Chapter 6:

Advanced GUI Programming -More on Swing Controls

Informatics Practices

Class XI (CBSE Board)

Revised as per CBSE Curriculum 2015

"Open Teaching-Learning Material"

Visit www.ip4you.blogspot.com for more….

Authored By:- Rajesh Kumar Mishra, PGT (Comp.Sc.)
Kendriya Vidyalaya Upper Camp, Dehradun (Uttarakhand)
e-mail : rkmalld@gmail.com

Objective

In this presentation you will learn the following-

- Introduction to GUI & Swing Controls
- Types of Swing Controls
- Working with Container Controls- jFrame & jPanel
- Working with Basic Controls –

  jTextFields, jLabels, jTextArea, jButton, jPasswordField,
- Working with Selection Controls-

  jCheckBox, jRadioButton , jList, jComboBox
- Working with JOptionPane component
- Concept of Class & objects

GUI & Swing Controls

- GUI refers to interface style between application program and users using some graphical components like Label,TextField and Buttons etc. In JAVA, the GUI programming is done through **Swing Controls** which are available as a part of Java Foundation Classes (JFC). JFC includes various ready-to-use libraries to facilitate developer in designing GUI Applications.
- A GUI application in JAVA contains three basic elements.-

  1. Graphical Component (Event Source):

  It is an object that defines a screen element such as Button, Text field, Menus etc. They are source of the Events.

  2. Event:

  An Event (occurrence of an activity) is generated when user does something like mouse click, dragging, pressing a key on the keyboard etc. Events are trapped by the Event Listeners.

  3. Event Handler Method:

  It contains method/functions which is attached to a component and executed in response to an event.

Types of Swing Controls

- **Component Control:**

  A Swing component is a self-contained graphic entity that can be customized and inserted into applications. Example: jLabel, jTextField, jButton etc.

- **Container Control:**

  A container is special type of component that can hold other components. Example: JFrame, jPanel, jDialog etc.

  Container controls are also divided into-

1. Top Level Container:

   These 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

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, Hand, Wait, DefaultCursor etc.	Specifies the type of mouse cursor when mouse moves on the frame.
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.

Working with Panel- jPanel

- A Panel is container that holds other components displayed on the frame. Generally, it is used to group controls visually.
- 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.

Properties	Value	Description
Background	Color	Sets the background color.
Border	No Border, Bevel Border, Etched Border, Line Border, Titled Border etc.	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.

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.
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 jButtons..

- Commonly used methods of JButton.

Method	Description
setText( )	Sets the text displayed on the button. Ex. jButton1.setText(“You Clicked Me”);

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

setEnabled( )	Enables or disables the button. Ex. jButton1.setEnabled(false);
isEnabled( )	Checks whether button is enabled or not. Ex. if(jButton1.isEnabled()) { …..};
setVisible(boolean)	Makes the button visible or invisible. Ex. jButton1.isVisible(false)

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.

Working with jLabel control

A Label control is 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 can display Text or Image or both and supports HTML formatted text.

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.
Icon	Image file to be displayed	Specifies the image file to be displayed.

Methods	**Description**
setText(String)	Sets the string of text to be displayed. Ex. **`jLabel1.setText("I am OK");`**
getText()	Returns the text displayed by the label. Ex. **`String st=jLabel1.getText();`**

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.
  - Click the Import to Project, if you want to import image in your project.
- **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.png”));
    ```

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, delete and select text.
- You can scroll the text, if not fit in visible area.

Properties	**Value**	**Description**
background	Color	Sets the background color.
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.

Commonly used Properties of jTextField

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

jTextField control always stores data in String form. To convert textual data from jTextField in to numeric type, use parse() method.

```
Byte.parseByte( jTextFiled1.getText() )   ⇨ string into byte.
Short.parseShort(jTextFiled1.getText())   ⇨ string into short.
Integer.parseInt( jTextFiled1.getText()) ⇨ string into integer.
Long.parseLong( jTextFiled1.getText())    ⇨ string into long.
Float.parseFloat( jTextFiled1.getText()) ⇨ string into float.
Double.parseDouble(jTextFiled1.getText()) ⇨string into double.
```

Working with jTextArea control

A jTextArea control is a multi-line text component, used to get input from user or to display text.

By default, it does not wrap (move next line) text, if line goes beyond the boundary, until LineWrap property is true.

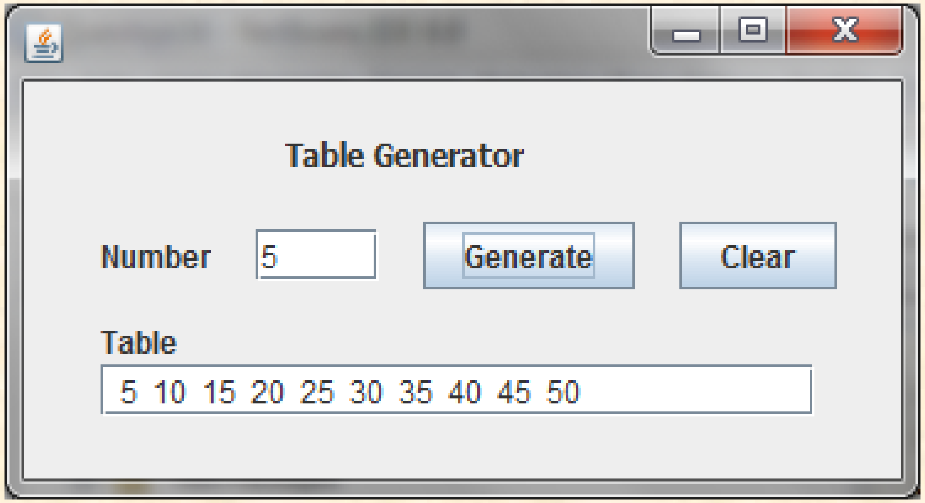
Properties	Value	Description
Background	Color	Sets the background color.
Foreground	Color	Sets the foreground color.
LineWrap	True/ false	Defines Wrapping feature enable/disable
Rows	number	Set the number of rows of in text area
Columns	number	Sets the number of columns
Text	Text	Sets the text to be displayed.
Font	Font and size	Defines the font and size of text.
Editable	True/False	Allow user to edit text, if set to true.

Commonly used Methods of jTextArea

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

Demo Application- Using TextArea

Consider the 'Table Generator' Application, which accepts a number and generates table of given number.
Controls used-
jTextField1- to accept number
jTextArea1- to display table
jButton1 – Button to generate table.



5 10 15 20 25 30 35 40 45 50

```
Private void jButton1ActionPerformed(…..)
{ int n= Integer.parseInt(jTextField1.getText());
   for (int i = 1; i<=10 ; i++ )
    { jTextArea1.append (“ ”+ (n*i) );
    }
 }
Private void jButton2ActionPerformed(…..)
{ jTextArea1.setText(“”);
}
```

Working with jCheckBox control

A CheckBox control is used to accept user's choice in terms of true/false or yes/no options.

Generally, Check Boxes works independently to each other, so that multiple check boxes can be selected at the same time.

Properties	**Value**	**Description**
background	Color	Sets the background color.
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
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

Commonly used Methods of jCheckBox

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

Working with jRadioButton control

A jRadioButton is used to get true/false or yes/no typed choices from the user. Generally, It is used as a grouped control, so that only one can be selected at a time.

By default, jRadioButtons works independently. To make them dependent, Radio Buttons should be kept in a ButtonGroup container control, so that only one button can be selected at a time in a group.

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.
Font	Font 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	Button will be selected, if set to true. (default is false)

Commonly used Methods of jRadioButton

Methods	Description
setText(String)	Sets the string of text to be displayed. Ex. `jRadioButton1.setText(“Science”);`
getText()	Returns the text displayed on the Button. Ex. `String st=jRadioButton1.getText();`
setEnabled()	Sets the Radio Button enables, if true is given. Ex. `jRadioButton1.setEnabled(true);`
isEnabled( )	Returns the state whether radio button is enabled. Ex. `boolean st=jRadioButton1.isEnabled();`
setSelected()	Sets the Radio Button selected, if true is given. Ex. `jRadioButton1.setSelected(true);`
isSelected( )	Returns the state whether Radio Button is selected or not. Ex. `if(jRadioButton1.isSelected())` `{……… }`

Demo–Using RadioButton & CheckBox

The Milton Casting Company has developed an application to calculate the wage of its workers. The following functionalities are expected.

- The Wage rate is Rs.150/- (per day) for male and Rs.130/- for females.
- An additional amount Rs.50/- per day is paid if worker is skilled.
- When Calculate Button is clicked the Total wage amount is calculated and displayed in relevant Text box.
- When Clear Button is clicked, all the text boxes get cleared and Male option is selected.

Important controls used-

RadioButton-

OptMale & OptFemale

CheckBox-

ChkSkill – Check for skilled

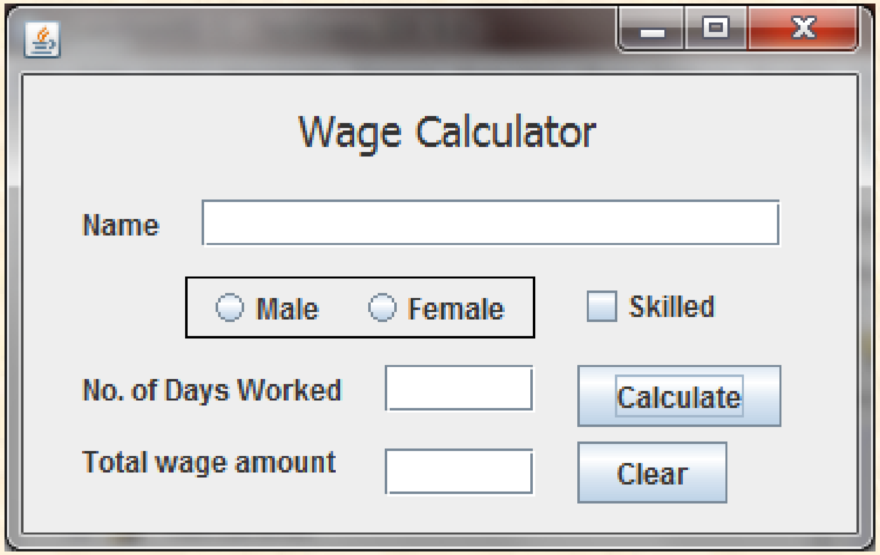
TextFields-

TxtDays – accept no. of days

TxtAmt – display wage amount

Buttons-

BtnCalculate & BtnClear



Demo–Using RadioButton & CheckBox

```
Private void BtnCalculateActionPerformed (…)
{int days, amt, rate;
  days= Integer.parseInt(TxtDays.getText());
  if( OptMale.isSelected() )
      rate=150;
  else
      rate=130;
  if (ChkSkill.isSelected() )
      rate= rate+50;
  amt= days*rate;
  TxtAmt.setText(“”+amt);
}
```

```
Private void BtnClearActionPerformed (…)
{TxtName.setText(“”);
  TxtDays.setText(“”);
  TxtAmt.setText(“”);
  OptMale.setSelected();
}
```

Working with jList control

A List (or List box) is box shaped control containing list of options, 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 in-built scroll bar to view a large list.
- valueChanged() method of ListSelection Listener is used to handle the JList events.
- After attaching a jList control, Model property is used to specify the list of items at design time.

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.)
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
clearSelection()	Clears the selection in the list. Ex. jList1.clearSelection();

getSelectedIndex()	Returns the index of selected items in single selection mode. Returns -1, if selection is not made.
	int x= jList1.getSlectedIndex();
	if (jList1.getSelectedIndex()==1) {…}

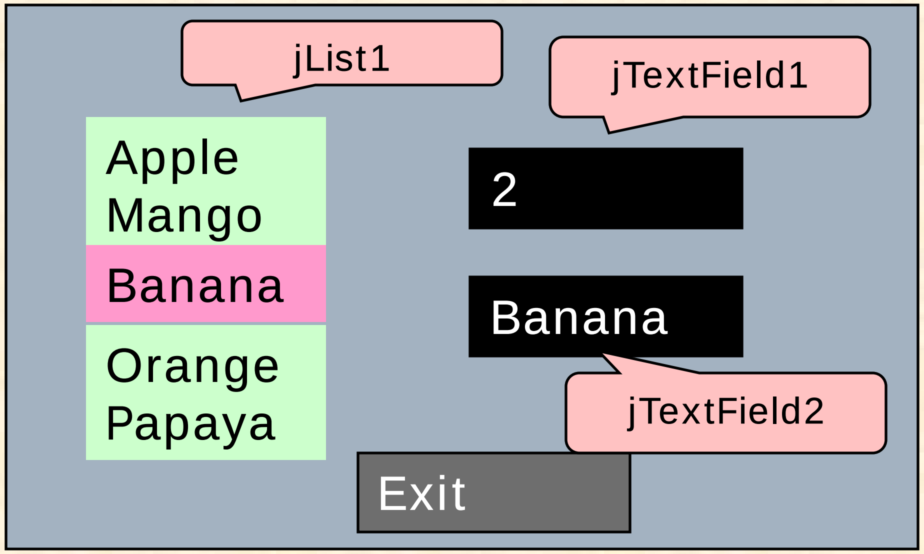
getSelectedValue()	Returns the value or selected items.
	String st= (String) jList1.getSlectedValue();
	if (jList1.getSelectedValue()==“Apple”){…}

isSelectedIndex(int)	Returns True if given Index is selected. Ex: if (jList1.isSlectedIndex(2)) {…}

Other methods like isEnabled(), setVisible(), setEnabled() etc. can be used with jList control.

How to handle Selections in the jList

Suppose we want to display the index and value of selected items i.e. 2 and ‘Banana’ in Text Fields when user selects Banana from 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);
```

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 <u>we can make it editable by setting ‘editable‘ property True</u>.

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
getSelectedIndex()	Returns the index of selected items. `if (jComboBox1.getSlectedItem()==1) {…};`
getSelectedItem()	Returns the selected items. `if (jComboBox1.getSlectedItem()==“Mango”) {…… …};`
isEditable()	Returns True, if Combo box is editable.
addItem(string)	Adds an item to the choice list at the end. Ex. `jComboBox1.addItem(“Banana”);`
getItemCount()	Returns the number of items in the combo Box. `int x = jComboBox1.getItemCount();`
getItemAt(int)	Returns the items at specified index. `String st=jComboBox1.getItemAt(2);`
removeAllItems()	Removes all the items from combo.
removeItemAt(index)	Removes items for given index from the combo. Ex. `jComboBox1.removeItemAt(2);`

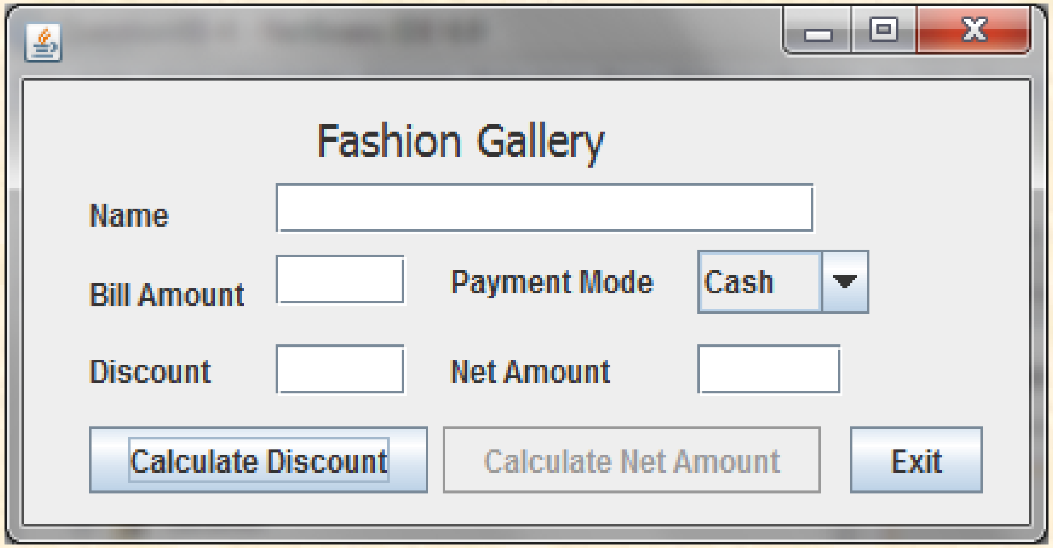
Demo Application- Using ComboBox

Consider the following application for Fashion Gallery- a garments shop, to calculate the discount Amount and net amount. The discount is given on the basis on payment mode as per following rates.

Cash – 10%, Cheque – 8% and Credit – 5% of bill amount.

- If Bill amount is more than 10000 then additional 5% discount is also given.
- Initially, Calculate Net Amount is disabled, but when user click on Calculate Discount button the discount amount is calculated and displayed, and Calculate Net Amount Button is enabled.
- When Calculate Net Amount is clicked the Net Amount is calculated and displayed in Net Amount Text Box.

Controls used-
TextFields-
TxtBill, TxtDis, TxtNet
ComboBox-
CboPayMode
Buttons-
BtnCalDis
BtnCalNet (with enabled property false)
BtnExit



Demo Application- Using ComboBox

```
Private void BtnCalDisActionPerformed (…)
{ float BillAmt, Disc;
  BillAmt= Float.parseFloat(TxtBill.getText());
  if( CboPayMode.getSelectedItem()==“Cash” )
      Disc = BillAmt*10/100;
  if( CboPayMode.getSelectedItem()==“Cheque” )
      Disc = BillAmt*8/100;
 if( CboPayMode.getSelectedItem()==“Credit” )
      Disc = BillAmt*5/100;
  if (BillAmt>=10000)
      Disc = Disc+ (BillAmt*5/100);
  TxtDis.setText(“”+ Disc);
  BtnCalNet.setEnabled(true); }
```

```
Private void BtnCalNetActionPerformed (…)
{ float BillAmt, DiscAmt, NetAmt;
  BillAmt= Float.parseFloat(TxtBillAmt.getText());
  DiscAmt= Float.parseFloat(TxtDis.getText());
  NetAmt= BillAmt - DiscAmt;
  TxtNet.setText(“”+NetAmt); }
```

Working with jPasswordField Control

- A jPasswordField is a type of Text field that shows Encrypted text (actual text is not shown for security purpose) like '*' or any other specified character as you type.
- The character displayed in place of typed character is called echo Character, which is controlled by echoChar property.

Properties	**Value**	**Description**
Background	Color	Sets the background color.
Foreground	Color	Sets the foreground color.
Text	Text	Sets the text to be displayed.
Font	Font 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
setEchoChar(char)	Sets the echo character. `Ex. jPasswordField1.setEchoChar( '#' );`
getPassword( )*	Returns the text displayed by the password field. `String pwd= new String (jPasswordField1.getPassword());`

* getPassword() method returns a character array. 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 equals( ) and 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 (pwd.equals("abc")) {……} else {…..}`

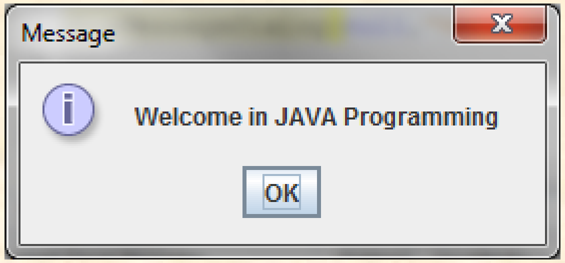
JOptionPane : Built-In Dialog of JAVA

JOptionPane is a ready to use dialog window which allows to display a message or accept input/response from user.

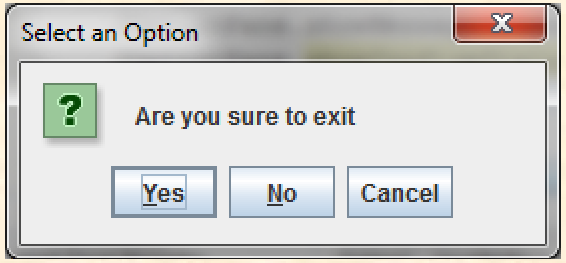
Java provides following types of Dialogs which can be used as per requirement.

Dialog Types	Method	Description
Message Dialog	showMessageDialog()	Used to inform user by displaying a message. It includes OK button only.
Input Dialog	showInputDialog()	Used to get user input using Text Field.
Confirm Dialog	showConfirmDialog()	Used to ask a user to confirm some information with Yes/No or OK/Cancel buttons.

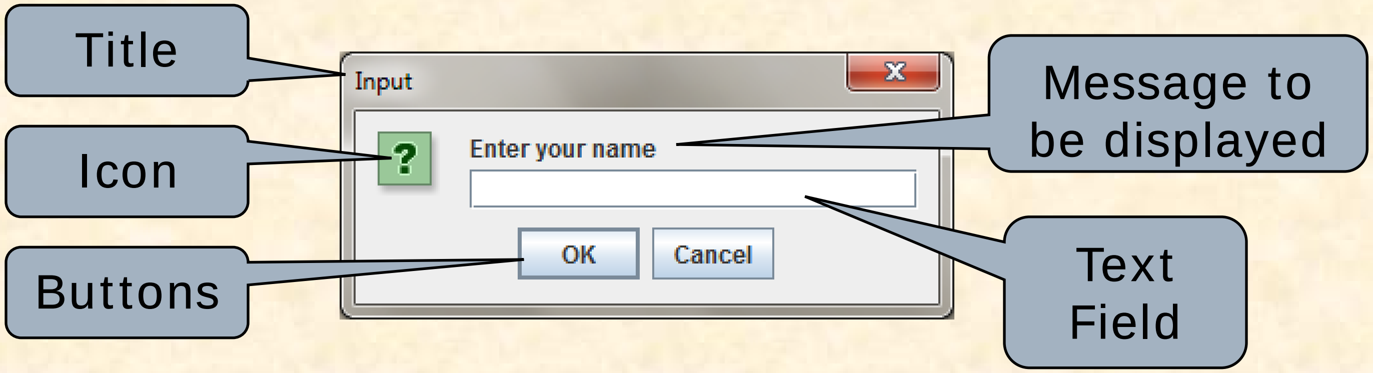
JOptionPane Dialog Types:



Message Dialog



Confirm Dialog



Input Dialog

Working with JOptionPane dialog

To use the JoptionPane dialog control in the application, you must import the following class(s).

import.javax.swing.JOptionPane;

In general, the following syntax of methods along with optional parameters can be used-

JoptionPane.show……([frame name],<“Message”> [,<“Title”>]);

- Frame Name: Generally **null** is used to indicate current frame.
- Message: User given string to covey the message.
- Title: Text to be displayed as Title on the Title bar.

Example(s):

```
JOptinPane.showMessageDialog(null,”JAVA Welcomes You”);
JOptinPane.showMessageDialog (null,”My Name is “+name);
String n= JOptinPane.showInputDialog (null,”Enter your Name ?”);
String n= JOptinPane.showInputDialog (null,”Enter your Name ?”, “Input Name”);
int ch=JOptinPane.showConfirmDialog(null,”Want to exit ?”);
int ch=JOptinPane.showConfirmDialog(null,”Want to exit ?”,”Confirm ?”);
```

Working with JOptionPane dialog

❑Value returned by Input Dialog:

ShowInputDialog() returns a string value which can be directly assigned on a string type variable. You may use parse...() method to convert it into other data types like int or float etc. for further use in the application.

```
String n=JOptionPane.showInputDialog(null, ”Enter Name? ”);
```

```
private void jButton1ActionPerformed(...) {
 // Program to calculate area of circle using Input dialog
String str=JOptionPane.showInputDialog(null, ”Enter radius
of circle ?”);
int radius=Integer.parseInt(str);
float  area=(22/7)*(radius*radius);
JOptionPane.showMessageDialog(null, “Area of circle=“+area);
}
```

Return Value of JOptionPane dialog

❑ Value returned by Confirm Dialog:

Sometimes it is required to check the response of user i.e. which button is pressed, so that application can respond accordingly.

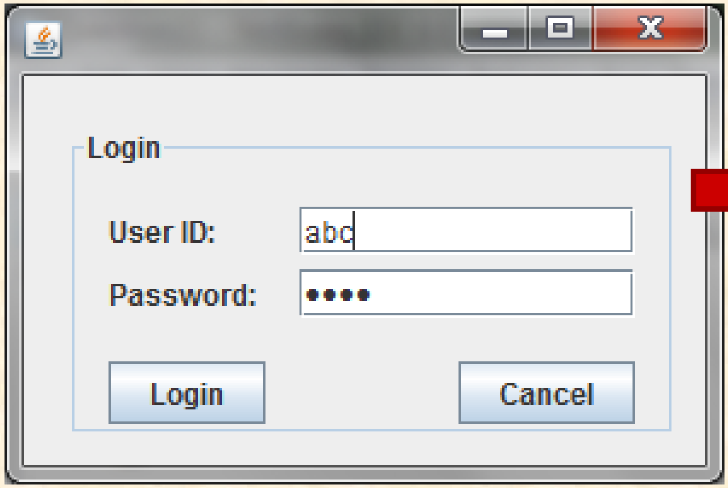
.showConfirmDialog() returns int type value which can be compared with the following constants to determine the status of button pressed by the user.

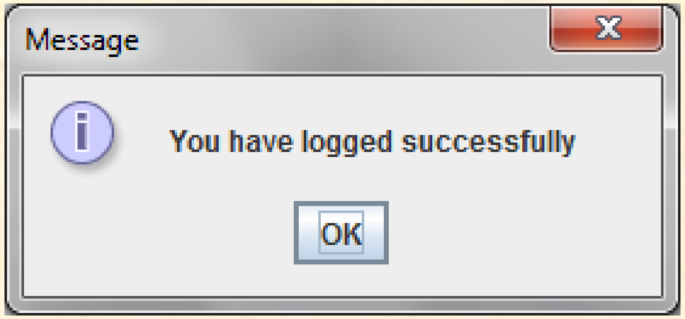
Returned Value	Indication
JOptionPane.YES_OPTION	YES button is pressed
JOptionPane.OK_OPTION	OK button is pressed
JOptionPane.NO_OPTION	NO button is pressed
JOptionPane.CANCEL_OPTION	CANCEL button is pressed
JOptionPane.CLOSE_OPTION	User closed the dialog using X button

```
int ans=JOptionPane.showConfirmDialog(null, ”Want to exit ?”);
If (ans==JOptionPane.YES_OPTION)
 System.exit(0);
```

Demo Application- Using PasswordField

Consider the following Login screen foe an Application. A Message dialog with relevant message appears, as per given valid or invalid password, when Login Button is pressed.





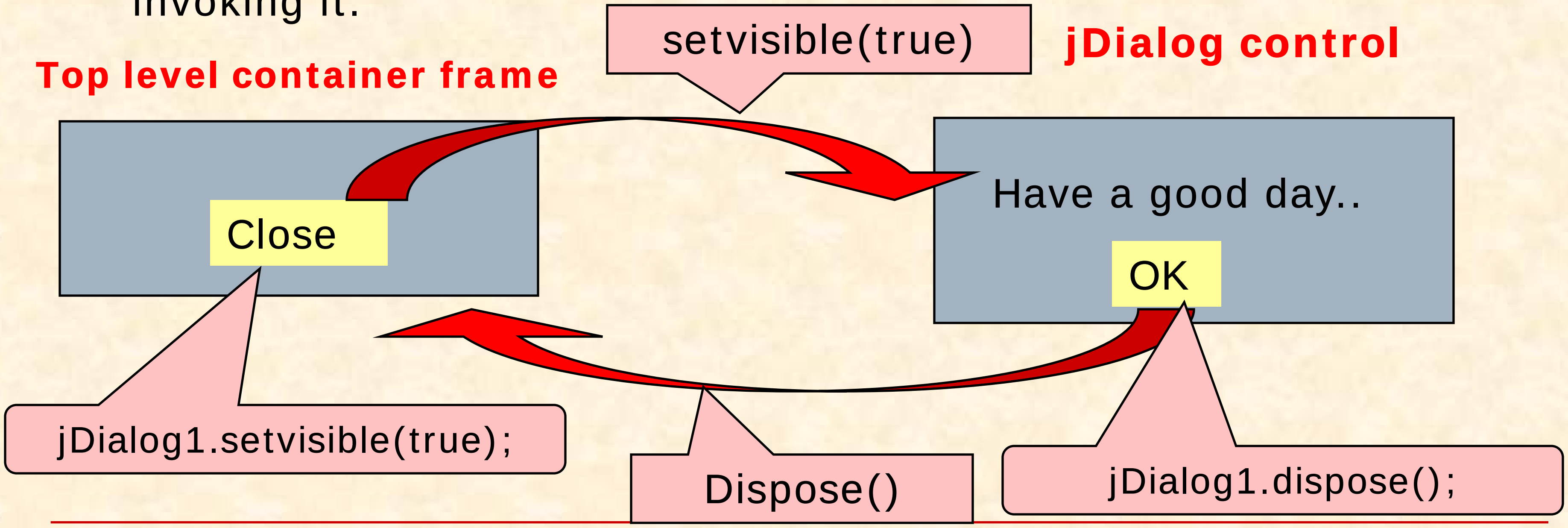
You have logged successfully

```
private void jButton1ActionPerformed(…)
{String pwd= new String (jPasswordField1.getPassword());
 if (pwd.equals(“abc”))
 JOptionPane.showMessageDialog(null,“You have logged sucessfully”);
 else
 JOptionPane.showMessageDialog(null, “Invalid Password”);
}
```

Working with Dialog Control

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

- Add jDialog control from Swing Windows 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.



How to add Dialog Control (jDailog)

- The JDialog is a swing window dialog equipped with Minimise, Maximise and Close functionality.
- JDialog creates a standard pop-up window for displaying desired information related to your application.
- Steps to Attach a JDialog Control-
  1. Design an application as required and drag Dialog control from swing palette and drop it into the application.
  2. To open Dialog frame, double click on jDialog node under other component in Inspector window.
  3. Attach message, button and title in Dialog control as per requirement.
  4. Attach the following ActionPerformed event handler of attached button.

     ```
     jDialog1.dispose();
     ```
  5. Now switch to Frame by double clicking on JFrame node in Inspector window.
  6. Attach the following code on button from where you want to run the dialog.

     ```
     jDialog.setVisible(true);
     ```

Using HTML in Swing Control

We can HTML code in Text Property of various Swing Controls, to make text more decorative by mixed fonts, color and formatting like bold, italic etc.

HTML formatting can be used in Text of Buttons, Tool tips, Labels, 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. e.g.

  <HTML> How are <b><u>You</b></u>

  it will display – How are **You**
- Commonly used HTML tags are

  <U>, <B>, <I>, <P> etc. with closing tags</>.

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() method 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.

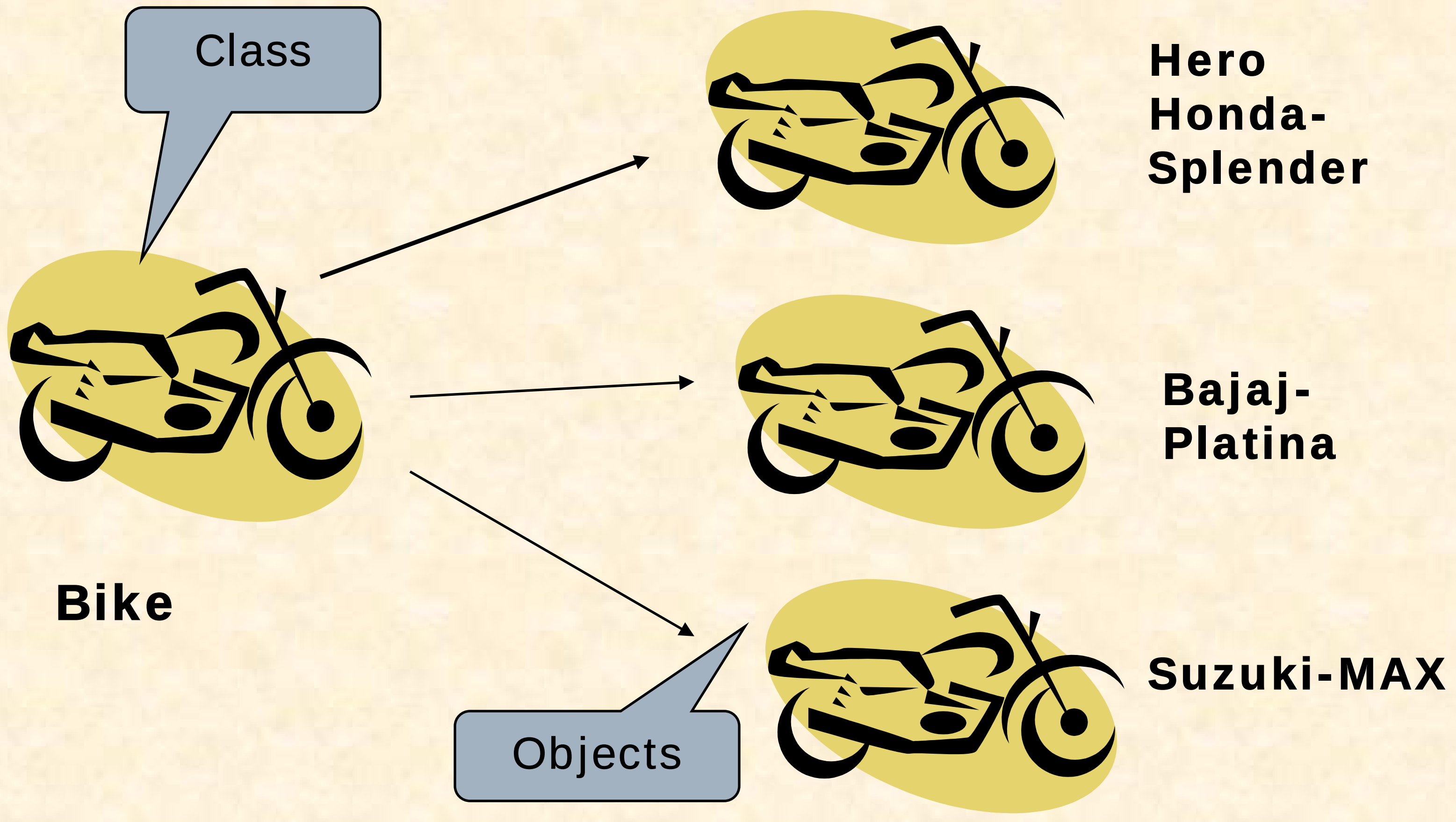
  Ex. `jTextBox2.requestFocus();`

Concept of Class & Object

As you know that JAVA is an Object Oriented Language, in which all the GUI components and programming is based on OOP approach.

- **Object:**
  - An Object is an identifiable entity with some characteristics and behavior.
  - 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 <u>identity</u> i.e. name, <u>characteristics</u> (data values) and <u>Behavior</u> (Methods) which are wrapped in a unit called Objects. E.g. 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 Behavior.
- **Class:**
  - Class is a Blue print of an object that represents a set of similar types of 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. E.g. Student, Man, Bird represents a Class. If <u>You are an Object</u>, then you are the member of <u>Student Class</u>.

Class and Objects in real life



Swing Controls – Class & Object 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, <u>jTextField1 is an object </u>which belongs to Swing control named JTextField and <u>setText() is a Method</u>.

A Method is attached with a object by using (.) and a Method 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.
- Most of them swing controls like jTextField, jButton, JComboBox, jList, jPanel, jRadioButton, jCheckBox, jLabel, jMenuBar, jColorChooser, jFileChooser, jTextField etc. are implemented as a Class Library.