Chapter-8

Creating Frame, Buttons, Checkbox and Radiobuttons

Dulal Acharjee

In GUI programming, the most important features are message box, window, frame, panel etc. which creates an area of displaying some information. On the desktop different areas can be defined for different works as example, some portion for displaying messages(like: error, welcome, opening wait etc. messages); then window defines work area with close, maximize-minimize, resize etc. icons; then frame and panel for drawing area for input/output information. Overall, these features give boundary of display message, work area and designing graphics within a predefined location and area.

In first example shown below, a dialog box or message box is displayed. If a look is given over the program, it will be seen that the class 'JOptionPane' under 'swing' package is imported first. Then, within the main() module of the program 'showMessageDialog(.., ..)' method is used to display the message. The dialog box has close icon symbol as (x) cross sign and another 'OK' button for user interactions. 'JOptionPane' has many methods like:

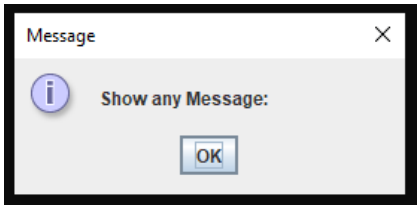
showMessageDialog(argument 1, argument 2);
showInputDialog(argument 1, Object message);
showMessageDialog(Component parentComponent, Object message, String title, int messageType);

showConfirmDialog(parameter 1, Object message);

Creating Message Box/window:

```
//MsgBox.java: to create any message box
import javax.swing.JOptionPane;
public class MsgBox
{
  public static void main(String[] args)
  {
        JOptionPane.showMessageDialog(null, "Show any Message:");
  }
}
```

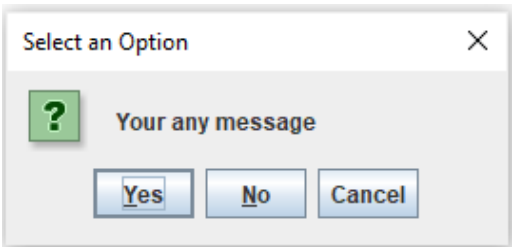
OUTPUT:



Below, another example is shown which uses 'JFrame' class to draw a frame within which message is displayed under the message box. 'showMessageDialog()' method creates message with three input buttons like: 'Yes', 'No' and 'Cancel' as shown in output of the program.

```
//MsgBox2.java: making frame and display message
import javax.swing.*;
public class MsgBox2 {
JFrame obf;
MsgBox2(){
   obf=new JFrame();
   JOptionPane.showConfirmDialog(obf,"Your any message");
}
public static void main(String[] args) {
   new MsgBox2();
}
}
```

OUTPUT:



Discussion: in this program, the changes to the previous programs are that, an object 'obf' is created as:

```
JFrame obf;
```

and the object is used in first parameter position of the method as:

```
JOptionPane.showMessageDialog(obf,"Your any message");
```

In java, **JFrame** class can create different types of frames to display an formatted area for the programmer like window having features as mouse pointer, icons like close, maximise, minimise etc and title of the frame. In programming concept, Jframe class is the window container having all methods, constants to create new window. Two important classes, **JFrame** and **JPanel**, are defined within the package , **javax.swing**. Other most useful classes for GUI designs are: **Toolkit**, **Image**, **Font**, **Point**, **Color**, **Dimension**, **FontMetrics**, and **Graphics** etc. Swing is a package containing classes required for writing stand alone applications or java programs.

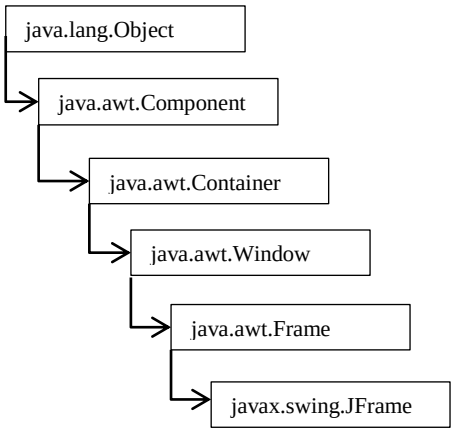


Fig. 8.1. inheritance hierarchy of Frame and JFrame

As shown in above figure 8.1, 'Frame' is a class under 'awt' package but, 'Jframe' is a class under 'swing' package. Both the packages 'awt' and 'swing' are very large container and can be learnt from the website of the product JAVA.

As Example, some components of 'Swing' class are shown in the figure 8.2. It is a very large container and all components are listed in website of the ORACLE /Java and the link is mentioned below:
https://docs.oracle.com/javase/7/docs/api/javax/swing/package-tree.html

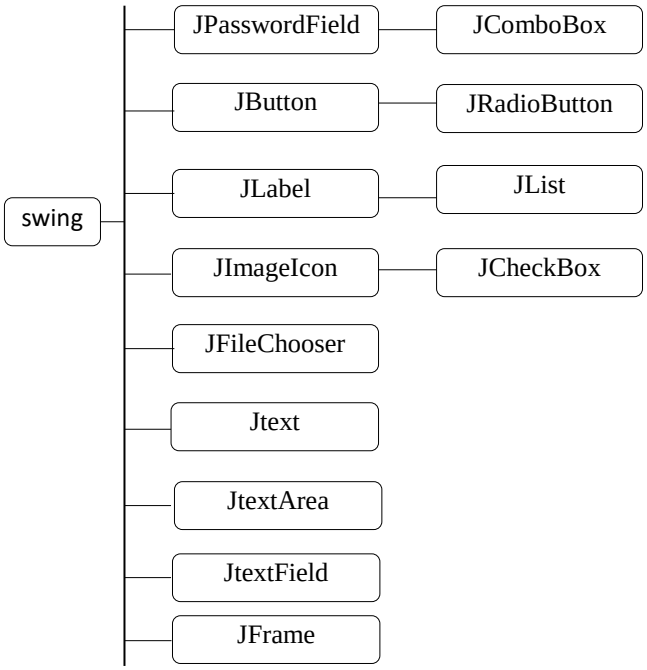


Fig.8.2. list of some components of Swing class required for developing GUI features within program.

JFrame has many methods and these are:

```
void setTitle(String);
String getTitle();
void setSize(int, int);
void setSize(Dimension);
void setResizable(boolean);
void setLocation(int,int);
boolean isResizable();
protected void addImpl(Component comp, Object constraints, int index);
```

A frame has features like resize, move, drag and drop to another location etc. as demonstrated by the next program 'FrameDesign.java'.

There are some constants related to Frame and Window for operating different events on the Frame or Window. As example, after pressing close icon- what would be the status of the program or methods/threads and attached database, color setup etc. these all issues can be managed by these constants. It is the constant related on close or exit button/icon of a window or frame. If the close operation is required to be programmed differently, then JFrame.setDefaultCloseOperation(int) method should be used. Some default close constants are mentioned below:

```
WindowConstants.DO_NOTHING_ON_CLOSE
Frame.EXIT_ON_CLOSE
JFrame.HIDE_ON_CLOSE
WindowConstants.HIDE_ON_CLOSE
WindowConstants.DISPOSE_ON_CLOSE
JFrame.DISPOSE_ON_CLOSE
```

The program shown below will create a frame as designed by JAVA. Default frame methods are used to define size of the frame and title of the frame.

```
//FrameDesign.java: to create a standard frame
import javax.swing.*;
public class FrameDesign
{
	public static void main(String[] args)
	{
		JFrame aFrame = new JFrame("Frame Title..");
		aFrame.setSize(450, 300);
		aFrame.setVisible(true);
	}
}
```

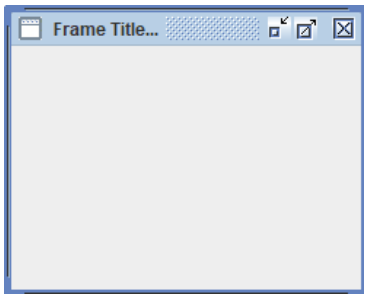
OUTPUT:

Discussions: a simple frame is displayed with three icons of close, maximize and minimize as default. Internally these three buttons are programmed and work properly; if close icon is pressed, window will be closed and so on.

This program will demonstrate decorated frame which will be visible at particular Row-Column position of the monitor area.

```
//FrameDesign2.java: designing decorated frame
import javax.swing.*;
public class FrameDesign2
{
	public static void main(String[] args)
	{
		JFrame.setDefaultLookAndFeelDecorated(true);
		JFrame frm = new JFrame(" Frame Title...");
		frm.setSize(250, 200);
		frm.setVisible(true);
		frm.setLocation(120, 260);//location on screen
	}
}
```

OUTPUT:



Discussions: at first of the program ‘swing’ package is imported that all componets under it can be accessed. This program creates a new class ‘FrameDesign2’ under which main() module is declared. Then a decorating method is invoked from the class ‘Jframe’ as:

```
JFrame.setDefaultLookAndFeelDecorated(true);
```

Effect of this command is visible within the frame as it is beautiful designed as for three control icons, color and dashed line in top header line. Then setSize(), setLocation() etc. methods are used for particular actions.

Home Work: different changes can be practiced over this program as: changing size of the Frame, changing location of the frame, deleting the line of decoration etc.

Creating Buttons within Frame:

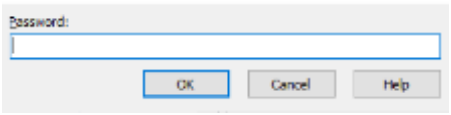
In the above figure of 8.2, hierarchy of components of ‘swing’ package, it is shown that ‘JButton’ is a class defined under ‘swing’. It is used for creating different push buttons as a GUI feature of window. Application of this push button is that if it is selected then it will do some works as per program. Selection of a button may be through keyboard, by touching the screen if the screen is a touch screen, pointing by mouse or by any pointing devices. When a button or a GUI component is selected by mouse or pointing device, three events may happen:
Entering within the area of the button
Clicking the mouse
Releasing the mouse
These are three events and for different events what are the works to be done should be programmed, code to be written for that separately. That is known as event driven programming.

Examples of some buttons:

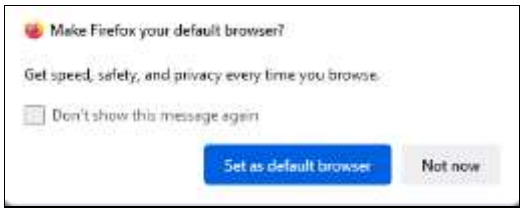
1. Some buttons with icons in MSword:

2. Two buttons in opening of Google Chrome for Cookies policy:

3. Three buttons in opening screen of winSCP(client server file transfer software)



4. Opening screen of Firefox browser with two buttons:




```
//swingDemo.java: demonstration of creating Button using JButton class.
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
public class swingDemo extends JFrame
{
 swingDemo()
 {
    JButton saveBtn = new JButton("Submit");
    JButton cancelBtn = new JButton("Cancel");
    int xpixel = 380;
    int ypixel = 300;
    setLayout(new FlowLayout());
    setSize(xpixel, ypixel); //
    setLocation(160, 200);//location within layout
    add(saveBtn);
    add(cancelBtn);
    setVisible(true);
 }
 public static void main(String[] args)
 {
  new swingDemo();
 }
}
```

OUTPUT:

Discussions: it is an example of JButton class and the hierarchy pattern of this class is shown below in the figure 8.3. It is an example of designing GUI interfaces of creating Push Button only. It is known as action driven or event driven programming. When we select any button, an action/event is called where under that event some code will run as action.

The class JButton has many constructors having different numbers of arguments of different data types and these are:

public JButton(): this constructor has not text as argument, so Button will have no text or name.
public JButton(String text, Icon icon): it has text to show and can display an icon picture.
public JButton(Icon icon): it will create a Button with an Icon picture.
public JButton(Action a): some actions/works can be assigned against this button as data type of class Action where some methods can be defined through the object of the class 'Action'.

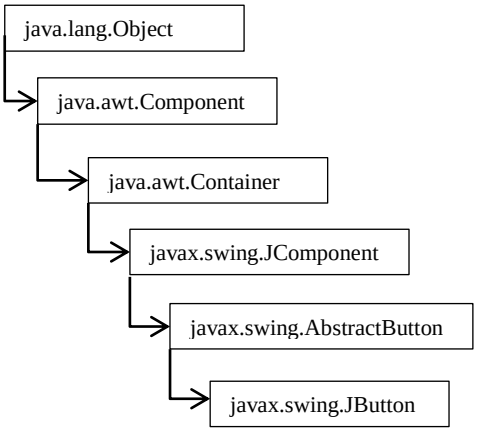


Fig.8.3. hierarchy of JButton class under 'swing' package.

Creating Checkbox:

Checkbox is another GUI feature and multiple options can be selected through this checkbox selections. Any Checkbox has two states 'on' or 'off'; when checked it is 'on' and when unchecked it is considered as 'off' but, these may be reverse states also as per programming logics. Within the program it is written that when checkbox is selected it will do a particular job and for unselect it will do another jobs. In this way, Checkbox controls the events. The methods addActionListener(), addItemListener() and addChangeListener() controls the Listener registering program of selection or unselection of any Checkbox.

Below, a simple program is demonstrated for how to create different checkboxes with a frame, close the frame etc.

```
//FormDesign.java : Demonstration of creating Checkbox
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
public class FormDesign extends JFrame
{
  public FormDesign()
```

```
  {
   JCheckBox checkbx1,checkbx2,checkbx3;
   checkbx1 = new JCheckBox("B.Tech");
   add(checkbx1);
   checkbx2 = new JCheckBox("M.Tech");
   add(checkbx2);
   checkbx3 = new JCheckBox("Ph.D");
   add(checkbx3);
   setLayout(new FlowLayout());
   setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   setSize(200, 100);
   setVisible(true);
  }
  public static void main(String[] args)
  {
   new FormDesign();
  }
}
```

Creating Radiobutton:

It is displayed as a group of buttons and only one can be selected at a time. (But it was dissimilar for checkbox selection; in checkbox selection, at a time any number of boxes can be selected even all number of checboxes can be selected at a time).
Eachtime, a Radiobutton is selected, it generates action events, item events and change events. To control and program these events there are many methods like: actionPerformed( ), getActionCommand( ) and setActionCommand( ).

```
//FormDesign3.java: Demonstration of creating Checkbox and Radiobutton
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
public class FormDesign3 extends JFrame
{
  public FormDesign3()
  {
      JCheckBox c1,c2,c3;
      c1 = new JCheckBox("B.Tech");
      add(c1);
      c2 = new JCheckBox("M.Tech");
      add(c2);
      c3 = new JCheckBox("Ph.D");
      add(c3);
```

```
     setLayout(new FlowLayout());
    //creating three instances radio button
    JRadioButton r1, r2, r3;
    //add radiobutton
    r1= new JRadioButton("Indian Institute of Technology");
    add(r1);
    r2 = new JRadioButton("National Institute of Technology");
    add(r2);
   r3 = new JRadioButton("Central University/State University");
   add(r3);
   setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   setSize(400, 300);
   setVisible(true);
 }
 public static void main(String[] args)
 {
     new FormDesign3();
 }
}
```

OUTPUT:

Discussions: Here, in a same frame, three checkboxes and three radiobuttons are created only. This example will develop knowledge of how to create these types of GUI components, but, actions for different events are not shown in this program which will make the program larger. A derived class or child class is created by name 'FormDesign3', within the body of the class a constructor is created as:

```
public FormDesign3()
{
 ....
 ....
}
```

Here, in this constructor, both checkboxes and radiobuttons are created, and this constructor is invoked from the main() module as:

```
new FormDesign3();
```

By default radiobuttons are displayed at centre of the display area but mentioning location of Row, Column position can be changed also.
In next chapter, more GUI features will be discussed.

Questions and Answers:

Find the Errors:

(a) Below, a portion of a program is shown, find out the errors:

```
DataInputStream din= new DataInputStream(System.in);
                int a;
                String st, age;
                try
                {
                        System.out.print("Input Your Name? ");
                        st = dout.readLine();
                        ……
                        ……
                        …….
                }
```

(b) Portion of code of a program is shown below, find error

```
public static void main(String[] args)
        {
                JFrame aFrame = JFrame("Frame Title..");
                aFrame.setSize(450, 300);
                aFrame.setVisible(true);
        }
```

(c) A portion of code of a program is shown, find the error:

```
public void actionPerformed(ActionEvent)
  {
     String user = txt.getText();
     String code = ptxt.getText();
     if (user.trim().equals("root") && code.trim().equals("root"))
     {
        ms.setText(" Valid User: " + user + "");
      }
```

(d) An error is seeded within main() module, find the error:

```
public class swingDemo extends JFrame
{
 swingDemo()
 {
  ….
  ….
 }
 public static void main(String[] args)
 {
  new swingDemo;
 }
}
```

(e)

```
class BitOr
{
        public static void main(String arg[])
        {
                int x= 14, y= 9;
                System.out.println(" after x OR y: " + (x | y));
                System.out.println (" after x EOR y: " +(x ^ y));
                System.out.println(" after NOT on x:" + (~x));
        }
}
```

Answers:

(a) object name is ‘din’, but, readLine() method is called from ‘dout’ which does not exist, it should be ‘din’.

(b) aFrame is an object of the class JFrame which should be initialized by a operator ‘new’ after the = operator as : JFrame aFrame = new JFrame("Frame Title..");

(c) In the first line, the event catching or handling method actionPerformed has one argument of type object, but, here, only ‘ActionEvent’ class is mentioned but object is not created from this class. The correct answer is: “public void actionPerformed(ActionEvent e)”

(d) Within main() module, the constructor ‘swingDemo’ is invoked, but, parenthesis ‘()’ is absent with constructor name. Corrected answer is : “new swingDemo();”

(e) Outputs of the program are:

after x OR y : 15;

after x EOR y : 7;

after NOT on x : -15

Link of PPT of this book and all source codes:

https://drive.google.com/drive/folders/1bMxMCaqPe0W35COAdn_-BrnV_uzQ-tNO?usp=sharing
(Note: to get access, copy and paste this link to your browser. Both PPT and PDF version of presentation of this book are uploaded in Google Drive.)