**8.3 Frame :-**

A Frame is a top-level window with a title and a border.

The size of the frame includes any area designated for the border. The dimensions of the border area may be obtained using the getInsets method, however, since these dimensions are platform-dependent, a valid insets value cannot be obtained until the frame is made displayable by either calling pack or show. Since the border area is included in the overall size of the frame, the border effectively obscures a portion of the frame, constraining the area available for rendering and/or displaying subcomponents to the rectangle which has an upper-left corner location of (insets.left, insets.top), and has a size of width - (insets.left + insets.right) by height - (insets.top + insets.bottom).

The default layout for a frame is BorderLayout.

A frame may have its native decorations (i.e. Frame and Titlebar) turned off with setUndecorated. This can only be done while the frame is not [displayable](http://docs.oracle.com/javase/1.4.2/docs/api/java/awt/Component.html" \l "isDisplayable%28%29).

In a multi-screen environment, you can create a Frame on a different screen device by constructing the Frame with [Frame(GraphicsConfiguration)](http://docs.oracle.com/javase/1.4.2/docs/api/java/awt/Frame.html#Frame%28java.awt.GraphicsConfiguration%29) or [Frame(String title, GraphicsConfiguration)](http://docs.oracle.com/javase/1.4.2/docs/api/java/awt/Frame.html" \l "Frame%28java.lang.String,%20java.awt.GraphicsConfiguration%29). The GraphicsConfiguration object is one of the GraphicsConfiguration objects of the target screen device.

In a virtual device multi-screen environment in which the desktop area could span multiple physical screen devices, the bounds of all configurations are relative to the virtual-coordinate system. The origin of the virtual-coordinate system is at the upper left-hand corner of the primary physical screen. Depending on the location of the primary screen in the virtual device, negative coordinates are possible, as shown in the following figure.

In such an environment, when calling setLocation, you must pass a virtual coordinate to this method. Similarly, calling getLocationOnScreen on a Frame returns virtual device coordinates. Call the getBounds method of a GraphicsConfiguration to find its origin in the virtual coordinate system.

The following code sets the location of the Frame at (10, 10) relative to the origin of the physical screen of the corresponding GraphicsConfiguration. If the bounds of the GraphicsConfiguration is not taken into account, the Frame location would be set at (10, 10) relative to the virtual-coordinate system and would appear on the primary physical screen, which might be different from the physical screen of the specified GraphicsConfiguration.

Frame f = new Frame(GraphicsConfiguration gc);

Rectangle bounds = gc.getBounds();

f.setLocation(10 + bounds.x, 10 + bounds.y);

Frames are capable of generating the following types of WindowEvents:

* WINDOW\_OPENED
* WINDOW\_CLOSING
* WINDOW\_CLOSED
* WINDOW\_ICONIFIED
* WINDOW\_DEICONIFIED
* WINDOW\_ACTIVATED
* WINDOW\_DEACTIVATED
* WINDOW\_GAINED\_FOCUS
* WINDOW\_LOST\_FOCUS
* WINDOW\_STATE\_CHANGED
* This program shows you how to create a frame in java AWT package. The frame in java works like the main window where your components (controls) are added to develop a application. In the Java AWT, top-level windows are represented by the **Frame** class. Java supports the look and feel and decoration for the frame. For creating java standalone application you must provide GUI to the user.
* The most common method of creating a frame is by using single argument constructor of the **Frame** class that contains the single string argument which is the title of the window or frame. Then you can add user interface by constructing and adding different components to the container one by one.
* In this program we are constructing a label to display "Welcome to Roseindia.net Tutorial." message on the frame. The center alignment of the label has been defined by the **Label.**CENTER. The frame initially invisible, so after creating the frame it need to visualize the frame by setVisible(true) method.
* add(lbl):  
  This method has been used to add the label to the frame. Method add() adds a component to it's container.
* setSize (width, height):  
  This is the method of the Frame class that sets the size of the frame or window. This method takes two arguments width (int), height (int).
* setVisible(boolean):  
  This is also a method of the Frame class sets the visibility of the frame. The frame will be invisible if you pass the boolean value *false* otherwise frame will be visible.

import java.awt.\*;

public class AwtFrame{

public static void main(String[] args){

Frame frm = new Frame("Java AWT Frame");

Label lbl = new Label("Welcome to Roseindia.net Tutorial.",Label.CENTER);

frm.add(lbl);

frm.setSize(400,400);

frm.setVisible(true); } }