Chapter 4:

JAVA GUI Programming

Revision Tour -II

Informatics Practices

Class XII

By- Rajesh Kumar Mishra

PGT (Comp.Sc.)

KV No.1, AFS, Suratgarh

e-mail : rkmalld@gmail.com

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).

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
2. Non-Top Level Containers: These can be displayed within the context of another top-level containers. Ex. jPanel

Various Swing Controls

Basic Controls	Simple components used to get input from the user	jButton, jCheckBox,jComboBox 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
Top-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 ration 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 allows you to specify a default alignment for components within columns or rows.

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 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 Lay out manager, you may customize the setting of the layout manager.

- To set up the properties, click on Layout Node in Inspector Window and get Lay out 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:

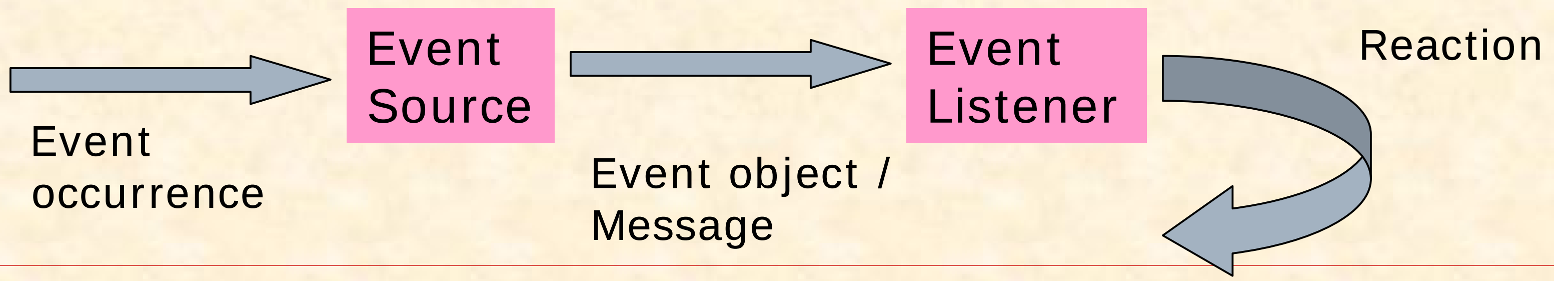
  It is the GUI component that generates the event, e.g. Button.

- 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.



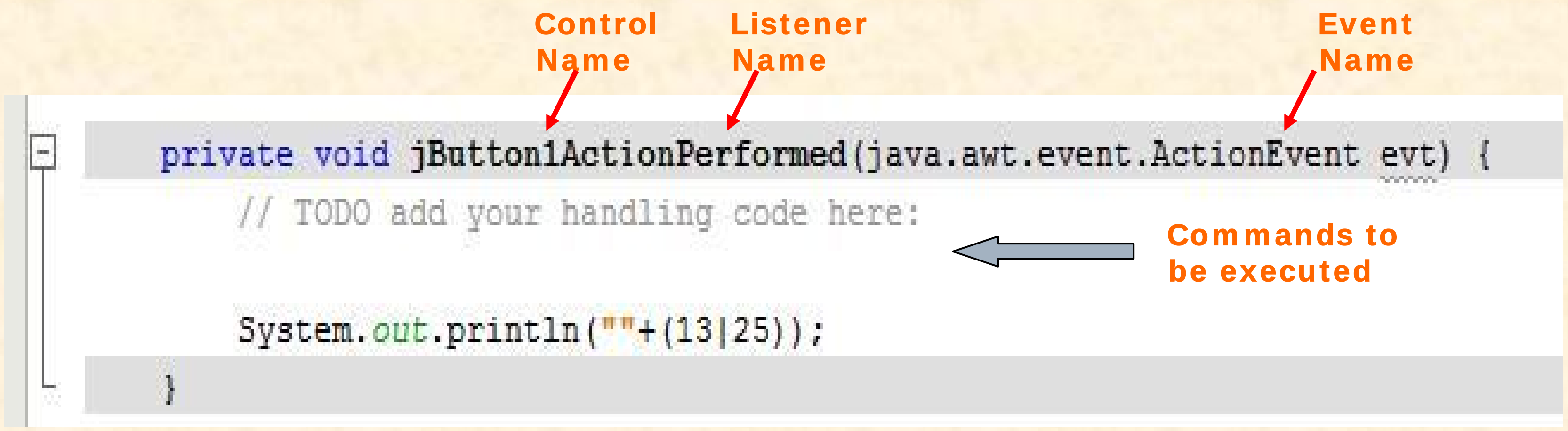
How to use Event Handlers in NetBeans

There 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.
- **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 Handeler 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 found a code window along with prototyped method to perform actions defined by you. You may write commands to be executed in //TODO section.



```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    System.out.println(""+(13|25));
}
```

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
defaultCloseOpe ration	DO_NOTHING, HIDE, DISPOSE, EXIT_ON_CLOSE	Defines the action when close button is pressed.

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.
MinimumSize	X, Y values	Defines the minimum width and height (x,y) in Twips(1/1440 inch)
MaximumSize	X, Y values	Defines the maximum(x,y) size.
PreferredSize	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 <b><u>You</b></u>

  it will display – How are **<u>You</u>**
- Commonly used HTML tags like <BR>, <U> , <B>, <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 **contentAreaFilled** 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 tool tip.
Text	Text	Caption of button.
mnemonic	Shortcut or Access key	Assign Shortcut key (Alt +key).
enabled	True/False	Determines whether Active or not.
font	Font name	Sets the font for the text of button.

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.
  - 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. jButton1.setText(“You Clicked Me”);

String getText()	Returns the text displayed on the button.
	Ex. String result=jButton1.getText();
	jLabel1.setText(result);

void setIcon(icon)	Sets the icon file to be displayed. Ex. jButton1.setIcon(new ImageIcon(“c:\\abc.png”));
void setSelected(Boolean)	Sets the button to appear as selected, mostly used with check boxes. Ex. jButton1.setSelected(true);
boolean isSelected()	Returns the status whether it is selected or not. Ex. If (jButton1.isSelected()=true) …..

Working with jLabel control

A Label control belongs to JLabel class and used to display non-editable text. The text to be displayed is controlled by text property (design time) and setText() method at run time. jLabel offers the following features-

- It can display Text or Image or both.
- It may have bordered appearance.
- Supports HTML for formatted text.

Commonly used Properties & Methods of jLabel

Properties	Value	Description
background	Color	Sets the background color. It works only when **opaque** is set to True.
foreground	Color	Sets the foreground color.
ToolTipText	Text	Sets the text for tool tip.
Text	Text	Sets the text to be displayed.
Font	Font name and size	Defines the font and size of text.
enabled	True/False	Determines whether Active or not
Icon	Image file to be displayed	Specifies the image file to be displayed.

Methods	Description
Void setText(String)	Sets the string of text to be displayed. Ex. jLabel1.setText(“I am OK”);
String getText()	Returns the text displayed by the label. Ex. String st=jLabel1.getText();
Void setIcon(Image)	Sets the image file to be displayed. Ex. jLabel1.setIcon(new ImageIcon(“c:\\abc.gif”);

Displaying Image with jLabel

- Setting up Image at Design Time :-
  - Add jLabel control and click on ellipse (...) of Icon property in property window.
  - In the dialogue box, select Image chooser option.
  - Specify the path and file name in External Image option.
  - Open source editor and go to top of the code and write-

    import javax.swing.ImageIcon;
- Setting up Image at Run time:-
  - Import the javax library by placing following command at top of the code.

    import.javax.swing.ImageIcon;
  - Use the following command in a Event method where image to be displayed or changed.

    jLabel.setIcon(new ImageIcom(“c:\\abc.gif”))

Working with jTextField control

A jTextField is a versatile control, used to get input from user or to display text. It is an object of jTextField class and allow the user to enter a single line of text.

jTextField offers the following features-

- You can insert and select text.
- You can scroll the text, if not fit in visible area.
- You can use selected text in other application using clipboard.

Commonly used Properties of jTextField

Properties	Value	Description
background	Color	Sets the background color. It works only when opaque is set to True.
foreground	Color	Sets the foreground color.
Text	Text	Sets the text to be displayed.
Font	Font name and size	Defines the font and size of text.
enabled	True/False	Determines whether Active or not
editable	True/False	Allow user to edit text, if set to true.

Methods	Description
Void setText(String)	Sets the string of text to be displayed. Ex. jTextField1.setText("I am OK");
String getText()	Returns the text displayed by the label. Ex. String st=jTextField1.getText();
boolean isEditable( )	Returns the setting whether it is editable or not. Ex. Boolean b=jTextField1.isEditable( );
boolean isEnabled( )	Returns the setting whether it is enabled or not. Ex. Boolean b=jTextField1.isEnabled( );

Working with jCheckBox control

A jCheckBox control belongs to JCheckBox class of Swing controls. It indicates whether a particular condition is on or off. You can use Check boxes to give users true/false or yes/no options.

Check Boxes may works independently to each other, so that any number of check boxes can be selected at the same time.

Some features of jCheckBox control's are-

- It can be used to input True/False or Yes/No typed input to the application.
- Multiple check boxes can be selected at the same time.

Commonly used Properties of jCheckBox

Properties	Value	Description
background	Color	Sets the background color.
foreground	Color	Sets the foreground color.
Text	Text	Sets the text to be displayed.
Label	Text/Picture	Sets the text/Picture to be displayed.
Font	Font name and size	Defines the font and size of text.
enabled	True/False	Determines whether Active or not
mnemonic	Character	Specifies the shortcut (access) key
selected	True/false	Check box will be selected, if set to true. (default is false)
Button Group	Button Group name	Adds Check Boxes in a Group

You must add a **ButtonGroup control** to the frame to group the check boxes by using Button Group property of the check box.

Commonly used Methods of jCheckBox

Methods	Description
Void setText(String)	Sets the string of text to be displayed. Ex. jCheckBox1.setText(“Computer”);
String getText()	Returns the text displayed by on the check box. Ex. String st=jCheckBox1.getText();
Void setEnabled(boolean)	Sets the check box enables, if true is given. Ex. jCheckBox1.setEnabled(true)
Boolean isEnabled( )	Returns the state whether check box is enabled. Ex. Boolean st=jCheckBox1.isEnabled(true)
void setSelected(boolean)	Sets the check box selected, if true is given. Ex. jCheckBox1.setSelected(true)
Boolean isSelected( )	Returns the state whether check box is selected or not. Ex. Boolean st=jCheckBox1.isSelected(true)

Working with jRadioButton control

A jRadioButton control belongs to JRadioButton class of Swing controls. It is used to get choices from the user. It is grouped control, so that only one can be selected at a time among them.

Radio Button works in group, so that they must be kept in a ButtonGroup container control like so that only one can be selected at the same time.

Some features of jRadioButton control’s are-

- It can be used to input choices typed input to the application.
- Only one Radio button can be selected at a time.
- They must be kept in a Button Group container control to form a group.

Commonly used Properties of jRadioButton

Properties	Value	Description
Background	Color	Sets the background color.
Foreground	Color	Sets the foreground color.
buttonGroup	Name of control	Specifies the name of group to which Radio Button belongs.
Text	Text	Sets the text to be displayed.
Label	Text/Picture	Sets the text/Picture to be displayed.
Font	Font name and size	Defines the font and size of text.
enabled	True/False	Determines whether Active or not
mnemonic	Character	Specifies the shortcut (access) key
selected	True/false	Check box will be selected, if set to true. (default is false)

Commonly used Methods of jRadioButton

Methods	Description
Void setText(String)	Sets the string of text to be displayed. Ex. jRadioButton1.setText("Science");
String getText()	Returns the text displayed by on the check box. Ex. String st=jRadioNutton1.getText();
Void setEnabled(boolean)	Sets the Radio Button enables, if true is given. Ex. jRadioButton1.setEnabled(true)
Boolean isEnabled( )	Returns the state whether radio button is enabled. Ex. Boolean st=jRadioButton1.isEnabled(true)
void setSelected(boolean)	Sets the Radio Button selected, if true is given. Ex. jRadioButton1.setSelected(true)
Boolean isSelected( )	Returns the state whether Radio Button is selected or not. Ex. Boolean st=jRadioButton1.isSelected(true)

Working with jList control

A List (or List box) is box shaped control containing list of objects, from which single or multiple selection can be made. jList control offers the following features-

- A box shaped control capable to displaying a list of choices (Text or graphics/images)
- It Allows single or multiple selection of items using mouse.
- Equipped with built-in scroll bar to view a large list.
- valueChanged() method of ListSelection Listener is used to handle the JList events

How to add jList control with Frame

A jList can be attached with frame in following way-

- Select jList control from Swing palette , and drag and drop on the frame.
- Select the jList control and click on (...) button of Model property. A dialog box appears.
- Type list of choices in ListModelEditor dialog box. Finally press OK button.
- List Box is displayed on the frame with typed choices.

Commonly used Properties of jList

Properties	Value	Description
Background	Color	Sets the background color.
Foreground	Color	Sets the foreground color.
model	Items for Choice	Specifies the items to be displayed as a choice.
selectionMode	SINGLE SINGLE_INTERVAL MULTIPLE_INTERVAL	User may select single item. User may select Single range of items by holding SHIFT key. User may select Multiple range of Items by holding CTRL key
selectedIndex	Value	Specifies the index of Items to appear selected (default is -1 since no items is selected.)
selectedIndices	values	Specifies the indices of selected items in form of an array to show multiple items. (default -1)
font	Font name	Specifies font's name and size etc.
enabled	True/False	Specifies that list will be active or not.

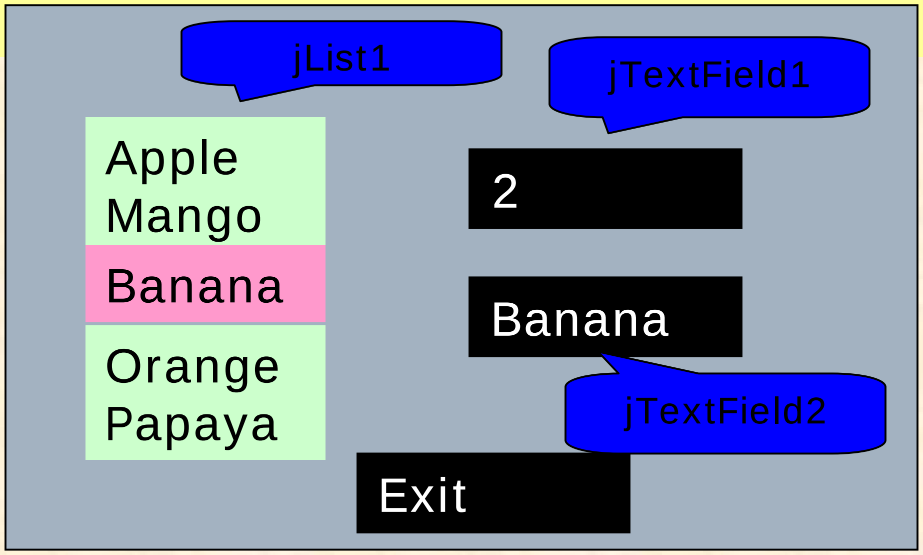
Commonly used Methods of jList

Methods	Description
Void clearSelection()	Clears the selection in the list. Ex. jList1.clearSelection();
int getSelectedIndex()	Returns the index of selected items in single selection mode. Returns -1, if selection is not made.
Object getSelectedValue()	Returns the value or selected items. String st= (String) jList1.getSlectedValue();
Object[] getSelectedValues()	Returns the values of selected items (multiple selection) String st[]= (string) jList1.getSlectedValues();
Boolean isSelectedIndex(int)	Returns True if given Index is selected. Ex: if (jList1.isSlectedIndex(2)) {…}
Other methods are- isEnabled(), setVisible(), setEnabled() etc. as used earlier.	

How to handle Selections in the jList

☐Single Selection

Suppose we want to display the index and value of selected items i.e. 2 and 'Banana' in Text Fields when user selects an items in the list.

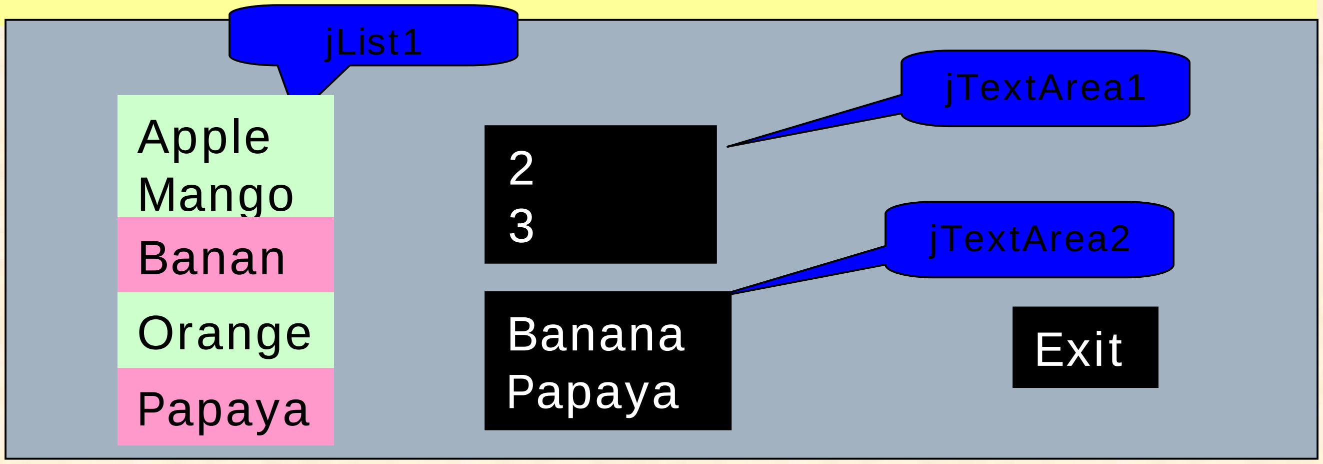


Just Right Click on jList control and choose
Events->ListSelection->ValueChanged
Write the following code in //TO DO Section............

```
        int x=jList1.getSelectedIndex();
        String st= (String ) jList1.getSelectedValue();
        jTextField1.setText(""+x);
        jTextField2.setText(st);
```

How to handle Selections in the jList

Multiple Selection



Go to Events->ListSelection->ValueChanged of JList1 control
Write the following code in //TO DO Section............

```
int x[]=jList1.getSelectedIndices();
Object st[]= jList1.getSelectedValues();
for(i=0;i< x.length ;i++)
    jTextArea1.append(“”+x[i]+’\n’);
for(i=0;i<st.length ;i++)
    jTextArea2.append((String) st[i]+’\n’);
```

Adding/Removing Items at run time (Dynamic List)

An items in the list can be deleted or inserted run time by using special methods available in DefaultListModel class. To implement Dynamic list follow the following steps-

- To import the DefaultListModel class, type the line at top -

  import javax.swing.DefaultListModel;

- Select jList control and go to Model (…) Property and choose Custom code in the dialog box.
- Type the following line in the dialog box and Press OK button.

  new DefaultListModel()

- The following Methods may be used to manipulate the list at run time-

Methods	Description
Int size()	**Returns the number of items currently in the list. Ex. int x=jList1.size();**
Boolean isEmpty()	**Returns True if the list is empty (no items)**
Void addElement (String)	**Adds given string at the end of the list.**
Void insertElementAt(string.position)	**Adds given item at given index.**
Boolean removeElement(string/index)	**Removes specified items from the list**
Void removeElementAt(object,position)	**Removes given object from given index.**
Void removeAllElements()	**Removes all entries from the list.**
Object getModel()	**Returns the ListModel of the list.**

Adding/Removing Items at run time (Dynamic List)

The following steps should be followed in the event handler of the button, where you want to implement the command.

1. Make a model object as-

```
DefaultListModel dlm = (DefaultListModel) jList1.getModel();
```

2. Use following command as per requirement-

```
// add item in the end of the list//
dlm.addElement("Apple");
// Insert item at given index in the list//
dlm.addElement("Apple",2);
// Remove selected item from the list//
dlm.removeElement(jList1.getSelectedIndex());
// Remove item from given position from the list//
dlm.removeElementAt(4);
// Remove given item from from the list//
dlm.removeElement("Banana");
```

3. Reflect all changes to the list

```
jList1.setModel(dlm)
```

Working with Image List (List of Images)

The following steps should be followed to implement the jList containing Images choices instead of Text choices.

1. Add JList Control on the Frame.
2. Right Click on the jList and choose Customised Code from the pop up menu.
3. Select Custom Property from the second drop down box of Code Customizer dialog Box.
4. Now replace the following pre written line with the following code.

```
String[] strings={“Item 1”, ”Item 2”, “Item 3”, …};

 // Write New code like this //
ImageIcon[] String = {new ImageIcon(“c:\\aaa.jpg”),
                      new ImageIcon(“c:\\bbb.jpg”),
                      new ImageIcon(“c:\\ccc.jpg”)
                     };
```

5. Press Ok Button to closed the dialog box.
6. Open Source editor and write the following code at the top.

```
import javax.swing.ImageIcon;
```

7. Now save the application and run.

Working with jComboBox control

A jComboBox (TextField + List Box) is a control which offers the list of choice which can be selected through a drop-down list.

By default it is an un-editable control, but we can make it editable by setting ‘editable‘ property as True.

jComboBox1ActionPerformed(..) method can be handled when user selects an item from the combo.

Difference Between List & Combo Box

- In Combo Box, user can select and edit an item but in List, Items can be selected and can not be edited.
- List does not offers Text Field where as Combo Box offers Text Field.
- Combo Box offers a Drop-down list, so that it takes less space in the frame, but List this feature is not available.
- List allows more than one items to be selected, but in Combo Box, only one item can be selected.

Commonly used Properties of jComboBox

Properties	Value	Description
Background	Color	Sets the background color.
Foreground	Color	Sets the foreground color.
model	Items for Choice	Specifies the items to be displayed as a choice.
selectedIndex	Value	Specifies the index of Items to appear selected (default is -1 since no items is selected.)
selectedItem	String/values	Specifies the indices of selected items.
font	Font name	Specifies font’s name and size etc.
editable	True/False	If True, you can edit/type new value or choice in the Combo Box.
enabled	True/False	Specifies that list will be active or not.

Commonly used Methods of jComboBox

Methods	Description
Void addItem(string)	Adds an item to the choice list at the end. Ex. jComboBox1.addItem("Banana");
Void insertItemAt(string,int)	Insert a given item at given index. Ex. jComboBox1.insertItemAt("Banana",2);
int getItemCount()	Returns the number of items in the combo Box. int x=jComboBox1.getItemCount();
String getItemAt(int)	Returns the items at specified index. String st=jComboBox1.getItemAt(2);
int getSelectedIndex()	Returns the index of selected items.
String getSelectedItem()	Returns the selected items. String st= (String) jComboBox1.getSlectedItem();
Boolean isEditable()	Returns True, if Combo box is editable.
Void removeAllItems()	Removes all the items from combo.
Void removeItem(String)	Removes specified items from the combo. Ex. jComboBox1.removeItem("Banana");
Void removeItemAt(index)	Removes items for given index from the combo. Ex. jComboBox1.removeItemAt(2);

Working with jTextArea control

A jTextArea control is a multi-line text component, used to get input from user or to display text. It is an object of JTextArea class.

By default, it does not wrap (move next line) lines of text like word processor, if line goes beyond the boundary. Some features are-

- You can insert and select multiple line of text.
- You can wrap text, if not fit in visible area.
- You can use selected text in other application using clipboard.

Commonly used Properties of jTextArea

Properties	Value	Description
Background	Color	Sets the background color.
Foreground	Color	Sets the foreground color.
lineWrap	True/ false	Defines Wrapping feature enable/disable
rows	number	Set the number of rows of in text area
columns	number	Sets the number of columns
Text	Text	Sets the text to be displayed.
Font	Font name and size	Defines the font and size of text.
enabled	True/False	Determines whether Active or not
editable	True/False	Allow user to edit text, if set to true.

Commonly used Methods of jTextArea

Methods	Description
Void setText(String)	Sets the string of text to be displayed. Ex. jTextArea1.setText(“I am OK”);
String getText()	Returns the text displayed by the label.
Void setEditable(Boolean)	Sets the TextArea editable.
boolean isEditable( )	Returns the setting whether it is editable or not. Ex. Boolean b=jTextArea.isEditable( );
Void setRows(int) Void setColumns( )	Sets number of rows and columns for the text area. Ex. jTextArea.setRows(5);
Void append(string)	Adds specified text in the text area. Ex: jTaxtArea1.append(“How are you”);
Void insert(string, int)	Inserts specified text at given position. Use 0 to insert at top. Ex. jTextArea1.insert(“Amit”,1);
Void setLineWrap(boolean)	Enables or disables line wrap feature.

Working with Password Field

A jPasswordField is a type of Text field that shows encrypted text i.e. actual text is not shown, rather than '*' is displayed.

The character displayed in place of typed character is called echo Character, which is controlled by echoChar property.

Commonly used Properties of jPasswordField

Properties	Value	Description
Background	Color	Sets the background color. It works only when opaque is set to True.
Foreground	Color	Sets the foreground color.
Text	Text	Sets the text to be displayed.
Font	Font name and size	Defines the font and size of text.
enabled	True/False	Determines whether Active or not
editable	True/False	Allow user to edit text, if set to true.
echoChar	Character	Specifies the character to be displayed in place of typed character.

Commonly used Methods of jPasswordField

Methods	Description
Void setEchoChar(char)	Sets the echo character. Ex. jPasswordField1.setEchoChar(‘#’);
Char [] getPassword( )*	Returns the text displayed by the password field. Ex. String pwd= new String (jPasswordField1.getPassword());

* getPassword() method returns a character array, not a string. To store it in a string variable you need to use constructor of string.

Comparing Strings in JAVA

In Java two strings can not be compared directly, by using = operator.

Two methods (1) equals( ) (2) compareTo( ) are used for this purpose.

Equals() returns TRUE/FALSE but compareTo() returns 0 if both are equal otherwise non-zero value is returned.

Ex. if (strname.equals(“Amit”)) {……} else {…..}

Working with jScrollBar control

A jScrollBar control belongs to jScrollBar class of Java. Generally it is used to set the input values to the application like Sound slider, Color contrast or brightness etc. Sometimes it is used to input numerical values to an application, in place of Keyboard.

There are two types of scroll bars i.e. Horizontal and Vertical, depending on its Orientation or appearance on the Frame.

Scroll bars offers a range of numbers starting from it MINIMUM to MAXIMUM values. The current value of indicator can be used in the application, which can be changes by scrolling the indicator.

When user changes the position of indicator, by clicking on Arrows of scroll bar, an Unit Increment is accessed, where as on clicking on scroll area, a Block Increment is occurred.

AdjustmentValueChanged method of Adjustment event is triggered when scrollbar is changed.

Some features of jScrollBar's are-

- It can be used to input some numerical values to the application.
- It may works as a slider to the various Multimedia controls to control the sound and properties of the picture or video.
- It may be Horizontal or Vertical, by setting its Orientation property.

Commonly used Properties of jScrollBar

Properties	Value	Description
enabled	True/False	Determines whether Active or not
minimum	Value	Determines the minimum value for the scrollbar
maximum	Value	Determines the maximum value
Orientation	value	Defines the orientation i.e. Horizontal or Vertical. Default is Vertical.
unitIncrement	value	Defines the amount of change when user clicks on Arrows. Default is 1.
value	value	Returns current value of indicator. Default is 0.

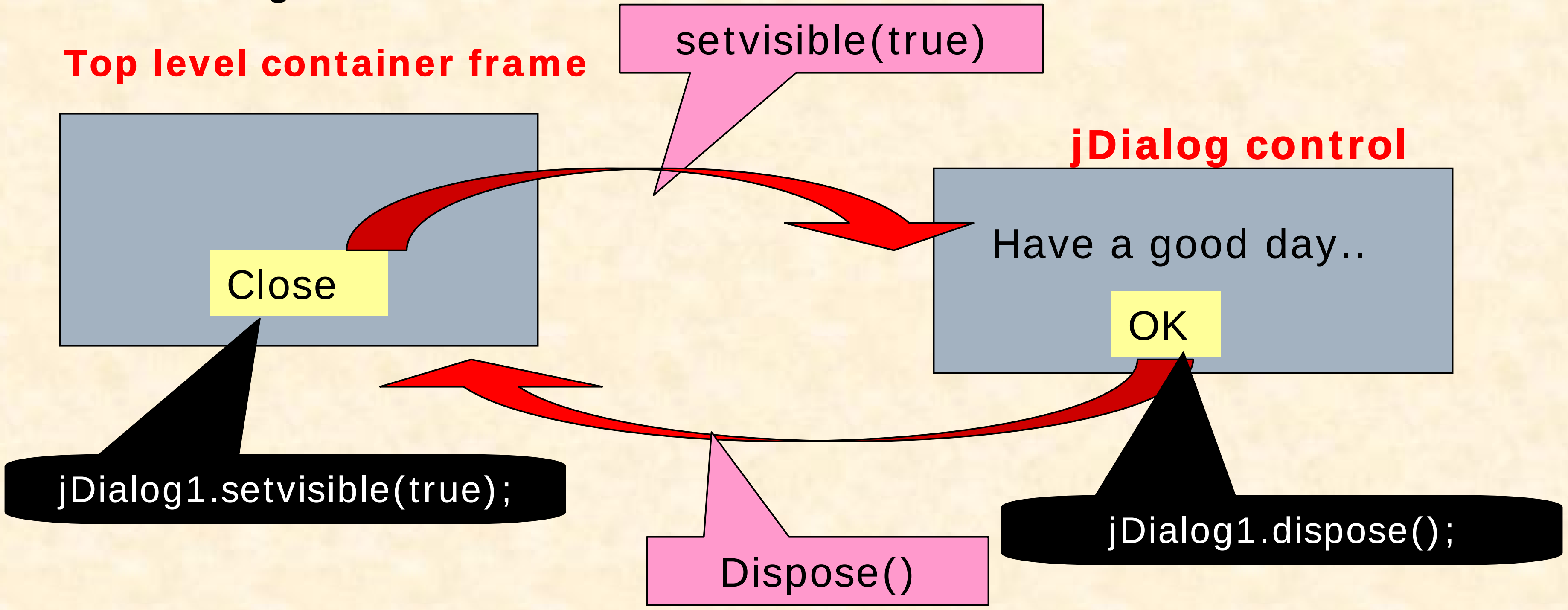
Commonly used Methods of jScrollBar

Methods	Description
Void setMinimum(int)	Sets the minimum value for the scroll bar. Ex. jScrollBar1.setMinimum(0);
Int getMinimum()	Returns the minimum value of the scroll bar. Ex. int x= jScrollBar1.getMinimum();
Void setMaximum(int)	Sets the maximum value for the scroll bar.
Int getMaximum()	Returns the maximum value of the scroll bar.
Void setOrientation(int)	Sets the orientation of the scroll bar. (0-vertical, 1- horizontal) Ex. jScrollBar1.setOrientation(1);
Int getOrientation()	Returns the current orientation value of the scroll bar.
Void setValue(int)	Sets the current value for the scroll bar.
Int getValue()	Returns the current value of the scroll bar. Ex. int x= jScrollBar1.getValue();
Void setEnabled(boolean)	Returns true id scrollbar is enabled.

Working with jDialog Control

A dialog control is a control that can be used to display messages in the application. It may contains message , image and buttons etc.

- Add jDialog control from swing controls and customize it as per you requirement with text, image and buttons.
- It can be invoked in ActionPerformed Event of a button in top level container jframe by jDialog1.setvisible(true) command.
- You can set text at run time by jDialog1.setText() method before invoking it.



Steps to add Dialog Control (jDailog)

A dialog control belongs to Swing Window class available in Swing control's palette. The following step may be followed –

Design and Application using jFrame and other controls, as you did earlier.

- Drag jDialog control from Swing Window tab of Swing Tool box. This will add jDialog1 node under Other component tab in Inspector window.
- Now double click jDialog1 node.. In Inspector window, this will open a blank Dialog frame in Design Area.
- Attach jLabel and jButton controls as per your choice and customize them Text messages.
- Double Click on the jButton control and write the given code in //todo section, to close the dialog box.

```
jDialog1.dispose()
```

- Now double on jFrame node in Inspector window to open jFrame control in design area.
- Double click on the button on which you want to attach Dialog window… and write the command in /TODo section , to invoke the Dialog control.

```
jDialog1.setVisible(true)
```

- Now RUN the application.

Understanding Focus

- A Focus is the ability to receive user input/ response through Mouse or Keyboard. When object or control has focus, it can receive input from user.
- An object or control can receive focus only if its enabled and visible property are set to true.
- Most of the controls provides FOCUS_GAINED() and FOCUS_LOST() method in FocusEvent by the FocusListener. FOCUS_LOST() is generally used for validation of data.
- You can give focus to an object at run time by invoking the requestFocus() method in the code.

Ex. jTextBox2.requestFocus();