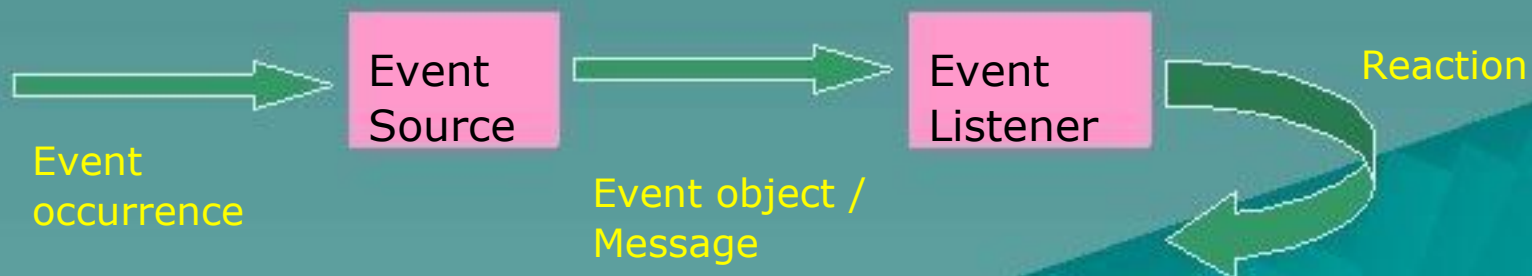
It is the GUI component that generates the event, e.g. Button.

- Event Handler or Event Listener:

It is implemented as in the form of code. It receives and handles events through Listener Interface.

- Event Object or Message:

It is created when event occurs. It contains all the information about the event which includes Source of event and type of event etc.



Major Events in Swing

Java API offers numerous types of events and event listeners. The commonly used events are-

Action Event:

- The Action events occurred when user completes an action on components like JButton, JCheckBox, JTextField etc. To handle Action Event, ActionListener Interface is used.

Focus Event:

- The Focus event occurred when any components gains or loses focus on components like JButton, JCheckBox, JTextField etc. To handle Focus Event, FocusListener Interface is used.

Key Event:

- The Key Event occurred when a key is pressed on the keyboard on input enabled components, JTextField etc. To handle Key Event, KeyListener Interface is used.

Mouse Events:

- When a mouse is clicked, entered or leaved a component area of a control then Mouse event is generated. Any component can generate this event. To handle Mouse Event, MouseListener Interface is used.

Mouse Motion Events:

- When a mouse is moved over a control then Mouse Motion event is generated. Any component can generate this event. To handle Mouse Motion Event, MouseMotion Listener Interface is used.

Window Event:

Major Event Listeners and their Events of JAVA Swing

Listener Interface	Listener Interface methods	Event	Major event-method
Action Listener	actionPerformed()	ActionEvent [Button,CheckBox,Combo and Text field etc.]	getAction() getSource()
Focus Listener	focusGained() focusLost()	FocusEvent [All component receives focus]	getComponent()
Item Listener	itemStateChanged()	ItemEvent [Button, CheckBox, Combo and Menu etc.]	getItem() getStateChange()
Key Listener	keyTyped() keyPressed() keyReleased()	KeyEvent [Button,CheckBox,Combo and Text field etc.]	getKeyChar()
Mouse Listener	mouseClicked() mousePressed() mouseReleased()	MouseEvent [All components]	getClickCount() getComponent()
MouseMotion Listener	mouseDragged() mouseMoved()	MouseMotionEvent [All components]	getClickCount() getComponent()
Window Listener	windowOpened() windowClosed()	WindowEvent [All top level Container]	getWindow()

How to use Event Handlers in NetBeans

There are two ways, as you can define events in NetBeans-


- **Using Property Sheet-** - Select the component in the Inspector Window or in Design View by clicking on it.
 - Click on Event buttons in Properties Window.
 - Click the value of desired Event in the list, where <none> is displayed.
 - Now add event handler code (//TODO code//) by clicking on the left column of property window. It will open Code Window, where you can type commands/code.
 - If you want, you may give some other name to event's handler by clicking on □ button under Event tab and giving new name. This can be added by Add button.
- **Using Contextual Menu-**
 - Right Click on desired control of the Form in Design view. - Choose desired Event from the Context Menu and its Sub Menu.
 - When you click on desired Event it opens source Code editor with default Handler Name, where you can type //TODO code for the Handler.

How to use Event Handlers in NetBeans .

As you attached an Event along with Listener, you will find a code window along with prototyped method to perform actions defined by you. You may write commands to be executed in //TODO section.



Giving Different Name to Event Handler Method:

- If you want to give some other/own name to event handler then do this-
 - Click on Event's ellipsis () button under Event tab of Property Window. It will open a Dialogue box.
 - Enter the Event Handler Name by clicking ADD button to add a new name to the Handler. It will create a Prototyped method containing name given by you.
 - Now click on control on the frame to open Code Window, Write the code you want to execute in the editor.

Swing Controls - An Object Oriented Approach

As you know that JAVA is an Object Oriented Language, in which all the GUI components and programming is based on OOP approach. Let us learn two important terms of OOP.

□ Object:

- An Object is an identifiable entity with some characteristics and behaviour.
- Any thing that is visible or tangible, intellectually, action or thought may be assumed as an Object in OOP.
- Each Object should have a unique identity i.e. name, characteristics (data values) and Behaviour (functions or Methods) which are wrapped as a unit like a Real world Objects.
- Ex. You may assume yourself as an object. Your name is object's Identity, Your Weight, Color, Height may be Characteristics and Your activities like Talk, Walk, Run, Eat, Sleep may be Behaviour.

- - The Behaviour of an Object is implemented by Method or function in OOP.

Class:

- Class is a Blue print of an object that represents a set of similar objects. Also we can say that collection of similar types of object is known as Class, and an Object is an instance of a Class.

- Ex. Student, Man, Bird represents a Class. But You are an Object, since you are the member of Class student.

Swing Controls - An Object Oriented Approach..

Now you can understand that all the Libraries in JAVA, can be assumed as Classes. So, Swing controls are implemented as Classes in the JAVA. JAVA Swings offers many Classes to GUI components.

Ex. If you are using a Command Button name jButton1, it is an object because it is an instance of a class JButton. Now jButton1, being an object offers various methods like setText() and getText() etc. Every Swing controls used in the frame works like an objects, which belongs to a Class. Each object offers various type of Method, which can be used for specific purpose in the GUI application. E.g if you are writing-
`jTextField1.setText("Hello")`

In this statement, jTextField1 is an object which belongs to Swing control named jTextField and setText() is a Method.

A Method is attached with a object by using (.) and a Mathod may require parameters like "Hello" which to be manipulated by setText() method.

- All the JComponents are Classes, some of them used as a Container like JApplet, JDialog, JFrame and JWindow.

□

Working with Container Control-- JFrame

- Every Swing Application must have at least one Top Level container (JFrame, JApplet, JDialog). A Top Level Container may have Mid-Level Container like Content Pane (JPanel, JMenuBar, JScrollBar etc.) or Components (JButton, JTextField etc.)
- A Frame (JFrame) is a Top Level (Desktop) container control having Title Border and other Properties.

Properties	Value	Description
Title	Text	Sets the title (appears on the top) of the frame
Cursor	Crosshair, East Resize, West Resize, Northwest Resize, Move, Hand, Wait, Default cursor	Specifies the type of mouse cursor when mouse moves on the frame.
Icon Image	Small Image file (.png, .ico etc)	Sets the icon image appears on the title bar.
Resizable	True /false	If checked, allows resizing of the frame
DefaultCloseOperation	DO_NOTHING, HIDE, DISPOSE, EXIT_ON_CLOSE	Defines the action when close button is pressed.

Jframe- Setting Icon

To set the ImageIcon property of the Frame, do the following-

- ❑ Click on the frame and select ImageIcon in property box.
- ❑ Click on ellipse (□) to open the dialog box.
- ❑ Select Custom Code from the combo.
- ❑ Define the path of image file like-
`new ImageIcon("c:\\myicon.ico").getImage()`
- ❑ Press the OK button.
- ❑ Now open Code Window and go to top of the page and write a command-
`import javax.swing.ImageIcon;`
- ❑ Save the application and Run.

Working with Panel-- JPanel

- A Panel is container that holds other components displayed on the frame.
- To add Panel, just drag JPanel component on the frame and resize it.
- Drag other components (JButton, JTextFields etc.) from the Swing Control Box and drop it onto panel.
- You can apply Layouts on the panel also, by selecting Layout Manager from Right click Context menu.

Properties	Value	Description
Background	Color	Sets the background color.
Border	No Border, Bevel Border, Compound Border, Empty order, Etched Border, Line Border, Matte Border, Soft Bevel Border, Titled Border	Specifies the type of Border applies on the boundary of the panel.
Foreground	Color	Sets the foreground color.
ToolTipText	Text	Sets the text for tooltip.
Minimum Size	X, Y values	Defines the minimum width and height (x,y) in Twips(1/1440 inch)
Maximum Size	X, Y values	Defines the maximum(x,y) size.
Preferred Size	X, Y values	Defines the preferred (x,y) size.

Using HTML in Swing Control

We can HTML code in Text Property of various Swing Controls, to make text more decorative by mixed fonts, colour and formatting like bold, italic etc. HTML formatting can be used in Text of Buttons, Tool tips, tables, menu items etc. Do the following steps- Select the Text property of the control.

- In text editing window, write the HTML code along with text to be appeared. Ex.

<HTML> How are <u>You</u> it
will display - How are **You** —

- Commonly used HTML tags like
, <U>, , <I>, <P> etc. can be used.

Working with Push Buttons-- JButton

- A button belongs to JButton class of Swing control API.
- It is mostly used action component, and can be used to trigger the associated events/methods in the application, when user clicks.

Properties	Value	Description
Background	Color	Sets the background color. It works only when <u>contentAreaFilled</u> is set to True.
Border	Border setting as in Jpanel	Specifies the type of Border applies on the boundary of the panel.
Foreground	Color	Sets the foreground color.
ToolTipText	Text	Sets the text for tooltip.
Minimum Size	X, Y values	Defines the minimum width and height
Maximum Size	X, Y values	Defines the maximum(x,y) size.
Preferred Size	X, Y values	Defines the preferred (x,y) size.
Horizontal & VericalAlignment	Right, Trailing, Left, Leading, Center (Horizontal) Top, Bottom , Center (vertical)	Specifies the horizontal and vertical alignment of text.
Text	Text	Caption of button
mnemonic	Shortcut or Access key	Assign Shortcut key (Alt +key)
enabled	True/False	Determines whether Active or not
ActionCommand	String/Text	Sets the string that identify the command. Default value is Text property

Working with JButton

Assigning Access keys to a Button:-

- - You may assign Access key (Shortcut key) to operate a button by Key board using Alt+ Key.
- Click on mnemonic property and set letter to be assigned e.g. P for Print.
- Alternatively, click on ellipse (□), choose Custom code and type `KeyEvent.VK_P`
- For underlining Key letter, use HTML tag in Text property. E.g. `<html><u>P</u>rint` to get Print. _

□ Adding Event Handlers to a Button:-

- You may define Action Event, Item Event, Mouse Event, Key Event and Mouse Motion Event to a button.
- Generally, `ActionPerformed()` Event is handled like-
`jButton1.ActionPerformed(java.awt.event.ActionEvent evt)`
- You can also rename the Event handler method.

Working with JButtons..

□ Commonly used Component methods of JButton.

Method	Description
Void setText(String)	Sets the text displayed on the button. Ex. <code>JButton1.setText("You Clicked Me");</code>
String getText()	Returns the text displayed on the button. Ex. <code>String result=JButton1.getText(); JLabel1.setText(result);</code>
Void setIcon(icon)	Sets the icon file to be displayed. Ex. <code>JButton1.setIcon(new ImageIcon("c:\\abc.png"));</code>
Void setSelected(Boolean)	Sets the button to appear as selected, mostly used with check boxes. Ex. <code>JButton1.setSelected(true);</code>
Boolean isSelected()	Returns the status whether it is selected or not. Ex. <code>If (JButton1.isSelected())=true) □ ..</code>
String getActionCommand() Void doClick(int)	Returns the string set in actionCommand property of the button. Ex. <code>JLabel1.setText(JButton1.getActionCommand());</code> Performs click operation on the button. Int is optional parameter, represent time (milisecond) that button look pressed. Ex. <code>JBitton1.doClick();</code>

Understanding User defined methods/functions

- As you know, object's behavior is simulated by methods in OOP. Methods can be assumed as set of commands, which can be invoked by sending a Message or by calling them.
- Message to an object, is a call or request to perform some specified action.
- A method can be written as -

```
<return type> methodname(parameters list)
{
    ..
}
```

Void is used when it returns nothing.

Once a method has been written, it can be called in any other method or event like-

```
<variable/object>= methodname(value list)
```

Example:

```
int Sum(int x, int y)
{
    int z= x+y;
    return(z);
}
```

It may be called as-

```
int c=sum(a,b);
int c=sum(3,5);
```