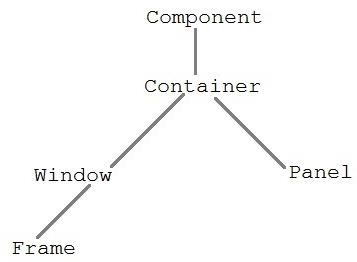
#### Java.awt package

* AWT(Abstract Window Tool kit) is an **API** to develop GUI , window based applications in java.
* AWT components are platform dependent it displays the application according to the view of operating system.
* AWT is a package it will provide very good predefined support to design GUI applications.
* Awt is heavy weight because these components are using operating system resources.
* By using java.awt package we are able to prepare static components to provide the dynamic nature to the component use **java.awt.event** package.**(it is a sub package of java.awt).**

**component :-**The root class of java.awt package is Component class.

**Container:-**It is a component in awt that contains another components like Button,TextField…etc

The classes that extends container classes those classes are containers such as Frame, Dialog and Panel.

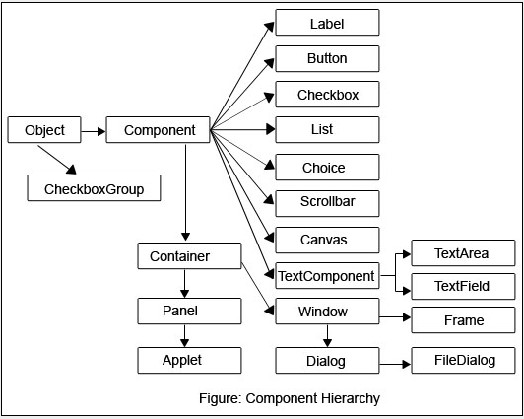


**Window** :- Window class creates a top level window. Window does not have borders and menu bar.

**Panel** : it is a sub class of Container. Panel does not contain title bar, menu bar or border.

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

##### AWT Component Hierarchy:-



##### Java.awt.Frame:-

* Frame is a container it contains other components like title bar, Button, Text Field...etc.
* When we create a Frame class object. Frame will be created automatically with invisible mode, so to provide the visible nature to the frame use setVisible() method of Frame class.

**public void setVisible(boolean b)** where b=true visible mode b=false invisible mode.

* When we created a frame, the frame is created with initial size 0 pixel heights & 0 pixel width hence it is not visible.To provide particular size to the Frame use setSize() method.

**public void setSize(int width,int height)**

* To provide title to the frame use,**public void setTitle(String Title)**
* When we create a frame, the default background color of the Frame is white. If you want to provide particular color to the Frame we have to use the following method.

**public void setBackground(color c)**

**Example-1:- There are two approaches to create Frame in java** 1. By creating object of Frame class.

2. By extending the Frame class.

**Approach 1:-** Creation of Frame by creating Object of Frame class. import java.awt.\*; class Demo

{ public static void main(String[] args)

{ Frame f=new Frame(); f.setVisible(true);

f.setSize(400,400);

f.setBackground(Color.red);

f.setTitle("myframe");

}

};

**Approach 2:-**Taking user defined class by extending Frame class.import java.awt.\*;

class MyFrame extends Frame

{ MyFrame() { setVisible(true); setSize(500,500); setTitle("myframe");

setBackground(Color.green);

}

public static void main(String[] args)

{ MyFrame f=new MyFrame();

}

}

**Example :** Displaying text on the screen

* To display some textual message on the frame override paint() method.

public void paint(Graphics g)

* To set a particular font to the text use Font class present in java.awt package

Font f=new Font(String type,int style,int size);

Font f= new Font("arial",Font.Bold,30); import java.awt.\*;

class MyFrame extends Frame

{ MyFrame() { setVisible(true); setSize(500,500); setTitle("myframe");

setBackground(Color.red);

}

public static void main(String[] args)

{ MyFrame t = new MyFrame();

}

public void paint(Graphics g)

{ Font f=new Font("arial",Font.ITALIC,25);

g.setFont(f);

g.drawString("hi balu how r u",100,100);

}

}

* In above example when we create the MyFrame class object ,jvm will executes MyFrame class constructor just before this JVM will execute Frame class zero argument constructor.
* The Frame class zero argument constructor calling repaint() method & this method will access predefined Frame class paint() method. But as per the requirement overriding paint() method will be executed.

##### Layout Managers:-

When we are trying to add the components into container without using layout manager the components are overriding hence the last added component is visible on the container instead of all. To overcome above problem to arrange the components into container in specific manner use layout manager. **Definitions:-**

The layout managers are used to arrange the components in a Frame in particular manner. **or**

A layout manager is an object that controls the size and the position of components in a container Different layouts in java,

1. java.awt.FlowLayout
2. java.awt.BorderLayout
3. java.awt.GridLayout
4. java.awt.CardLayout
5. java.awt.GridBagLayout

##### java.awt.FlowLayout

The FlowLayout is used to arrange the components into row by row format. Once the first row is filled with components then it is inserted into second row. And it is the default layout of the applet.

**Java.awt.BorderLayout:-**

The BoderLayout is dividing the frame into five areas north,south,east,west,center so we can arrange the components in these five areas.

To represent these five areas borderlayout is providing the fallowing 5-constans

public static final java.lang.String NORTH; public static final java.lang.String SOUTH; public static final java.lang.String EAST; public static final java.lang.String WEST;

public static final java.lang.String CENTER;

import java.awt.\*; class MyFrame extends Frame

{ Button b1,b2,b3,b4,b5;

MyFrame()

{ //this keyword is optional because all methods are current class methods only this.setSize(400,400); this.setVisible(true); this.setTitle("BorderLayout"); this.setLayout(new BorderLayout()); b1=new Button("Boys"); b2=new Button("Girls"); b3=new Button("management"); b4=new Button("Teaching Staff");

b5=new Button("non-teaching staff");

this.add("North",b1); this.add("Center",b2); this.add("South",b3); this.add("East",b4);

this.add("West",b5);

}

}

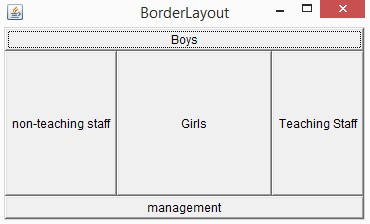
class Demo

{ public static void main(String[] args)

{ MyFrame f=new MyFrame();

}

};



Example 2: project level reduce the length of the code. import java.awt.\*;

class Test

{ public static void main(String[] args)

{ Frame f = new Frame("BorderLayout"); f.setVisible(true);

f.setSize(300,300);

f.setLayout(new BorderLayout());

f.add(new Button("NORTH"),BorderLayout.NORTH);

f.add(new Button("SOUTH"),BorderLayout.SOUTH);

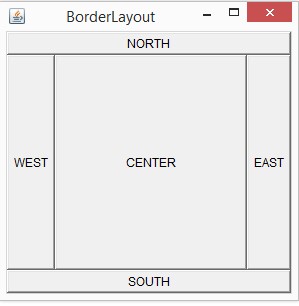
f.add(new Button("EAST"),BorderLayout.EAST);

f.add(new Button("WEST"),BorderLayout.WEST);

f.add(new Button("CENTER"),BorderLayout.CENTER);

}

}



**Preparation of components:-**

**Label: -**Label is a constant text which is displayed along with a TextField or TextArea. **Constructor:** Label l=new Label();

Label l=new Label(“user name”);

**TextField:-**TextField is an editable area&it is possible to provide single line of text.Enter Button doesn’t work on TextField.

To set Text to the textarea : **t.setText(“Sravya”);**

To get the text form TextArea : **String s=t.getText();**

**Constructors:** TextFiled tx=new TextFiled();

TextField tx=new TextField(“balu”);

**TextArea:-**TextArea is a Editable Area&enter button will work on TextArea.

**Constructors:** TextArea t=new TextArea();

TextArea t=new TextArea(int rows,int columns);

TextArea t=new TextArea(String text, int rows,int columns);

* To set Text to the textarea : **ta.setText(“Sravya”);**
* To get the text form TextArea : **String s=ta.getText();**

**Button :-** Used to perform operations by clicking.

**Example :-** import java.awt.\*; class MyFrame

{ MyFrame()

{ Frame f=new Frame(); f.setVisible(true);

f.setTitle("balu");

f.setBackground(Color.red);

f.setSize(400,500);

f.setLayout(new FlowLayout()); **//information about layout check next page**

Label l1=new Label("user name:");

Label l2=new Label("user password:");

TextField tx1 = new TextField(30);

TextField tx2 = new TextField(30);

Button b = new Button("login");

f.add(l1);

f.add(tx1);

f.add(l2);

f.add(tx2);

f.add(b);

}

public static void main(String[] args)

{ MyFrame f = new MyFrame();

}

}

1. **Choice:**-List is allows to select multiple items but choice is allow to select single Item. **Choice ch=new Choice();** 
   * To add items to the choice :**add()**
   * To remove item from the choice based on String :**choice.remove(“HYD”);**
   * To remove the item based on the index position :**choice.remove(2);**
   * To remove the all elements : **ch.removeAll();**
   * To inset the data into the choice based on the position :**choice.insert(2,”balu”);**
   * To get selected item from the choice :**String s=ch.getSelectedItem();**
   * To get the selected item index number : **int a=ch.getSelectedIndex();**

1. **List:**  List is providinglist of options to select.

**CONSTRUCTOR:-**

**List l=new List();**It will creates the list by default size is four elements.

**List l=new List(3);**  It will display the three items size and it is allow selecting the only one.

**List l=new List(5,true);** It will display five items and it is allow selecting the multiple items.

* + To add the elements to the List : **list.add(“c”);**
  + To add the elements to the List at specified index : **list.add(“balu”,0);**
  + To remove element from the List : **list.remove(“c”);**
  + To get selected item from the List : **String x=l.get SelectedItem();**  To get selected items from the List : **String[] x=s.getSelectedItems()**

**Example :-** import java.awt.\*; class Test

{ public static void main(String[] args)

{ Frame f=new Frame();

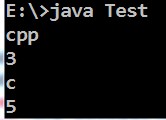
f.setVisible(true); f.setTitle("balu");

f.setBackground(Color.red); f.setSize(400,500);

f.setLayout(new FlowLayout());

**//Choice information**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Choice ch=new Choice(); |  | |  |
|  |  | ch.add("c"); ch.add("cpp"); | ch.add("java"); | |  |
|  |  | ch.add(".net"); ch.remove(".net"); | ch.remove(0); |  |  |
|  |  | ch.insert("balu",2); f.add(ch); |  |  |  |
|  |  | System.out.println(ch.getItem(0)); |  |  |  |
|  |  | System.out.println(ch.getItemCount()); |  |  |  |
|  |  | **//List information** |  |  |  |
|  |  | List l=new List(3,true); |  |  |  |
|  |  | l.add("c"); l.add("cpp"); | l.add("java"); |  | l.add(".net"); |
|  |  | l.add("balu"); l.add("arun",0); | l.remove(0); |  | f.add(l); |
|  |  | System.out.println(l.getItem(0)); |  |  |  |
|  |  | System.out.println(l.getItemCount()); |  |  |  |
| } | } |  |  |  |  |



1. **Checkbox: -** The user can select more than one checkbox at a time.

Checkbox cb2=new Checkbox(“MCA”);

Checkbox cb3=new Checkbox(“BSC”,true);

* + - To set a label to the CheckBox explicitly : **cb.setLabel(“BSC”);**
    - To get the label of the checkbox :**String str=cb.getLabel();**
    - To get state of the CheckBox : **Boolean b=ch.getState();**

1. **RADIO BUTTON:-** 
   * AWT does not provide any predefined support to create Radio Buttons directly.
   * It is possible to create RadioButton by using two classes.

o CheckboxGroup o Checkbox

step 1**:-** Create Checkbox group object. **CheckboxGroup cg=new CheckboxGroup();** step 2**:-** pass Checkboxgroup object to the Checkbox class argument.

Checkbox cb1=new Checkbox(“male”,cg,true);

Checkbox cb2=new Checkbox(“female”,cg,false);

 To get the status of the RadioButton :**String str=Cb.getState();**  To get Label of the RadioButton :**String str=getLabel().**

**Example:-** import java.awt.\*; class Test

{ public static void main(String[] args)

{ Frame f=new Frame();

f.setVisible(true); f.setTitle("balu");

f.setBackground(Color.red); f.setSize(400,500);

f.setLayout(new FlowLayout());

Label l1 = new Label("Qualifications:");

Label l2 = new Label("Gender:");

Checkbox cb1=new Checkbox("SSC",true);

Checkbox cb2=new Checkbox("DEGREE");

Checkbox cb3=new Checkbox("MCA"); Checkbox cb4=new Checkbox("BTECH");

f.add(l1); f.add(cb1); f.add(cb2); f.add(cb3);f.add(cb4);

System.out.println(cb1.getLabel());

System.out.println(cb2.getState());

CheckboxGroup cg=new CheckboxGroup();

Checkbox r1=new Checkbox("male",cg,true);

Checkbox r2=new Checkbox("female",cg,false);

f.add(l2); f.add(r1); f.add(r2);

System.out.println(cb1.getLabel());

System.out.println(cb1.getState());

}

}



##### Event delegation model:-

1. When we create a component the components visible on the screen but it is not possible to perform any action on that component because those components are by default static. **Example:**Whenever we create a Frame it can be minimized and maximized and resized but it is not possible to close the Frame even if we click on Frame close Button. Because frame is a static component so it is not possible to perform actions on the Frame.

1. To make static component into dynamic component we have to add some actions to the Frame. To attach these actions to the Frame component we need event delegation model.

**Event:** Event is nothing but a particular action generated on the particular component.

1. When an event generates on the component the component is unable to respond because component can't listen the event.
2. To make the component listen the event we have to add listeners to the component. Wherever we are adding listeners to the component the component is able to respond based on the generated event.
3. **java.awt.event** package contains listeners and event classes for event handling.
4. The listeners are different from component to component.

**A component delegate event to the listener and listener is designates the event to appropriate method by executing that method only the event is handled. This is called Event Delegation Model.**

**Delegates**

**Delegates**

**Delegates**

**Delegates**

Click the button

**(**

**event is raised**

**)**

component

listeners

Added listener to the

component

Handling method

Handling method

Handling method

**To attach a particular listener to the Frame we have to use following method**

**Public void AddxxxListener(xxxListener e)**

**Where xxx may be ActionListener,windowListener**

**The Appropriate Listener for the Frame is “windowListener” ScrollBar:-**

1. By using ScrollBar we can move the Frame up and down.

ScrollBar s=new ScrollBar(int type)

Type of scrollbar

1. VERTICAL ScrollBar
2. HORIZONTAL ScrollBar

To create a HORIZONTAL ScrollBar**:-**

ScrollBar sb=new ScrollBar(ScrollBar.HORIZONTAL);

To get the current position of the scrollbar we have to use the following method.

public int getValue()

To create a VERTICAL ScrollBar**:-**

ScrollBar sb=new ScrollBar(ScrollBar.VERTICAL);

##### Appropriate Listeners for Components:-

**GUI Component Event Name Listner Name Lisener Methods**

|  |  |  |
| --- | --- | --- |
| 1.Frame | Window Event | Window Listener 1.Public Void WindowOpened(WindowEvent e) |
|  |  | 2.Public Void WindowActivated(WindowEvent e) |
|  |  | 3.Public Void WindowDeactivated(WindowEvent e) |
|  |  | 4.Public Void WindowClosing(WindowEvent e) |
|  |  | 5.Public Void WindowClosed(WindowEvent e) |
|  |  | 6.Public Void WindowIconfield(WindowEvent e) |
|  |  | 7.Public Void WindowDeiconified(WindowEvent e) |
| 2.Textfield | ActionEvent | ActionListener Public Void Actionperformed(ActionEvent ae) |
| 3.TextArea | ActionEvent | ActionListener Public Void Actionperformed(ActionEvent ae) |
| 4.Menu | ActionEvent | ActionListener Public Void Actionperformed(ActionEvent ae) |
| 5.Button | ActionEvent | ActionListener Public Void Actionperformed(ActionEvent ae) |
| 6.Checkbox | ItemEvent | ItemListener Public Void ItemStatechanged(ItemEvent e) |
| 7.Radio | ItemEvent | ItemListener Public Void ItemStatechanged(ItemEvent e) |
| 8.List | ItemEvent | ItemListener Public Void ItemStatechanged(ItemEvent e) |
| 9.Choice | ItemEvent | ItemListener Public Void ItemStatechanged(ItemEvent e) |
| 10.Scrollbar | AdjustmentEvent AdjustmentListener Public Void AdjustementValueChanged | |
|  | (AdjustementEvent e) | |
| 11.Mouse | MouseEvent MouseListener 1.Public Void MouseEntered(MouseEvent e) | |
|  | 2.Public Void MouseExited(MouseEvent e) | |
|  | 3.Public Void MousePressed(MouseEvent e) | |
|  | 4.Public Void MouseReleased(MouseEvent e) | |
|  | 5.Public Void MouseClicked(MouseEvent e) | |
| 12.Keyboard | KeyEvent KeyListener 1.Public Void KeyTyped(KeyEvent e) | |
|  | 2.Public Void KeyPressed(KeyEvent e) | |
|  | 3.Public Void KeyReleased(KeyEvent e) | |

##### Event handling code:-

It is possible to write the event handling code in fallowing ways In different class.

zSame class

By using anonymous inner classes.

**Steps to perform event handling:-** 1) Prepare the required components.

1. Implements the listener interface override the methods to write the even handling code.
2. Add the listener to the component.

##### Applying WindowListener on the Frame :-

The Frame contains WindowLister it contains 7-methods listed below.

windowActivated(WindowEvent e)

Invoked when the Window is set to be the active Window.

windowClosed(WindowEvent e)

Invoked when a window has been closed as the result of calling dispose on the window. windowClosing(WindowEvent e)

Invoked when the user attempts to close the window from the window's system menu.

windowDeactivated(WindowEvent e)

Invoked when a Window is no longer the active Window.

windowDeiconified(WindowEvent e)

Invoked when a window is changed from a minimized to a normal state. windowIconified(WindowEvent e)

Invoked when a window is changed from a normal to a minimized state.

windowOpened(WindowEvent e)

Invoked the first time a window is made visible.

**In below example we are providing even handling code in separate class :-**

###### \*\*\*PROVIDING CLOSING OPTION TO THE FRAME\*\*\*\*

import java.awt.\*; import java.awt.event.\*; class MyFrame extends Frame

{ MyFrame()

{ this.setSize(400,500); **//here this keyword is optional because it is a current class**  this.setVisible(true); this.setTitle("myframe"); this.setBackground(Color.green);

this.addWindowListener(new myclassimpl());

}

}

class myclassimpl implements WindowListener

{ public void windowActivated(WindowEvent e)

{ System.out.println("window activated");

}

public void windowDeactivated(WindowEvent e) { System.out.println("window deactivated");

}

public void windowIconified(WindowEvent e) { System.out.println("window iconified");

}

public void windowDeiconified(WindowEvent e) { System.out.println("window deiconified");

}

public void windowClosed(WindowEvent e) { System.out.println("window closed");

}

public void windowClosing(WindowEvent e) { System.out.println("window closing");

System.exit(0);

}

public void windowOpened(WindowEvent e) { System.out.println("window Opened");

}

};

class Demo

{ public static void main(String[] args)

{ MyFrame f=new MyFrame();

}

};

##### \*\*PROVIDING CLOSEING OPTION TO THE FRAME BY USING WINDOWADAPTOR CLASS \*\*

In above example when our implements the windowListener interface then we must override all (7)the methods of WindowListener interface .but to close the frame we required only one method that is windowCloseing( ) method.

To overcome above limitation use WindowAdaptor class it contains all empty implementations of interface methods.

**Note : if our class implements WindowListener interface we must override all the methods of windowListener interface . But if our class extends windowAdaptor class it is possible to override application required methods**. import java.awt.\*; import java.awt.event.\*; class MyFrame extends Frame

{ MyFrame()

{ this.setVisible(true); this.setSize(500,500);

this.addWindowListener(new Listenerimpl());

}

};

class Listenerimpl extends WindowAdapter

{ public void windowClosing(WindowEvent we)

{ System.exit(0);

}

};

class Demo

{ public static void main(String[] args)

{ MyFrame f=new MyFrame();

}

};

**Writing event handling code in anonymous inner class:-** import java.awt.\*; import java.awt.event.\*; class MyFrame extends Frame

{ MyFrame()

{ this.setVisible(true); this.setSize(500,500);

this.addWindowListener(**new WindowAdapter() { public void windowClosing(WindowEvent we)**

**{ System.exit(0);**

**}**

**}**);

}

}

class Demo

{ public static void main(String[] args)

{ MyFrame f=new MyFrame();

}

};

**Example :- WRITE SOME TEXT INTO THE FRAME**

import java.awt.\*; class MyFrame extends Frame

{ MyFrame()

{ this.setVisible(true); this.setSize(500,500); this.setBackground(Color.red);

this.setTitle("balu");

}

public void paint(Graphics g) { Font f=new Font("arial",Font.BOLD,20);

g.setFont(f);

this.setForeground(Color.green);

g.drawString("HI BTECH ",100,100);

g.drawString("good boys &",200,200);

}

}

class FrameEx

{ public static void main(String[] args)

{ MyFrame f=new MyFrame();

}

};

**Example : CardLayout** import java.awt.\*; class MyFrame extends Frame

{ MyFrame() { this.setSize(400,400); this.setVisible(true); this.setLayout(new CardLayout()); Button b1=new Button("button1");

Button b2=new Button("button2");

Button b3=new Button("button3");

Button b4=new Button("button4");

Button b5=new Button("button5");

this.add("First Card",b1); this.add("Second Card",b2);

this.add("Thrid Card",b3);

this.add("Fourth Card",b4);

this.add("Fifth Card",b5);

}

}

class Demo

{ public static void main(String[] args)

{ MyFrame f=new MyFrame();

}

};

**Example :- GRIDLAYOUT** import java.awt.\*;

class MyFrame extends Frame

{ MyFrame()

{ this.setVisible(true); this.setSize(500,500); this.setTitle("rattaiah"); this.setBackground(Color.red); this.setLayout(new GridLayout(4,4)); for (int i=0;i<10 ;i++ )

{ Button b=new Button(""+i); this.add(b);

}

}

};

class Demo

{ public static void main(String[] args)

{ MyFrame f=new MyFrame();

}

};

**Example :-ACTIONLISTENER**

The below example we are preforming addition and multiplications when we click add & mul buttons. So whenever we clicking button the button is able to listen the even to do this add the Listener to button.

The appropriate listener for button is **ActionListener.**

import java.awt.\*; import java.awt.event.\*;

class MyFrame extends Frame implements ActionListener

{ TextField tx1,tx2,tx3;

Label l1,l2,l3; Button b1,b2; int result;

MyFrame()

{ this.setSize(250,400);

this.setVisible(true);

this.setLayout(new FlowLayout()); l1=new Label("First Value :"); l2=new Label("Second Value :"); l3=new Label("Result :");

tx1=new TextField(25); tx2=new TextField(25); tx3=new TextField(25);

b1=new Button("add"); b2=new Button("mul"); b1.addActionListener(this);**// this represent current class object**

b2.addActionListener(this);

this.add(l1); this.add(tx1); this.add(l2); this.add(tx2); this.add(l3);

this.add(tx3);

this.add(b1); this.add(b2);

}

public void actionPerformed(ActionEvent e)

{ try{

int fval=Integer.parseInt(tx1.getText()); int sval=Integer.parseInt(tx2.getText()); String label=e.getActionCommand(); if (label.equals("add"))

{ result=fval+sval;

}

if (label.equals("mul"))

{ result=fval\*sval;

}

tx3.setText(""+result);

}

catch(Exception ee) { ee.printStackTrace();

}

}

};

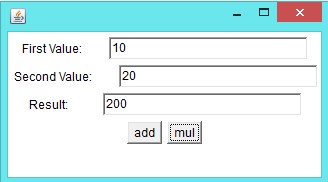
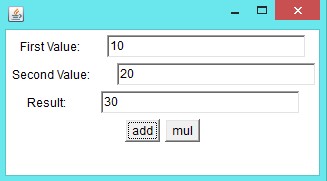
class Demo

{ public static void main(String[] args)

{ MyFrame f=new MyFrame();

}

};



Example :LOGIN STATUS import java.awt.\*; import java.awt.event.\*;

class MyFrame extends Frame implements ActionListener

{ Label l1,l2;

TextField tx1,tx2;

Button b;

String status="";

MyFrame()

{ setVisible(true); setSize(400,400); setTitle("girls"); setBackground(Color.red);

l1=new Label("user name:");

l2=new Label("password:"); tx1=new TextField(25); tx2=new TextField(25);

b=new Button("login");

b.addActionListener(this);

tx2.setEchoChar('\*');

this.setLayout(new FlowLayout());

this.add(l1); this.add(tx1); this.add(l2);

this.add(tx2); this.add(b);

}

public void actionPerformed(ActionEvent ae)

{ String uname=tx1.getText(); String upwd=tx2.getText();

if (uname.equals("Sravya")&&upwd.equals("dss"))

{ status="login success"; }

else

{ status="login failure"; }

repaint();

}

public void paint(Graphics g)

{ Font f=new Font("arial",Font.BOLD,30);

g.setFont(f);

this.setForeground(Color.green);

g.drawString("Status:----"+status,50,300);

}

}

class Demo

{ public static void main(String[] args)

{ MyFrame f=new MyFrame();

}

};

Example :

import java.awt.\*; import java.awt.event.\*;

class MyFrame extends Frame implements ActionListener

{ String label="";

MenuBar mb;

Menu m1,m2,m3;

MenuItem mi1,mi2,mi3;

MyFrame()

{ this.setSize(300,300); this.setVisible(true); this.setTitle("myFrame");

this.setBackground(Color.green);

mb=new MenuBar(); this.setMenuBar(mb);

m1=new Menu("new"); m2=new Menu("option"); m3=new Menu("edit"); mb.add(m1); mb.add(m2); mb.add(m3);

mi1=new MenuItem("open"); mi2=new MenuItem("save");

mi3=new MenuItem("saveas");

mi1.addActionListener(this); mi2.addActionListener(this); mi3.addActionListener(this);

m1.add(mi1); m1.add(mi2); m1.add(mi3);

}

public void actionPerformed(ActionEvent ae)

{ label=ae.getActionCommand(); repaint();

}

public void paint(Graphics g)

{ Font f=new Font("arial",Font.BOLD,25);

g.setFont(f);

g.drawString("Selected item....."+label,50,200);

}

}

class Demo

{ public static void main(String[] args)

{MyFrame f=new MyFrame();

}

};

**Example :- MOUSELISTENER INTERFACE**

import java.awt.\*; import java.awt.event.\*;

class myframe extends Frame implements MouseListener

{ String[] msg=new String[5];

myframe()

{ this.setSize(500,500); this.setVisible(true);

this.addMouseListener(this);

}

public void mouseClicked(MouseEvent e)

{ msg[0]="mouse clicked......("+e.getX()+","+e.getY()+")";

repaint();

}

public void mousePressed(MouseEvent e)

{ msg[1]="mouse pressed......("+e.getX()+","+e.getY()+")";

repaint();

}

public void mouseReleased(MouseEvent e)

{ msg[2]="mouse released......("+e.getX()+","+e.getY()+")";

repaint();

}

public void mouseEntered(MouseEvent e)

{ msg[3]="mouse entered......("+e.getX()+","+e.getY()+")";

repaint();

}

public void mouseExited(MouseEvent e)

{ msg[4]="mouse exited......("+e.getX()+","+e.getY()+")";

repaint();

}

public void paint(Graphics g)

{ int X=50; int Y=100;

for(int i=0;i<msg.length;i++)

{ if (msg[i]!=null)

{ g.drawString(msg[i],X,Y);

Y=Y+50;

}

}

}

};

class Demo

{ public static void main(String[] args)

{ myframe f=new myframe();

}

};

**Example : ITEMLISTENER INTERFACE**

import java.awt.\*; import java.awt.event.\*;

class myframe extends Frame implements ItemListener

{ String qual="",gen="";

Label l1,l2;

CheckboxGroup cg;

Checkbox c1,c2,c3,c4,c5;

Font f; myframe()

{ this.setSize(300,400); this.setVisible(true);

this.setLayout(new FlowLayout());

l1=new Label("Qualification: "); l2=new Label("Gender: ");

c1=new Checkbox("BSC"); c2=new Checkbox("BTECH");

c3=new Checkbox("MCA");

cg=new CheckboxGroup(); c4=new Checkbox("Male",cg,false);

c5=new Checkbox("Female",cg,true);

c1.addItemListener(this); c2.addItemListener(this); c3.addItemListener(this); c4.addItemListener(this); c5.addItemListener(this);

this.add(l1); this.add(c1); this.add(c2); this.add(c3); this.add(l2); this.add(c4);

this.add(c5);

}

public void itemStateChanged(ItemEvent ie)

{ if(c1.getState()==true)

{ qual=qual+c1.getLabel()+",";

}

if(c2.getState()==true) { qual=qual+c2.getLabel()+",";

}

if(c3.getState()==true) { qual=qual+c3.getLabel()+",";

}

if(c4.getState()==true)

{ gen=c4.getLabel();

}

if(c5.getState()==true)

{ gen=c5.getLabel();

}

repaint();

}

public void paint(Graphics g) { Font f=new Font("arial",Font.BOLD,20);

g.setFont(f);

this.setForeground(Color.green);

g.drawString("qualification------>"+qual,50,100);

g.drawString("gender-------------->"+gen,50,150);

qual=""; gen="";

}

}

class Demo

{ public static void main(String[] args)

{ myframe f=new myframe();

}

};

**Example :-KEYLISTENER INTERFACE**

import java.awt.\*; import java.awt.event.\*;

class myframe extends Frame

{ myframe() { this.setSize(400,400); this.setVisible(true); this.setBackground(Color.green);

this.addKeyListener(new keyboardimpl());

}

};

class keyboardimpl implements KeyListener

{ public void keyTyped(KeyEvent e)

{ System.out.println("key typed "+e.getKeyChar());

}

public void keyPressed(KeyEvent e)

{ System.out.println("key pressed "+e.getKeyChar());

}

public void keyReleased(KeyEvent e)

{ System.out.println("key released "+e.getKeyChar());

}

}

class Demo

{ public static void main(String[] args)

{ myframe f=new myframe();

}

};

**Example : CHECK LIST AND CHOICE**

import java.awt.\*; import java.awt.event.\*;

class myframe extends Frame implements ItemListener

{ Label l1,l2;

List l;

Choice ch;

String[] tech; String city=""; myframe()

{ this.setSize(300,400); this.setVisible(true); this.setLayout(new FlowLayout());

l1=new Label("Technologies: "); l2=new Label("City: ");

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | l=new List(3,true); |  | | |
|  | l.add("c"); l.add("c++"); | l.add("java"); | | |
|  | l.addItemListener(this); |  | | |
|  |  |  | | |
|  | ch=new Choice(); |  | | |
|  | ch.add("hyd"); ch.add("chenni"); | ch.add("Banglore"); | | |
|  | ch.addItemListener(this); |  |  |  |
|  | this.add(l1); this.add(l); | this.add(l2); |  | this.add(ch); |
|  | } |  |  |  |
|  | public void itemStateChanged(ItemEvent ie) |  |  |  |
|  | { tech=l.getSelectedItems(); |  |  |  |
|  | city=ch.getSelectedItem(); |  |  |  |
|  | repaint(); |  |  |  |
|  | } |  |  |  |
|  | public void paint(Graphics g) |  |  |  |
|  | { Font f=new Font("arial",Font.BOLD,20); |  |  |  |
|  | g.setFont(f); |  |  |  |
|  | String utech=""; |  |  |  |
|  | for(int i=0;i<tech.length ;i++ ) |  |  |  |
|  | { utech=utech+tech[i]+" "; | } |  |  |
|  | g.drawString("tech:-------"+utech,50,200); |  |  |  |
|  | g.drawString("city---------"+city,50,300); |  |  |  |
|  | utech=""; |  |  |  |
| } | } |  |  |  |
| class Demo  { public static void main(String[] args)  { myframe f=new myframe();  }  }; | |

Example :- AdjustmentListener import java.awt.\*; import java.awt.event.\*;

class myframe extends Frame implements AdjustmentListener

{ Scrollbar sb; int position; myframe()

{ this.setSize(400,400); this.setVisible(true);

this.setLayout(new BorderLayout());

sb=new Scrollbar(Scrollbar.VERTICAL);

this.add("East",sb);

sb.addAdjustmentListener(this);

}

public void adjustmentValueChanged(AdjustmentEvent e)

{ position=sb.getValue();

}

public void paint(Graphics g)

{ g.drawString("position:"+position,100,200);

repaint();

}

}

class scrollbarex

{ public static void main(String[] args)

{ myframe f=new myframe();

}

};

**SWINGS**

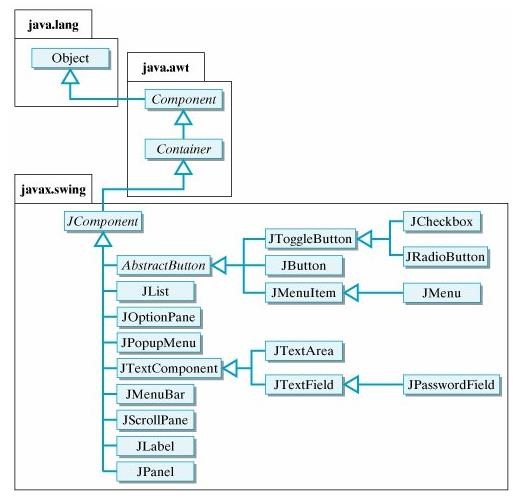
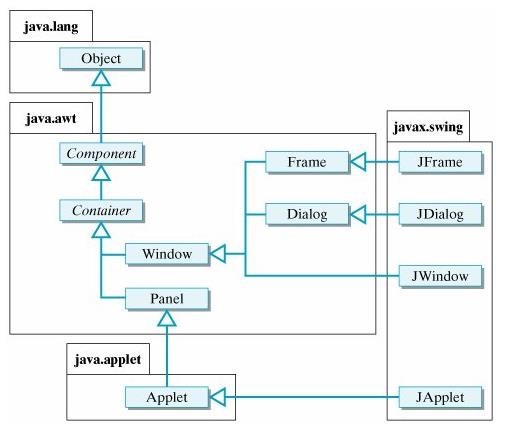
1. Sun Micro Systems introduced AWT to prepare GUI applications but awt components not satisfy the client requirement.
2. An alternative to AWT Netscape Communication Corporation has provided set of GUI components in the form of IFC(Internet Foundation Class) but **IFC** also provide less performance and it is not satisfy the client requirement.
3. In the above context[sun &Netscape] combine and introduced common product to design GUI applications is called JFS(java foundation classes) then it is renamed as swings.

**Differences between awt and Swings:**

* AWT components are platform dependent but Swings components are platform independent because these components are completely written in java .
* AWT components are heavyweight component but swing components are light weight component.
* AWT components consume more number of system resources Swings consume less number of system resources.
* AWT is provided less number of components whereasa swing provides more number of components.
* AWT doesn’t provide Tooltip Test support but swing components have provided Tooltip test support.
* In awt to close the window : windowListenerwindowAdaptor In case of swing use small piece of code.

f.setDefaultCloseOption(JFrame.EXIT-ON-CLOSE);

* In case of AWT we will add the GUI components in the Frame directly but Swing we will add all GUI components to panes to accommodate GUI components.
* The awt classes & interfaces are present in java.awt packages & swing classes are present in javax.swing package.



**Example :**package swingss;

import java.awt.Color;

import java.awt.FlowLayout;

import javax.swing.JButton; import javax.swing.JFrame; import javax.swing.JLabel; public class MyFrame extends JFrame{ public MyFrame() { setVisible(true); setSize(300, 300); setTitle("MyFrame");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); setLayout(new FlowLayout());

JLabel label = new JLabel("Hello World:"); JButton button = new JButton("click me");

add(label); add(button);

}

public static void main(String[] args) {

MyFrame f = new MyFrame();

}

}

**Example 2:-** package swingss; import java.awt.\*; import javax.swing.\*; class Test2 extends JFrame { JLabel l1,l2,l3,l4,l5,l6,l7;

JTextField tf;

JPasswordField pf;

JCheckBox cb1,cb2,cb3;

JRadioButton rb1,rb2;

JList l;

JComboBox cb;

JTextArea ta;

JButton b;

Container c;

Test2()

{ setVisible(true); setSize(150,500); setTitle("SWING GUI COMPONENTS EXAMPLE"); setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); setLayout(new FlowLayout()); getContentPane().setBackground(Color.red); setBackground(Color.green); l1=new JLabel("User Name"); l2= new JLabel("password"); l3= new JLabel("Qualification"); l4= new JLabel("User Gender"); l5= new JLabel("Technologies"); l6= new JLabel("UserAddress"); l7= new JLabel("comments"); tf=new JTextField(15); tf.setToolTipText("TextField"); pf=new JPasswordField(15); pf.setToolTipText("PasswordField"); cb1=new JCheckBox("BSC",false); cb2=new JCheckBox("MCA",false); cb3=new JCheckBox("PHD",false); rb1=new JRadioButton("Male",false); rb2=new JRadioButton("Female",false); ButtonGroup bg=new ButtonGroup(); bg.add(rb1); bg.add(rb2);

String[] listitems={"cpp","c","java"}; l=new JList(listitems);

String[] cbitems={"hyd","pune","bangalore"}; cb=new JComboBox(cbitems); ta=new JTextArea(5,20); b=new JButton("submit");

|  |  |  |  |
| --- | --- | --- | --- |
| add(l1); |  |  |  |
| add(tf); | add(l2); | add(pf); |  |
| add(l3); | add(cb1); | add(cb2); | add(cb3); |
| add(l4); | add(rb1); | add(rb2); | add(l5); |
| add(l); | add(l6); | add(cb); | add(l7); |
| add(ta); | add(b); |  |  |

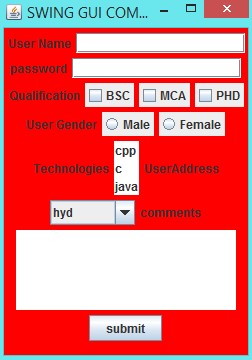
}

public static void main(String[] args)

{ Test2 f=new Test2();

}

};



**Example 3:**

package swingss;

import java.awt.BorderLayout; import java.awt.Color; import java.awt.FlowLayout;

import javax.swing.ImageIcon; import javax.swing.JButton; import javax.swing.JFrame;

import javax.swing.JLabel;

public class MyFrame extends JFrame{ public MyFrame() { setVisible(true); setSize(200, 200); setTitle("MyFrame");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(new BorderLayout());**//setting layout to frame**

JLabel label = new JLabel(new ImageIcon("F:\\corejava images\\download.jpg"));

add(label);

label.setLayout(new FlowLayout());**// setting layout to back ground image**

JLabel label2 = new JLabel("Hello World:");

JButton button = new JButton("click me");

label.add(label2);

label.add(button);

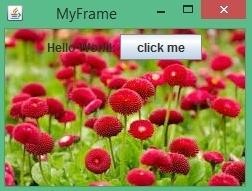
}

public static void main(String[] args) {

MyFrame f = new MyFrame();

}

}



**Example 4:-** package swingss;

import java.awt.FlowLayout; import java.awt.event.ActionEvent; import java.awt.event.ActionListener;

import javax.swing.ButtonGroup; import javax.swing.JButton; import javax.swing.JFrame; import javax.swing.JOptionPane; import javax.swing.JRadioButton;

public class MyFrame2 extends JFrame implements ActionListener {

JRadioButton button1,button2; JButton button; public MyFrame2() { setVisible(true); setTitle("MyFrame"); setSize(300, 300);

setLayout(new FlowLayout());

button1 = new JRadioButton("Male"); button2 = new JRadioButton("Female");

button = new JButton("click");

ButtonGroup group = new ButtonGroup();

group.add( button1); group.add( button2);

add(button1); add(button2); add(button); button.addActionListener(this);

}

@Override

public void actionPerformed(ActionEvent e) { if(button1.isSelected())

{ JOptionPane.showMessageDialog(this, "U r selected Male option thankyou");} if(button2.isSelected())

{ JOptionPane.showMessageDialog(this, "U r selected Female option thankyou");}

}

public static void main(String[] args) {

MyFrame2 frame2 = new MyFrame2();

}

}



**Example 5:**

package swingss;

import javax.swing.JFrame; import javax.swing.JScrollPane; import javax.swing.JTable; public class MyFrame3 extends JFrame { public MyFrame3() { setVisible(true); setSize(400, 200);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

String[][] data={{"101","balu","10000"},{"102","anu","20000"},{"101","durga","30000"}};

String[] col={"Eid","Ename","Esal"}; JTable jTable = new JTable(data,col);

add(jTable);

JScrollPane jScrollPane = new JScrollPane(jTable);

add(jScrollPane);

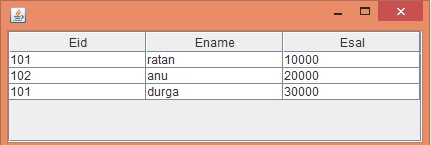
}

public static void main(String[] args)

{ MyFrame3 frame3 = new MyFrame3();

}

}



**Application 6: - JCOLORCHOOSER**

import java.awt.\*; import java.awt.event.\*; import javax.swing.\*; import javax.swing.event.\*;

class MyFrame extends JFrame implements ChangeListener

{ JColorChooser cc;

Container c;

MyFrame()

{ this.setVisible(true); this.setSize(500,500);

this.setTitle("SWING GUI COMPONENTS EXAMPLE"); this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

c=getContentPane();

cc=new JColorChooser();

cc.getSelectionModel().addChangeListener(this);

c.add(cc);

}

public void stateChanged(ChangeEvent c)

{ Color color=cc.getColor(); JFrame f=new JFrame(); f.setSize(400,400);

f.setVisible(true);

f.getContentPane().setBackground(color);

}

}

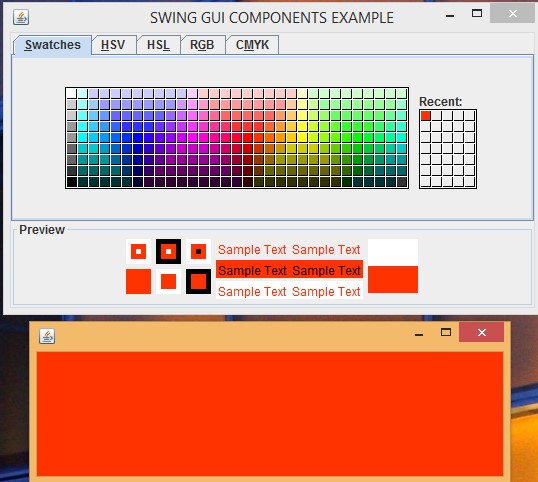
class Demo

{ public static void main(String[] args)

{ MyFrame f=new MyFrame();

}

};



**Application 7: JFILECHOOSER**

import java.io.\*; import java.awt.\*; import java.awt.event.\*; import javax.swing.\*; import javax.swing.event.\*;

class MyFrame extends JFrame implements ActionListener

{ JFileChooser fc;

Container c;

JLabel l;

JTextField tf;

JButton b;

MyFrame()

{ this.setVisible(true); this.setSize(500,500); this.setTitle("SWING GUI COMPONENTS EXAMPLE");

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); c=getContentPane();

l=new JLabel("Select File:");

tf=new JTextField(25); b=new JButton("BROWSE"); this.setLayout(new FlowLayout()); b.addActionListener(this);

c.add(l); c.add(tf); c.add(b);

}

public void actionPerformed(ActionEvent ae)

{ class FileChooserDemo extends JFrame implements ActionListener

{ FileChooserDemo()

{ Container c=getContentPane(); this.setVisible(true); this.setSize(500,500); fc=new JFileChooser(); fc.addActionListener(this);

fc.setLayout(new FlowLayout());

c.add(fc);

}

public void actionPerformed(ActionEvent ae) { File f=fc.getSelectedFile(); String path=f.getAbsolutePath(); tf.setText(path);

this.setVisible(false);

}

}

new FileChooserDemo();

}

}

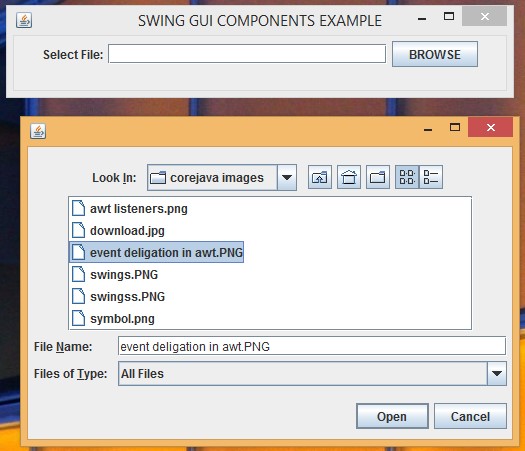
class Demo

{ public static void main(String[] args)

{ MyFrame f=new MyFrame();

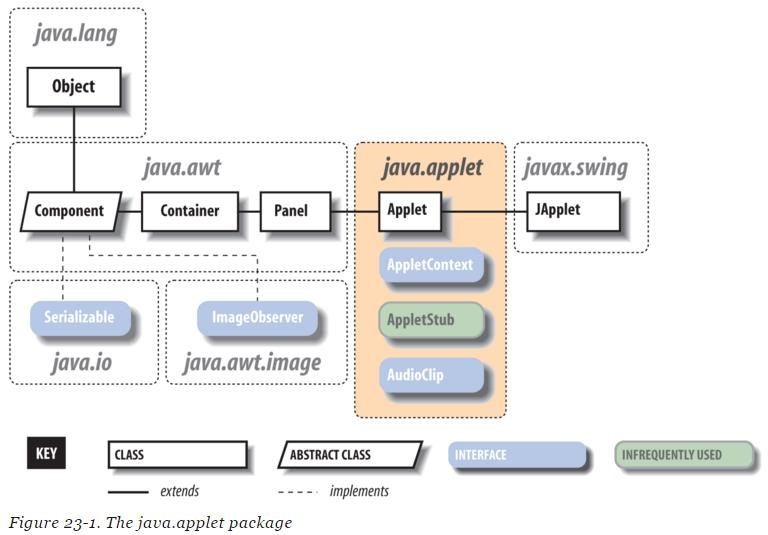
}

};



##### Java.awt.Applet

* The applet is runs on browser window to display the dynamic content on browser window.
* The applet does not contains main methods to start the execution but it contains life cycle methods these methods are automatically called by web browser.
* To run the applet in browser window we need to install plugin in.



The applet contains four life cycle methods

* 1. Init ()
  2. Start()
  3. Stop()
  4. Destroy()

These methods are called by applet viewer or web browser Init() : used to initialize the applet and it called only once.

Start() : it invoked after init() method to start the applet then the applet become visible.

Stop() : it is used to stop the applet then the applet become invisible.

Destroy(): it called after stop method it gives to applet last chance to cleanup.

Java.awt.component class provide one life cycle method of applet

Public void paint(Graphics g)

The above method used to paint the applet to print the data in applet.

**Steps to design the application:-**

1. Create the applet by extending **Applet** class.
2. Then configure the applet in html code.

**Applet First Application:- Test.java:-** import java.applet.Applet; import java.awt.\*; public class Test extends Applet { public void paint(Graphics g) {

g.drawString("Hello, world this is applet first example!", 150, 150);

}

}

**Firstapplet.html**

<html>

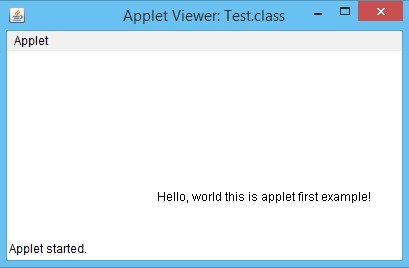
<body>

<applet code="Test.class" height="300" width="300"/>

</body>

</html>

**Execution : appletviewer Firstapplet.html**



**Running applet:-**

It is possible to run the applet in two ways

1. By using html file

Configure the applet in html file then open the html file in browser window. Click on **Firstapplet.html** then the applet is displayed on browser window.

1. By using applet viewer tool

Run the application by using below command

**Appletviewer Firstapplet.html**

**Application 2:** Test.java:- import java.awt.\*; import java.applet.\*;

public class Test extends Applet

{ String msg="";

public void paint(Graphics g)

{ Font f=new Font("arial",Font.BOLD,20);

g.setFont(f);

g.drawString("Durga Software Solutions "+msg,100,200);

}

public void init()

{ msg=msg+"initialization"+" ";

System.out.println("init()");

}

public void start()

{ msg=msg+"starting"+" ";

System.out.println("start()");

}

public void stop()

{ msg=msg+"stoping";

System.out.println("stop()");

}

public void destroyed()

{ msg=msg+"destroyed";

System.out.println("destroy()");

}

};

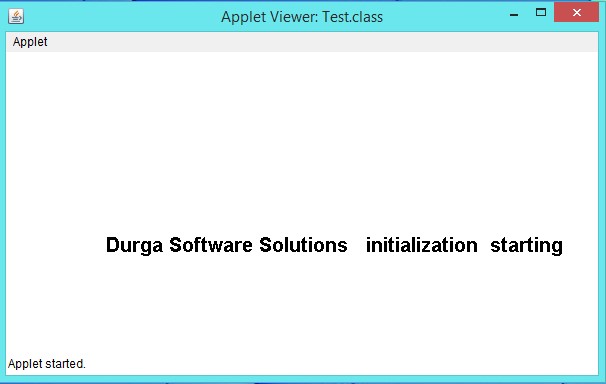
###### Configuration of Applet:- <html>

<applet code="Test.class" width="500" height="500">

</applet>

</html>

**Execution : appletviewer Firstapplet.html**



**Application 3: displaying image on applet.**

**Test.java:**

import java.awt.\*; import java.applet.\*; public class Test extends Applet { Image picture; public void init() {

picture = getImage(getDocumentBase(),"flower.jpg");

}

public void paint(Graphics g) {

g.drawImage(picture, 30,30, this);

}

}

Firstapplet.html

<html>

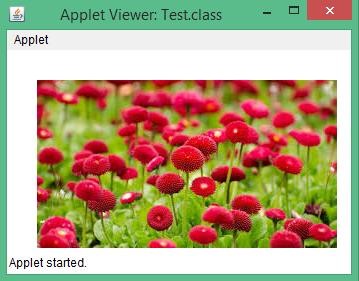
<body>

<applet code="Test.class" height="300" width="300"></applet>

</body>

</html>

**Execution : appletviewer Firstapplet.html**



Different types of methods in java (must know information about all methods)

|  |  |
| --- | --- |
| 1. Instance method 2. Static method 3. Normal method 4. Abstract method 5. Accessor methods 6. Mutator methods 7. Inline methods 8. Call back methods 9. Synchronized methods 10. Non-synchronized methods 11. Overriding method 12. Overridden method | 1. Instance Factory method 2. Static factory method 3. Pattern factory method 4. Template method 5. Default method 6. Public method 7. Private method 8. Protected method 9. Final method 10. Strictfp method 11. Native method |

Different types of classes in java (must know information about all classes)

1. Normal class /concrete class /component class
2. Abstract class /helper class
3. Tightly encapsulated class
4. Public class
5. Default class
6. Adaptor class
7. Final class
8. Strictfp class
9. JavaBean class /DTO(Data Transfer Object) /VO (value Object)/BO(Business Object)

10) Singleton class

1. Child class
2. Parent class
3. Implementation class

Different types of variables in java (must know information about all varaibles)

1. Local variables 6) Protected variables
2. Instance variables 7) Volatile variables
3. Static variables 8) Transient variables
4. Final variables 9) Public variables
5. Private variables

**Java Versions, Features and History:-** Released on 23 January 1996, JDK 1.0 version. **Released on 19 February 1997 JDK 1.1 version.**

* + JDBC  Java Beans&RMI
  + Inner Classes  Reflection (introspection)

**Released on 8 December 1998 J2SE 1.2 version. New features in J2SE 1.2**  Collections framework.

* + Java String memory map for constants.
  + Just In Time (JIT) compiler.
  + Jar Signer for signing Java ARchive (JAR) files.
  + Policy Tool for granting access to system resources.
  + Java Foundation Classes (JFC) which consists of Swing 1.0, Drag and Drop, and Java 2D class libraries.
  + Java Plug-in
  + Scrollable result sets, BLOB, CLOB, batch update, user-defined types in JDBC.  Audio support in Applets.

**Released on 8 May 2000 J2SE 1.3 version.New features in J2SE 1.3**

* + Java Sound
  + Jar Indexing
  + A huge list of enhancements in almost all the java area.

**Released on 6 February 2002 J2SE 1.4 version.New features in J2SE 1.4**

|  |  |
| --- | --- |
| * XML Processing * Java Print Service * Logging API * Java Web Start * JDBC 3.0 API | * Preferences API * Chained Exception * IPv6 Support * Regular Expressions * Image I/O API |

* + Assertions

**Released on 30 September 2004 J2SE 1.5 version.New features in J2SE 1.5**

* + Generics  Varargs
  + Enhanced for Loop  Static Import
  + Autoboxing/Unboxing  Metadata (Annotations)
  + Enum  Instrumentation

**Released on 11 December 2006 J2SE 1.6 version.New features in J2SE 1.6**

* + Scripting Language Support  Pluggable Annotations
  + JDBC 4.0 API  Integrated Web Services.
  + Java Compiler API  Lot more enhancements.

**Released on 28 July 2011 J2SE 1.7 version.New features in J2SE 1.7**

* + Strings in switch Statement  Try with Resources
  + Type Inference for Generic  Java nio Package

Instance Creation  Binary Literals, underscore in

* + Multiple Exception Handling literals

**Released on 18th march 2014 JDK 1.8 version.New features in JDK 1.8**

|  |  |
| --- | --- |
| * Default and Static methods in Interface * Lambda Expressions * Functional interface | * Streams * Method References * Data Time API |

* + Optional