UNIT-V
EVENT DRIVEN PROGRAMMING

Graphics programming-Frame-Components-working with 2D shapes-Using color, fonts, and images-Basics of event Handling-event handlers-adapter classes-actions mouse events-AWT event hierarchy-Introduction to Swing-layout management-Swing Components-Text Fields, Text Areas-Buttons-Check Boxes-Radio Buttons-Lists-choices-Scrollbars-windows-Menus-Dialog Boxes and Interfaces, Exception handling, Multithreaded programming, Strings, Input/output

Graphics programming

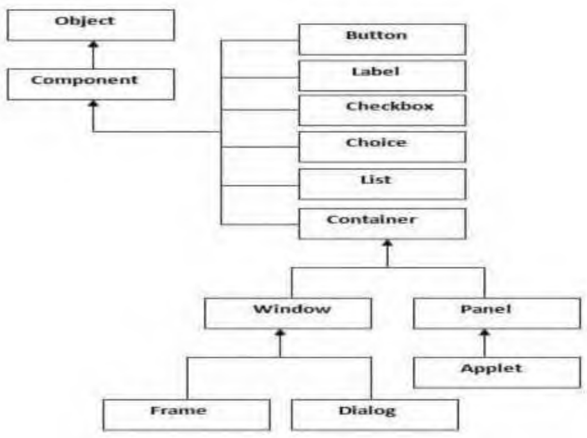
- Java contains support for graphics that enable programmers to visually enhance applications
- Java contains many more sophisticated drawing capabilities as part of the Java 2D API

AWT

- Java AWT (Abstract Window Toolkit) is an API to develop GUI or window-based applications in java.
- Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system.
- AWT is heavyweight i.e. its components are using the resources of OS.The java.awt package provides classes for AWT api such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.

Java AWT Hierarchy

The hierarchy of Java AWT classes are given below.



Container

The Container is a component in AWT that can contain another components like buttons, textfields, labels etc. The classes that extend Container class are known as container such as Frame, Dialog and Panel.

Window

The window is the container that has no borders and menu bars. You must use frame, dialog or another window for creating a window.

Panel

The Panel is the container that doesn't contain title bar and menu bars. It can have other components like button, textfield etc.

Frame

The Frame is the container that contain title bar and can have menu bars. It can have other components like button, textfield etc.

There are two ways to create a Frame. They are,

- By Instantiating Frame class
- By extending Frame class

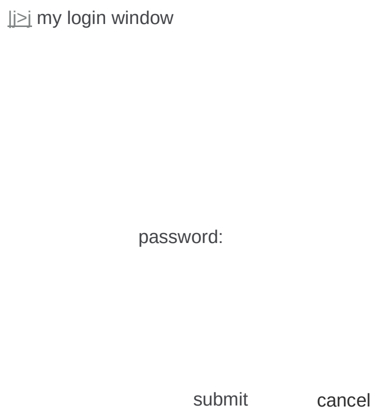
Example:

```
import java.awt.*;

import java.awt.event.*;
class MyLoginWindow extends Frame
{
   TextField name,pass;
   Button b1,b2;
   MyLoginWindow()
   {
      setLayout(new FlowLayout());
      this. setLayout(null);
      Label n=new Label("Name:",Label.CENTER);
      Label p=new Label("password:",Label.CENTER);
      name=new TextField(20);
      pass=new TextField(20);
      pass.setEchoChar('#');
      b1=new Button("submit");
      b2=new Button("cancel");
      this.add(n);
      this.add(name);
      this.add(p);
      this.add(pass);
      this.add(b1);
      this.add(b2);
```

```
        n.setBounds(70,90,90,60);
        p.setBounds(70,130,90,60);
        name.setBounds(200,100,90,20);
        pass.setBounds(200,140,90,20);
        b1.setBounds(100,260,70,40);
        b2.setBounds(180,260,70,40);
    }
    public static void main(String args[]) {
        MyLoginWindow ml=new MyLoginWindow(); ml.setVisible(true);
        ml.setSize(400,400);
        ml.setTitle("my login window");
     }}
```

Output:



|j>j my login window

password:

submit cancel

Event handling:

Changing the state of an object is known as an event. For example, click on button, dragging mouse etc. The java.awt.event package provides many event classes and Listener interfaces for event handling. Event handling has three main components,

- **Events** : An event is a change in state of an object.
- **Events Source** : Event source is an object that generates an event.
- **Listeners** : A listener is an object that listens to the event. A listener gets notified when an event occur

How Events are handled ?

A source generates an Event and send it to one or more listeners registered with the source. Once event is received by the listener, they process the event and then return. Events are supported by a number of Java packages, like **java.util**, **java.awt** and **java.awt.event**.

Important Event Classes and Interface

Event Classes	**Description**	**Listener Interface**
ActionEvent	generated when button is pressed, menu-item is selected, list-item is double clicked	ActionListener
MouseEvent	generated when mouse is dragged, moved,clicked,pressed or released and also when it enters or exit a component	MouseListener
KeyEvent	generated when input is received from keyboard	KeyListener
ItemEvent	generated when check-box or list item is clicked	ItemListener
TextEvent	generated when value of textarea or textfield is changed	TextListener
MouseWheelEvent	generated when mouse wheel is moved	Mou seWheelLi stener
WindowEvent	generated when window is activated, deactivated, deiconified, iconified, opened or closed	WindowListener
ComponentEvent	generated when component is hidden, moved, resized or set visible	ComponentEventLi stener
ContainerEvent	generated when component is added or removed from container	ContainerLi stener
AdjustmentEvent	generated when scroll bar is manipulated	Adj u stmentLi stener
FocusEvent	generated when component gains or loses keyboard focus	FocusListener

Steps to handle events:

- Implement appropriate interface in the class.
- Register the component with the listener.

How to implement Listener

1. Declare an event handler class and specify that the class either implements an ActionListener(any listener) interface or extends a class that implements an ActionListener interface. For example:

```
public class MyClass implements ActionListener

{

// Set of Code

}
```

2. Register an instance of the event handler class as a listener on one or more components. For example:

```
someComponent.addActionListener(instanceOfMyClass);
```

3. Include code that implements the methods in listener interface. For example:

```
public void actionPerformed(ActionEvent e) {
   //code that reacts to the action

}
```