UNIT - V

Applets

There are two types of Java Programs.

1) Standalone application programs
2) Applet programs

Standalone application program is a program that is run on our system using java runtime environment. Applet is a small java program that can be executed by a java compatible web browser or applet viewer. Applet is a graphical user interface used to execute forms.

Applet is a special type of program that is embedded into a webpage to generate the dynamic content. It runs inside the web browser and works at client side. Applet is embedded in a HTML page using the APPLET or OBJECT tag and hosted on a web server.

All Applets are sub-classes (either directly or indirectly) of *java.applet.Applet*class. Applets are not stand-alone programs. Instead, they run within either a web browser or an appletviewer. JDK provides a standard applet viewer tool called appletviewer.

In general, execution of an applet does not begin at main() method. Output on applet window is not performed by *System.out.println()*. Rather it is handled with various AWT(Abstract Window Toolkit) methods, such as *drawString()*.

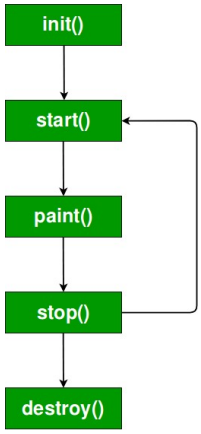
DifferencesbetweenApplicationprogramsandAppletprograms:

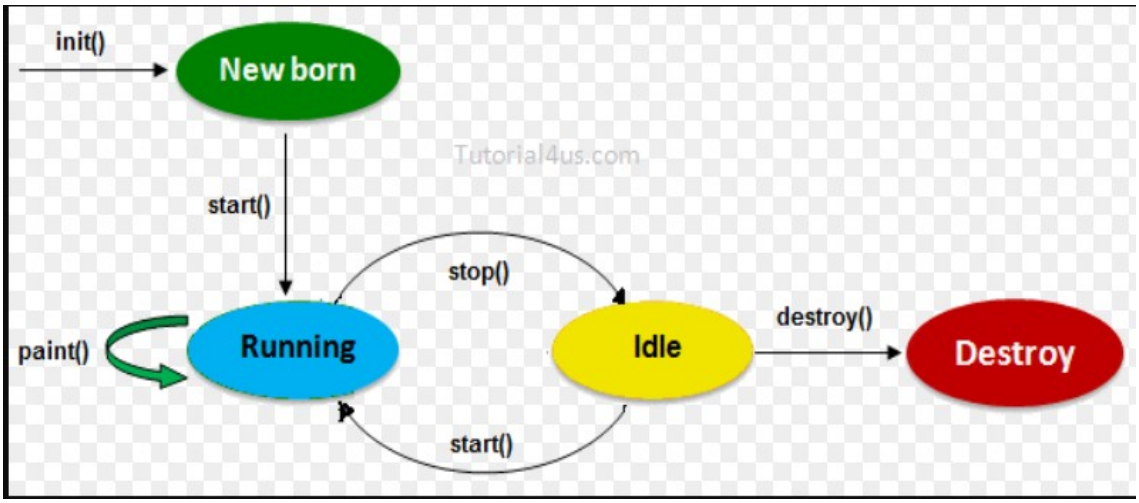
ApplicationPrograms	AppletPrograms
Standaloneapplicationprogramis aprogram thatisrunonoursystemusingjavaruntime environment.	Appletisasmalljavaprogramthatcanbe executedbyajavacompatiblewebbrowser or appletviewer.
Itrequiresamain()methodforits execution.	Itdoesnotrequireamain()methodforits execution.
Created bywritingtheprograminsidethe main() method.	Created byextendingtheAppletclass.
Executedusing JavaRuntimeEnvironment	Executedeitherbyjavacompatibleweb browser or appletviewer tool.

AdvantagesofApplet:

- Appletsareused tomaketheweb sitemoredynamic.
- It works atclient sidesolessresponsetime.
- Secured
- ItcanbeexecutedbybrowsersrunningundermanyplatformsincludingLinux, Windows, MacOs etc.

JavaAppletLifeCycle:





Note:init()anddestroy()methodsareexecutedonlyoncefortheentirelifecycle.

LifecyclemethodsforApplet:

Thejava.applet.Appletclassprovides5lifecyclemethods.

java.applet.Appletclass

For creatingany appletjava.applet.Applet classmust beimported.Itprovides 5life cycle methods of applet.

1. **publicvoidinit():**The**init()**method is the firstmethodtobe calledand isused to initialize the Applet. It is invoked only once.

2. **publicvoidstart():** isinvoked after theinit()method orbrowser ismaximized.It is used to start the Applet. This is running state of Applet.

   Note: **init()**iscalledoncei.e.when thefirsttimeanappletis loaded whereas **start()**is called eachtime anapplet's HTMLdocument isdisplayed onscreen.So, ifauser leaves a web page and comes back, the applet resumes execution at **start( )**.

3. **publicvoidpaint(Graphicsg):** isusedtopaintthe Applet.ItprovidesGraphics class object that can be used for drawing oval, rectangle, arc etc.
   The**paint()**methodiscalledeachtimeanAWT-basedapplet'soutputmustbe redrawn.**paint()**is also called when the applet begins execution. Whatever the cause,whenever the applet must redraw its output, paint ()is called .The **paint()** method hasoneparameter oftypeGraphics.Thisparameter willcontain the graphics context, which describes the graphics environment in which the appletis running.

   Note:Graphics classispresentinjava.awtpackage.Soimportjava.awt.*;

4. **publicvoidstop():** isused tostop theApplet.Itisinvoked whenAppletisstopor browser is minimized.The**stop( )**method is called when a web browser leaves the HTML document containing the applet.
5. **publicvoiddestroy():**is usedtodestroytheApplet.Itisinvokedonly once.The**destroy( )**methodis calledwhentheenvironmentdeterminesthatyourapplet needs to be removed completely from memory.

Appletisa**predefinedclass** and ithasthefollowing lifecyclemethods Class

```
Applet{
        publicvoid
        init(){ showStatus("appletinitialized");
        }
        publicvoid
        start(){ showStatus("appletstarted");
        }

    publicvoid
stop(){ showStatus("appletstopped")
;
     }
    publicvoid destroy(){
            showStatus("appletdestroyed");
     }
    publicvoidpaint(Graphicsg){
      //nullimplementation;
     }
  }
```

Therearetwo standardwaysin whichyou canrun an applet:

1. Executingthe appletwithin aJava-compatible webbrowser.
2. Using applet viewer, such as the standard tool, appletviewer. An applet viewer xecutesyourappletinawindow.Thisisgenerallythefastestandeasiestwaytotest your applet.

Therearetwowaystorun anappletthroughappletviewer

(i) By appletviewertool (alongwithjavaprogram).

(ii) Byhtml file (java program+htmlfile)

(i) Byappletviewertool Program 1

```
//AppletEx2.java

importjava.applet.Applet;
importjava.awt.Graphics;

    /*<appletcode="AppletEx2.class"width="500"height="500">

</applet>*/

publicclassAppletEx2extends Applet{

        //Overriding paint() method public

void

paint(Graphicsg){ g.drawString("Thisismyfirstapplet",1

00,150);

   }

}
```

Note: Commentscanbewrittenanywhereintheprogram.Butbestpracticeis towrite before class

CompileProgram:javacAppletEx2.java

ExecuteProgram:appletviewerAppletEx2.java

(xii) Byhtml file(javaprogram+htmlfile separately)

Program 2

```
//AppletEx1.java

import java.applet.Applet;
importjava.awt.Graphics;

publicclassAppletEx1extendsApplet {

          //Overridingpaint()method

    publicvoid paint(Graphicsg) {
```

```
                g.drawString("Thisismyfirst applet",100,150);
    }
}
```

Compileprogram:javacAppletEx1.java

//myapplet.html

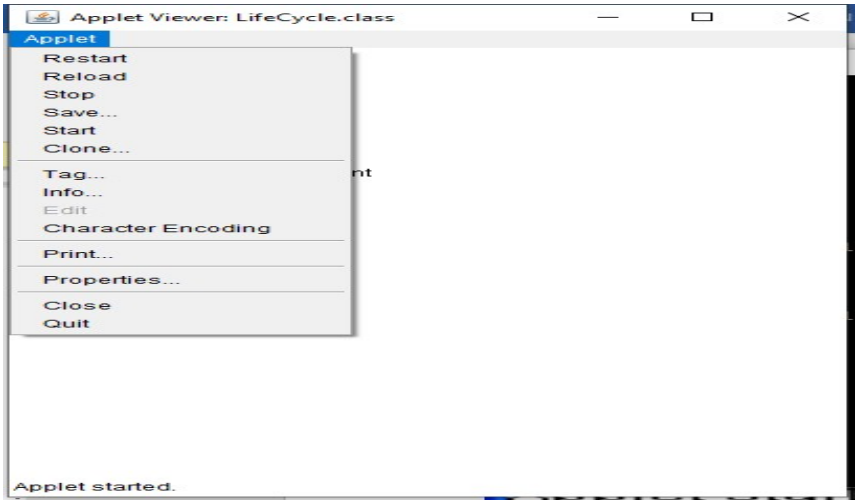
```
<html>
<body>
<appletcode="AppletEx1.class"width="300"height="300">
</applet>
</body>
</html>
```

Executeprogram:appletviewermyapplet.html

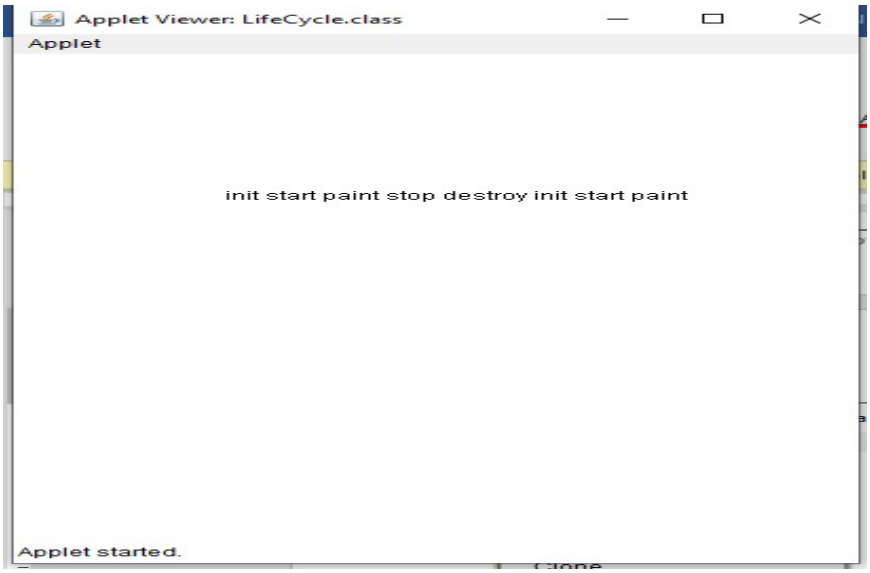
Program3Appletprogramtodemonstrateappletlifecyclemethods

```
import java.awt.*;

importjava.applet.*;

/*<appletcode="LifeCycle.class"height=500width=400>

</applet>*/

public

    classLifeCycleextendsApplet{ Stri

    ngS="";
    //OverridingallmethodsofAppletlifeCycle
    public
    voidinit(){ S=S+"
    init";
    }
    public
    voidstart(){ S=S+"start;
    }
    public
    voidpaint(Graphicsg){ S=S+"pai
    nt "; g.drawString(S,100,150);
    }
    public
    voidstop(){ S=S+"
    stop";
    }
public
voiddestroy(){ S=S+"destroy";
}

}
```

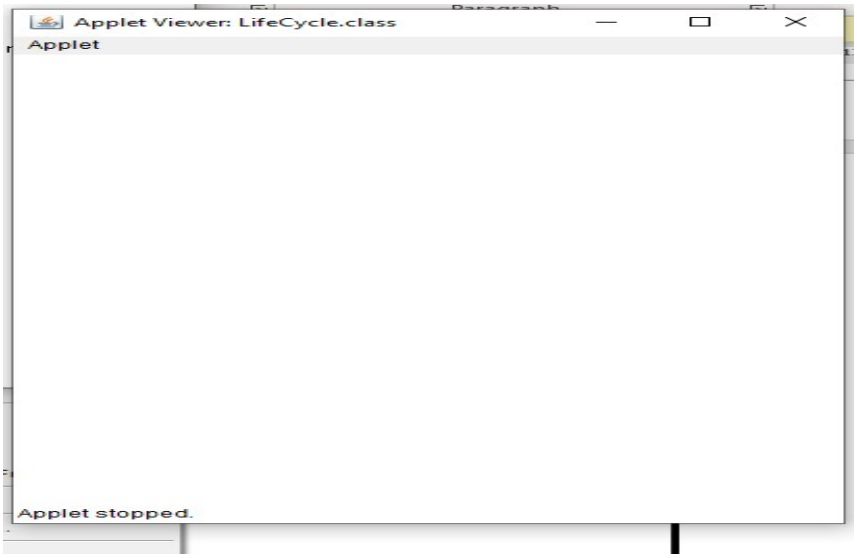
Note:Click on Applet-on-Appletwindow



Note:Afterclickingon Restart weget thefollowingoutput



Note:Afterclickingon Stopweget thefollowingoutput.Theapplet entersidlestate.



Note:Toenterintotherunning state fromthe idle stateclick onthe start thenthe following applet window is created.

Graphicsclassmethods:-

Graphicsis apredefinedclasswhichis availableinawtpackage.Methodsofgraphics are

1) **void**drawString(stringstr,intx,inty)

2) **void**drawLine(intstart_x,intstart_y,intend_x,intend_y)

3) **void**drawRect(intstart_x,intstart_y,intwidth,intheight)

4) **void**fillRect(intstart_x,intstart_y,int width,intheight)

5) **void**drawRoundRect(int start_x,int start_y,int width,int height ,int x_Diameter,inty_ Diameter)

6) **void**fillRoundRect(int start_x,int start_y,int width,intheight ,intx_Diameter,int y_ Diameter)

7) **void**drawOval(intstart_x,int start_y,intwidth,intheight)

8) **void**fillOval(intstart_x,intstart_y,intwidth,intheight)

9) **void**drawArc(intstart_x,int start_y,intwidth,intheight,intstartAngle,intsweepAngle)

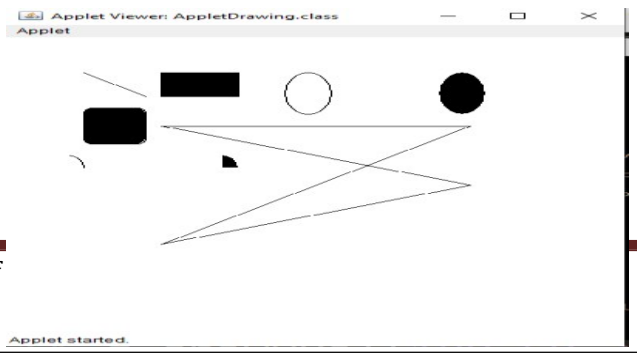
9) **void**fillArc(intstart_x,intstart_y,intwidth,intheight,intstartAngle,intsweepAngle)

10) **void**drawPolygon(intx[],inty[],intno_of_points)

Note: In drawPolygon, x and y are integer arrays which hold (x,y) points. The first point and the last point should be same. For example todraw a square we need four points but whileusingdrawPolygonmethod werequire5points.Asthestartingpointand endingpoint are same.

Program4:

```
importjava.awt.*;
import java.applet.*;

/*<appletcode="AppletDrawing.class"height=500width=400>

</applet>*/

public

   classAppletDrawingextendsApplet{ pub

   lic voidpaint(Graphicsg){

       g.drawLine(50,60,90,100);
       g.drawRect(100,60,50,40);
       g.fillRect(100,60,50,40);
       g.drawRoundRect(50,120,40,60,10,20);
       g.fillRoundRect(50,120,40,60,10,20);
       g.drawOval(180,60,30,70);
       g.fillOval(280,60,30,70);
       g.drawArc(30,200,20,40,0,90);
       g.fillArc(130,200,20,40,0,90);

       intx[]={100,300,100,300,100};
       inty[]={150,150,350,250,150};

       g.drawPolygon(x,y,5);

       }
   }
```



Passingparameters toapplets:

WecangetanyinformationfromtheHTMLfileasaparameter.Forthispurpose,Applet class provides a method named getParameter().

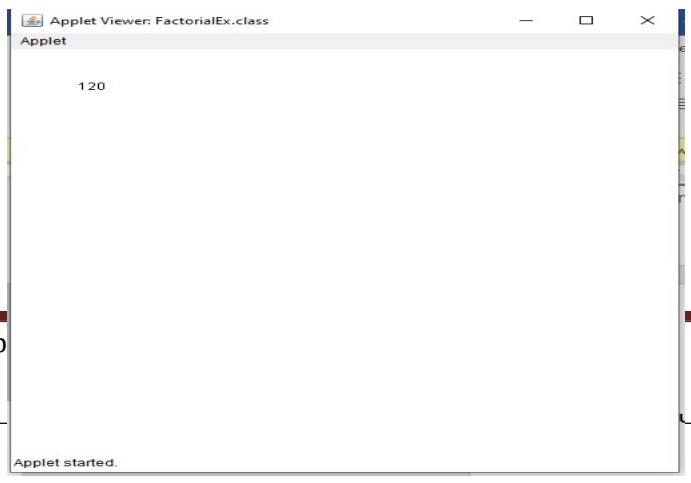
Syntax: publicStringgetParameter(StringparameterName)

Program5:

```
//PassingparameterstoApplets

importjava.applet.Applet;
importjava.awt.Graphics;

/*<appletcode="AppletParam.class"width="300"height="300">
<paramname="msg"value="Welcome to applet">
</applet>  */

public

classAppletParamextendsApplet{ public

voidpaint(Graphicsg){ Stringstr=getPara

meter("msg"); g.drawString(str,50, 50);

}

}
```

Program6:

```
//PassingparameterstoApplets (FactorialProgram)

importjava.applet.*;
importjava.awt.*;
/*<appletcode="FactorialEx6.class"width="500"height="500">
<param name="x"value=5></param>
</applet>        */
public classFactorialEx6extends Applet{

    public voidpaint(Graphicsg){

        Stringstr=getParameter("x");
        int y=Integer.parseInt(str);
        int fact=1;
        for(inti=1;i<=y;i++)
        fact=fact*i; g.drawString(fact+"",50,50);
}

}
```

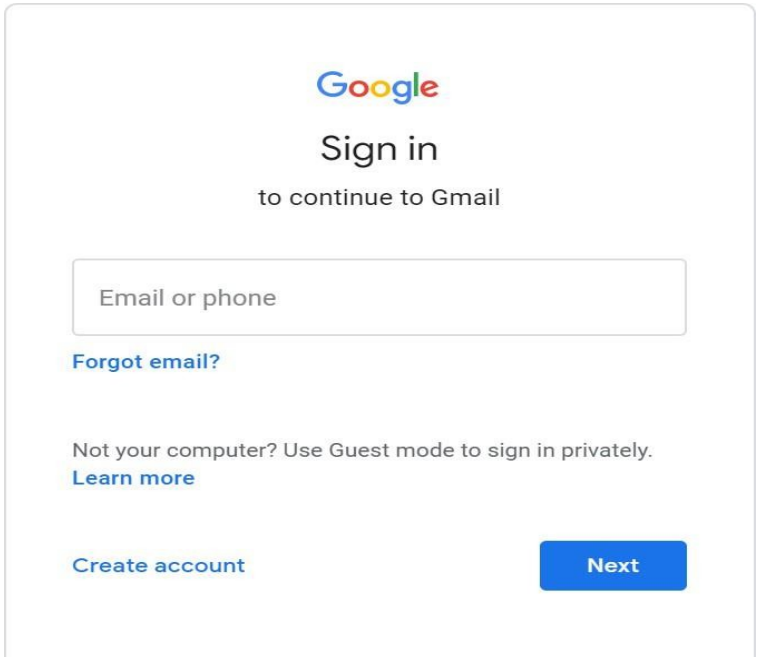


EventHandling

WhatisEventHandling?(DelegationEventModel)

Event Handling is the mechanism that controls the event and decides what should happen if an event occurs.The **delegation event model** defines standardandconsistentmechanismstogenerateandprocessevents.Itsconcept isquitesimple:**a*source* generatesaneventandsendsittooneor more *listeners*.**In this scheme, **the listener simply waits until it receives an event. Once received, the listener processes the event and then returns.**

Note:Inthefollowingexamplethebutton**Next**is**Eventsource.**Whenwe**click thebutton**an**Eventisgenerated**.The **Eventislistened** bythe**listeners**to perform the next action going to the next page.



TheDelegationEventModelhasthefollowingkeyparticipantsnamely:

- **Event -** In the delegation model, an event is an object that describes a state changeinasource.Itcanbegenerated asaconsequenceofapersoninteracting with the elementsin agraphical user interface.Some of the activitiesthat cause events to be generated are pressing a button, entering a character via the keyboard, selecting an item in a list, and clicking the mouse.

- **EventSource**-Asourceisanobjectthatgeneratesanevent.Thisoccurswhen theinternalstateofthatobjectchangesinsomeway.Sourcesmaygeneratemore than one type of event. Asource must register listeners in order for the listeners to receive notifications about a specific type of event. Each type of event hasits own registration method.

  Hereisthegeneralform:publicvoidaddTypeListener(TypeListenertl) Note: TypeListener is changed depending on the Event.
- **EventListener**-Alistenerisanobjectthatisnotifiedwhenanevent

  occurs.Ithastwo major requirements.First,itmusthave beenregistered with one or more sources to receive notifications about specific types of events. Second, it must implement methods to receive and process these notifications.

EventDelegationMethods

(i) Event

predefinedclass

(ii) Event Source

awt controls (text area,text fields,scroll bars,list,button, labels, choice)

(iii) EventListener

predefined interface

- Whenanactionisperformed onaneventsource,eventisgenerated.
- Theevent listenerswillwait untilan event isgenerated.
- Onceanevent isperformed,oneor morelistenerswill listentheeventsandparse the corresponding methods.

JavaEventclassesandListenerinterfaces

EventClasses	ListenerInterfaces
ActionEvent	ActionListener
MouseEvent	MouseListenerand MouseMotionListener
KeyEvent	KeyListener
ItemEvent	ItemListener
TextEvent	TextListener
AdjustmentEvent	AdjustmentListener

WindowEvent	WindowListener
ComponentEvent	ComponentListener
ContainerEvent	ContainerListener
FocusEvent	FocusListener

Note: Ineventhandlingprograms,we mayoverrideinit()andpaint()method.

Ininit()methodisoverriddenwhenweneedtodofollowing:

```
public voidinit(){
        //1.Createobjectsfor classusingconstructor
                                (foreventsourceswhichareawt controls)
       //2.Registrationoflistenerstosource(sourcecanbeapplet  window/awtcontrols)
                                (publicvoidaddTypeListener(TypeListenertl))
        //3.Addingourown awtcontrolsto applet.
                                 (add()methodisused)
   }
```

paint() methodis overriddenwhenwewanttodisplaysomethingon theapplet window.

HandlingmouseandkeyboardEvents

1) Mouseevents:-

Thiseventindicatesamouseactionoccurred inacomponent.Thislow-leveleventis generated by a component object for Mouse Events and Mouse Motion events.

MouseEvent:-(class) //classnamemustbetheargumentofthemethodsofinterface

```
int getX()
intgetY() intgetClickCounts()
```

MouseListener:-(Interface)

```
voidmouseClicked(MouseEvent me)
void mousePressed(MouseEvent me)
voidmouseReleased(MouseEventme)
voidmouseEntered(MouseEvent me)
voidmouseExited(MouseEvent me)
```

MouseMotionListener:-(interface)

```
voidmouseDragged(MouseEvent me)
```

voidmouseMoved(MouseEvent me)

//Note:For MouseEvent and Key board Eventprogramsawt controlsarenot required. For these two programs the source is applet window.

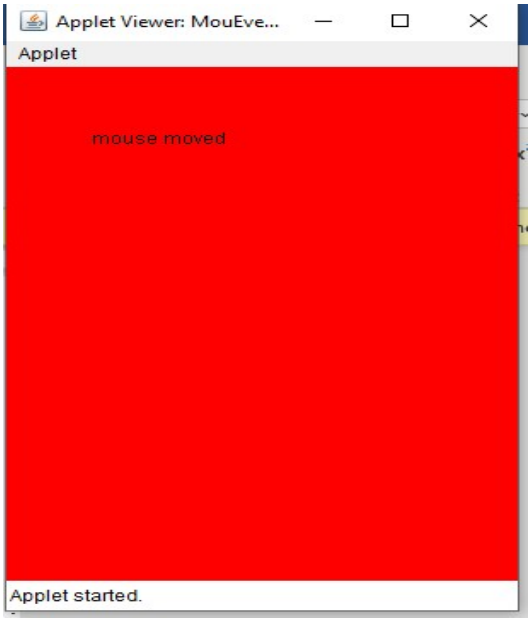
Program:Handling MouseEvent

```
importjava.applet.*;
importjava.awt.*;
importjava.awt.event.*;
/*<appletcode="MouEve.class"width=300 height=400></applet>*/

public classMouEveextends Appletimplements MouseListener,
MouseMotionListener{
Stringmsg="";
public voidinit(){
//overridinginit()
addMouseListener(this);                    //toregisterourlistenerto
applet window addMouseMotionListener(this);
}
    public voidmouseClicked(MouseEvent
me){ msg="Mouseclicked"; repaint();
    }
    public
        voidpaint(Graphicsg){ g.dra
        wString(msg,50,60);
}
public
voidmouseEntered(MouseEventme){ setBackground(Color.red);
repaint();
}
public
voidmouseExited(MouseEventme){ setBac
kground(Color.yellow);  msg="mouse
exited";
repaint();
}
public
            voidmouseReleased(MouseEventme)
            { setBackground(Color.green);
            repaint();
}
    public
    voidmousePressed(MouseEventme){ setBackground(Color.blue);
    repaint();
```

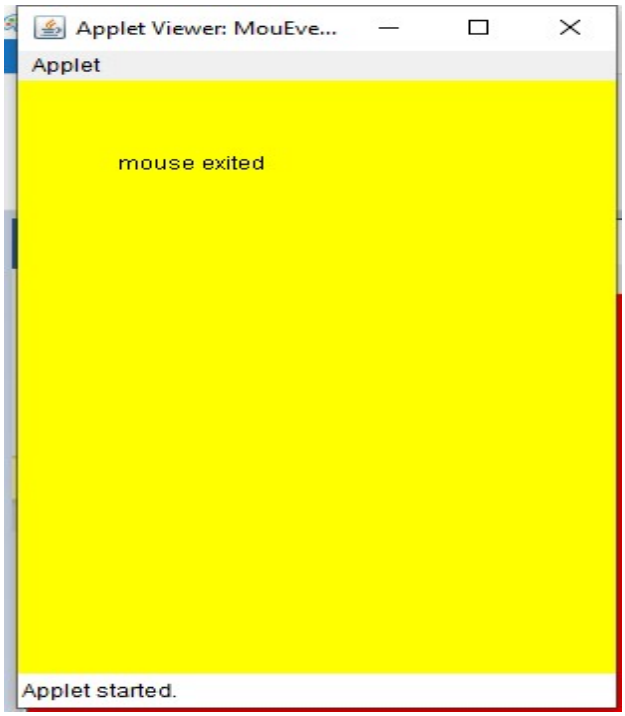
```
repaint();
}
public
    voidmouseMoved(MouseEventme){ m
    sg="mousemoved";
        repaint();
 }
}
```

Note:Outputofpaint()methodwhen theappletis run

Note: output after moving the mouse pointer into the applet window. When we move the mouse pointer into the applet window two methods areperformed.mouseMoved() andmouseEntered() methods.Soweare able to see the message “mouse moved” present in the mouseMoved() methodand the applet window color is changed to red as mouseEntered() method is also executed.

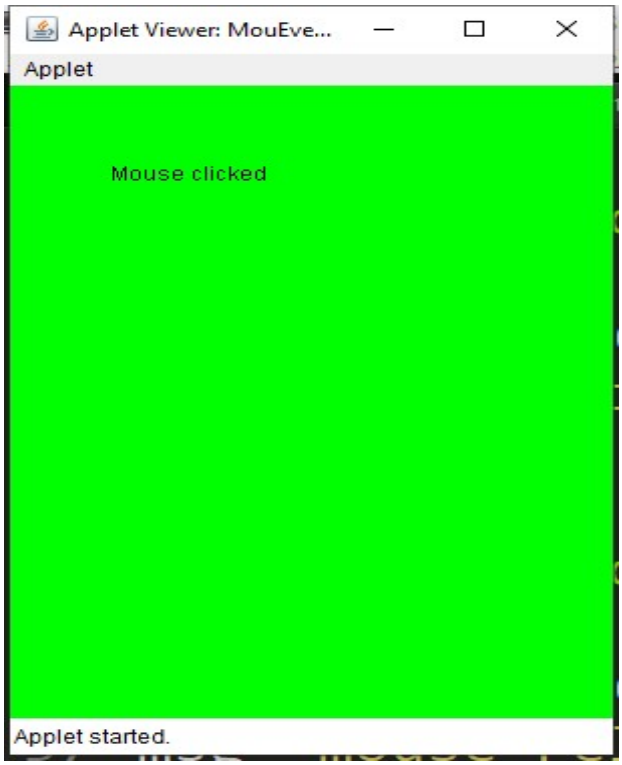


Note: outputaftermoving themousepointer outoftheappletwindow. MouseExited method is executed.



Note: For draggingthe mousewehaveto press themouse buttonand drag so the two methods are performed mousePressed() and mouseDragged().SofromthemousePressed()method thecolorofthe applet window is changed and from the mouseDragged() method the message " mouse dragged" is displayed on the applet window

Note: When we click the mouse three methods are executed mousePressed(), mouseReleased() and mouseClicked(). From the mouseClicked() method the message "mouse clicked" is displayed and fromthe mouseReleased()methodthecoloroftheappletischanged to green.



repaint()method

Thepaint()methodsupportspaintingviaa**Graphics**object.Therepaint()method is used to cause paint() to be invoked by the AWT painting thread. It's not necessary to call repaint unless you need to render something specific onto a component. As the applet needs to update information displayed in its window (i.e. redraw thewindow),each timethe mouse is beingclickedsothisispossible with theuseofrepaint()method. Tosumup, the repaint()methodis invokedto refresh the viewing area

2) Keyevents:-Thisevent indicateshowthekeysin keyboardarepressed/clicked.

KeyEvent(class)

(i) chargetKeyChar()

(ii) intgetKeyCode()

KeyListener(interface)

(i) voidkeyPressed(KeyEventke)

(ii) voidkeyTyped(KeyEventke)

(iii) voidkeyReleased(KeyEventke)

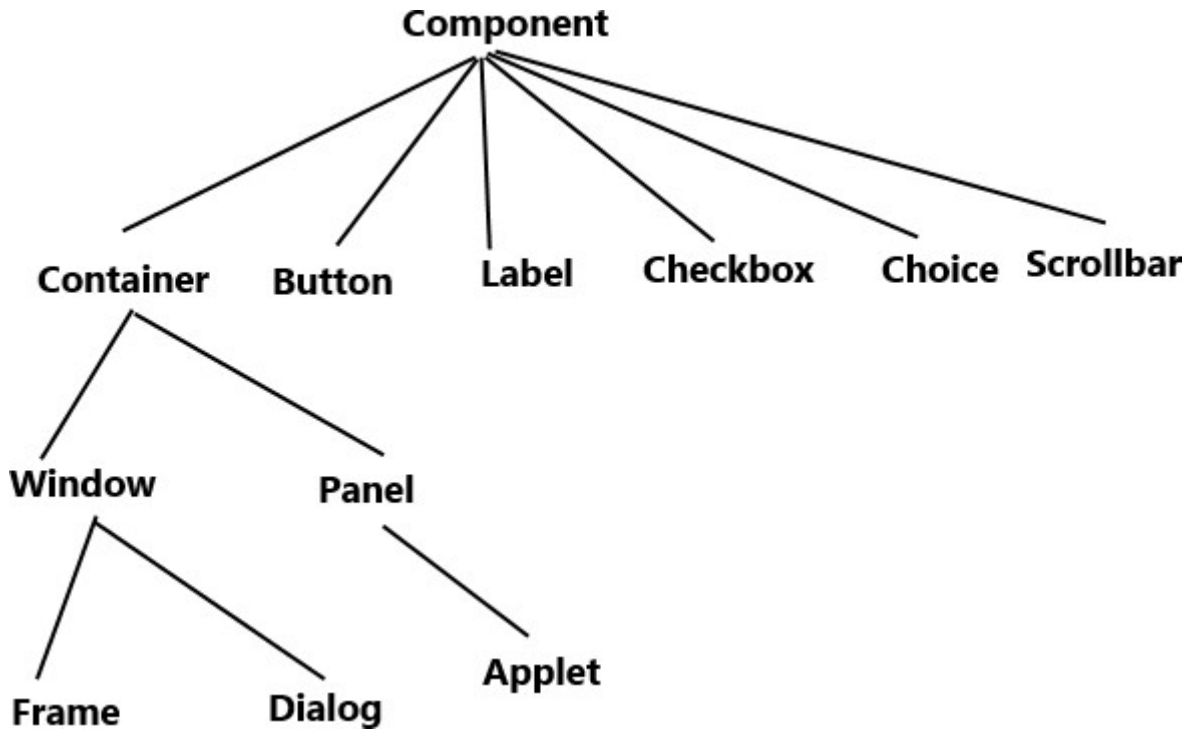
// Key events program

```
importjava.applet.Applet;
importjava.awt.*;
importjava.awt.event.*;
/*<appletcode="KeyEve.class"width=200height=300>
</applet>*/
public classKeyEveextends Applet
    implementsKeyListener{ Char ch=' ';
    public voidinit(){
    addKeyListener(this);            //to register our listeners to applet
    requestFocus();  //tomovethefocusofkeywordtoappletwindow
    }
    public
    voidkeyPressed(KeyEventke){ setBackground(Color.green);
    }
    public
    voidkeyReleased(KeyEventke){ showStatus("key
    released");
```

```
    public voidkeyTyped(KeyEventke){
    ch=ke.getKeyChar();
    repaint();
    }
    public
    voidpaint(Graphicsg){ g.drawString(""+ch,100,100);
    }
}
```

Note:When the keyistyped keyPressed(),keyReleased()andkeyTyped()methodsare executed.

AWTClassHierarchy:-



AwtComponents

Label,TextField,Button,Choice,List,Checkbox,CheckboxGroup,TextArea,Scrollbars

1. Label :-

Label–isaclass.

The objectof Label classis a component for placing text in a container.It is used to displayasingle lineof read onlytext.The text canbe changed byan applicationbut a user cannot edit it directly.

Constructors:

(i) Label()

(ii) Label(Stringstr)

(iii) Label(String str,int how)

how=Label.LEFT,Label.RIGHT,Label.CENTER

Methods in Label:

voidsetText(Stringstr)
String getText()
voidsetAlignment(inthow)
int getAlignment()

2. Text Fields:-

TextField-isaclass.

The objectof aTextField classis atext component that allows theediting of asingleline text. It inherits TextComponent class.

(i) TextField()

(ii) TextField(intnumchar)//forvisibility

(iii) TextField(Stringstr)//defaultString

(iv) TextField(Stringstr,intnumchar)

Methods inTextField:

StringgetText()

voidsetText(string str)

voidgetSelectedText()//selectusingthecursor void
select(int startIndex,intendIndex)
booleanisEditable(Booleancanedit)//defaulttrue
void setEchoChar(char ch)// for passwords
booleanechoCharIsSet()
chargetEchoChar()

3. Buttons:-Button–isaclass

The button classisusedtocreate alabeled buttonthat hasplatformindependent implementation.The applicationresult in some actionwhen thebuttonis pushed.

(i) Button()

(ii) Button(Stringstr)

MethodsinButtonClass:

void setLabel(String str)
String getLabel()

Action Events:-

ActionEvent-isaclass

MethodsinActionEventclass:

ObjectgetSource()//returnsthereferencevariable oftheButton String
getActionCommand()//it returns the label of Button

ActionListener–isanInterface

The Java ActionListener is notified whenever you click on the button or menu item. It is notifiedagainstActionEvent.TheActionListenerinterfaceisfoundin java.awt.eventpackage. It has only one method: actionPerformed().

MethodinActionListener:

public voidactionPerformed(ActionEventae)

// ProgramonLabelsandTextfields

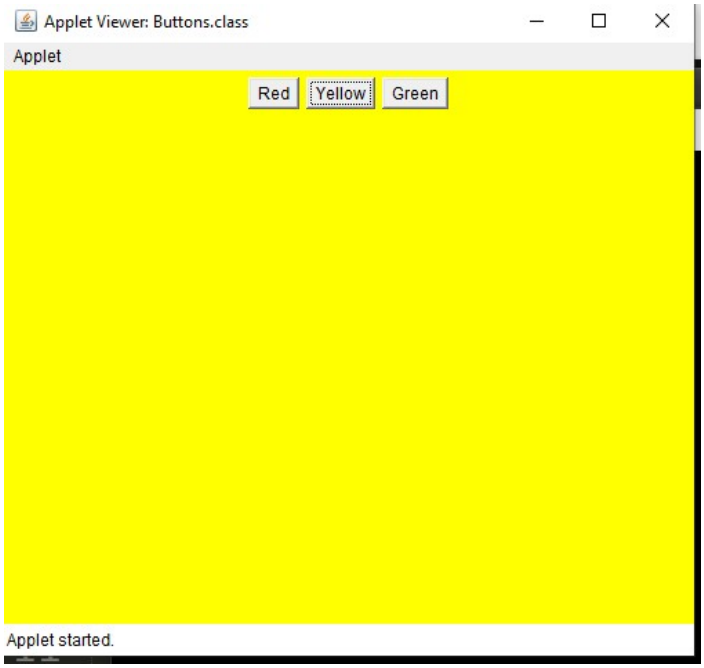
```
   importjava.applet.*;
importjava.awt.*;
   importjava.awt.event.*;

   /*<appletcode="LabelsTextEx9.class"width="600"height="600">

   </applet>*/

   public

      classLabelsTextEx9extendsAppletimplementsActionListener{ Labell1,l2;

      TextFieldt1,t2;

      Button b1;

      //Firstinit()andthenpaint()methodwill beexecutedandinitialoutputisprinted

      ontheappletwindow.

      //wheneverweclicktheButtonthenactionPerformed() methodis executed

   public voidinit(){

         l1=newLabel("username");//creatingobjects

l2=new Label("password");

         t1=new TextField(" ");
         t2=new TextField(" ");
         t2.setEchoChar('*');
         b1=newButton("signin");
         b1.addActionListener(this);//registersthelistenertotheButton
         add(l1);//adding the objects to applet window
         add(t1);
         add(l2);
        add(t2);
        add(b1);
```

```
}

    //whenweclicktheButtonactionPerformedisinvokedbytheactionListener

public voidactionPerformed(ActionEventae){

            repaint();
}

public voidpaint(Graphics

      g){ g.drawString(t1.getText(),100,100);
        g.drawString(t2.getText(),100,150);

}

}
```

//javaProgramonButtons

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;

/*<appletcode="ButtonsEx10.class"width=500

height=400></applet>*/

public class ButtonsEx10 extends Applet

implementsActionListener{ Buttonb1,b2,b3;

    //Firstinit()andthenpaint()methodwill beexecutedand
    //initial outputisprintedontheappletwindow.
    //whenevertheButtonisclickedactionPerformed()methodisexecuted

public voidinit(){

        b1=new Button("Red");
        b2=newButton("Yellow");
        b3=newButton("Green");

       b1.addActionListener(this);
       b2.addActionListener(this);
       b3.addActionListener(this);
       add(b1);
add(b2);
add(b3);
}

public

            voidactionPerformed(ActionEventae){ if(ae.getSourc

            e()==b1)

            //getSource()methodreturnstheobjectreferenceofButton

            setBackground(Color.red);

 else if(ae.getSource()==b2)

            setBackground(Color.yellow);

 else

            setBackground(Color.green);
```

```
}

}
```



4. Choice

Theobjectof **Choice** classisusedtoshowpopup menuofchoices.Choiceselected by user is shown on the top of a menu.It inherits Component class. Only one can be selected, visibility and selection may or may not be same.

Constructor:

Choice()

MethodsinChoiceclass:-

void add(String name)
intgetSelectedIndex()//returnstheindexoftheselected item String
getSelectedItem()// returns the item which is selected int
getItemCount()// Total number of items in Choice
void selected(intindex)//defaultselection

void select(String name)// selecting a String from the
itemsStringgetItem(int index)//returnsthe itemin theparticularindex

ItemEvents:-

ItemEvent–isaclass

ObjectgetItem()

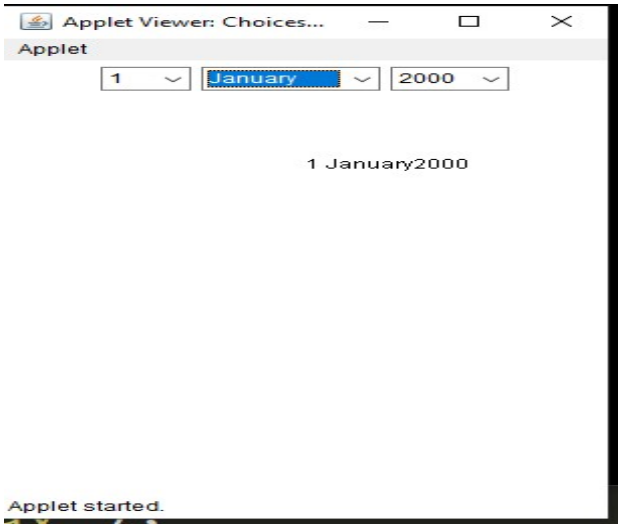
ItemListener–isanInterface

voiditemStateChanged(ItemEventie)

//theitemStatewillchangefromeither selectiontonon-selectionornon-selectionto selection

```
//ProgramonChoicecomponent
    importjava.applet.*;
    importjava.awt.*;
    importjava.awt.event.*;

/*<appletcode="ChoiceEx12.class"height=400width=300></applet>*/

    public class ChoiceEx12 extends Applet

implementsItemListener{ Choicec1,c2,c3;

//Firstinit()andthenpaint()methodwill beexecutedandinitial outputis
printedontheappletwindow.whenever thestateof anyChoiceis
changeditgoestoitemStateChangedmethod

public voidinit(){

                    //creatingobjectsofChoiceandaddingtheitemstothechoice

 c1=new Choice();

for(inti=1;i<=30;i++)
   c1.add(i+"");

2=new Choice();

2.add("January");
          c2.add("February");
          c2.add("March");
          c2.add("April");
          c2.add("May");
          c2.add("June");
          c2.add("July");
          c2.add("August");
          c2.add("September");
c2.add("October");
c2.add("November");
c2.add("December");


            c3=new Choice();
            for(inti=2000;i<=2015;i++)
```

```
            c3.add(i+"");

            add(c1);
            add(c2);
            add(c3);

            c1.addItemListener(this);
c2.addItemListener(this);
c3.addItemListener(this);
            }

            public

                  voiditemStateChanged(ItemEventie){ repaint(

                  );

            }

            public

              voidpaint(Graphicsg){ Stringstr1=c1.getSelectedItem()+c2.getSelectedItem(

              )+c3.getSelectedItem();
                g.drawString(str1,150,100);

            }
```



List

TheobjectofListclassrepresentsalistoftextitems.Bythehelp oflist,user canchoose either one item or multiple items. It inherits Component class.

List-isaclass

Used formultipleselectionsonitems

Constructors:

List()//only singleselectionandvisibilityisoneitem

List(int numRows)//onlysingleselectionandvisibility isnumRowsList(int numRows,Booleanmultipleselect)
//ifmultipleselct=truethenonlymultipleselectionispossibleotherwisesingleselection

MethodsinList:

void add(Stringname)

voidadd(Stringname,intindex)
String getSelectedItem()

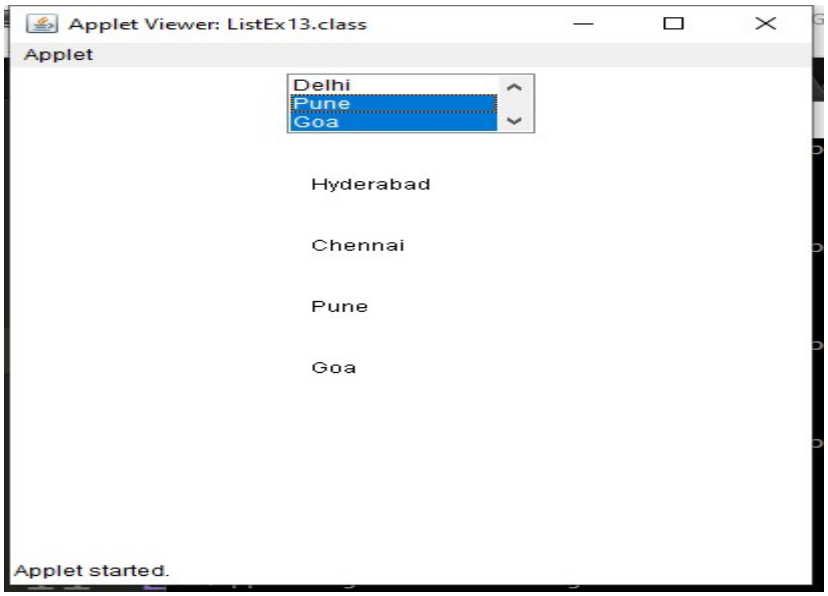
String[] getSelectedItems()
int getSelectedIndex()
int[]getSelectedIndexes()
int getItemCount()
void Select(int Index)
StringgetItem(intIndex)

//ProgramonListcomponent

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;

/*<appletcode="ListEx13.class"height=400width=400>

</applet>*/

public

classListEx13extendsAppletimplementsItemListener{ ListL1;//forcity

        //Firstinit()andthenpaint()methodwillbeexecutedandinitialoutputisprintedontheappletw
        indow.wheneverthestateof anyListischangeditgoestoitemStateChangedmethod
```

```
public

voidinit(){ L1=newList(3,true);

//3visibleoutof6 anyno.of itemsareselected

L1.add("Hyderabad");
L1.add("Bangolore");
L1.add("Chennai");
L1.add("Delhi");
L1.add("Pune");
L1.add("Goa");

                L1.addItemListener(this);

add(L1);

            }

            public

                        voiditemStateChanged(ItemEventie){ repa

                        int();

            }

            public voidpaint(Graphicsg){

                        Stringstr[]=L1.getSelectedItems(); for(int
                        i=0;i<str.length;i++)

                                    g.drawString(str[i],150,100+i*50);

            }
```

Note: Wecanselectmultipleitems byholding shiftkey.Selecteditems areprintedas outputontheapplet window



5. Checkbox

The **Checkbox** classis usedto create a checkbox. It is used to turn anoption on (true) or off (false).Clickingon aCheckboxchangesitsstate from"on" to"off" orfrom"off" to "on".

Constructors:

CheckBox()

CheckBox(String str)// default not selected
CheckBox(String str,boolean on)// true then selected
CheckBox(Stringstr,booleanon,CheckBoxGroupcbg)
CheckBox(String str, CheckBoxGroupcbg)

MethodsinCheckBox:

BooleangetState()

voidsetState(booleanon)
String getLable()
voidsetLabel(Stringstr)

//EventsourceisCheckbox ItemEvent:
//Event
ObjectgetItem()//itreturnsthereference variableoftheobject
ItemListener:// Listener

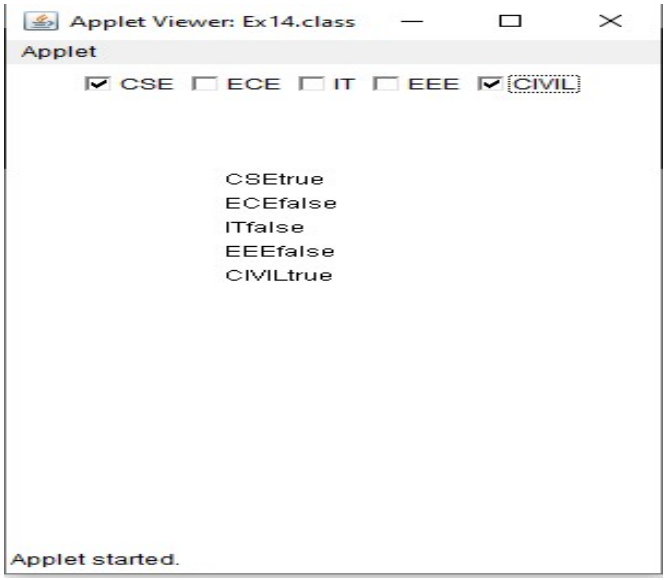
voiditemStateChanged(ItemEventie)

//ProgramonCheckBox

```
importjava.applet.*;
importjava.awt.*;
importjava.awt.event.*;

/*<appletcode="CheckboxEx14.class"height=400width=300>
</applet>*/

public class CheckboxEx14 extends Applet implements
ItemListener{

Checkboxcb1,cb2,cb3,cb4,cb5;

//Firstinit()andthenpaint()methodwill beexecutedandinitial outputis

printedontheappletwindow.wheneverthestateof anyCheckboxis

changeditgoestoitemStateChangedmethod

public voidinit(){

        cb1=newCheckbox("CSE");//notselected
        cb2=newCheckbox("ECE");
        cb3=newCheckbox("IT");
        cb4=newCheckbox("EEE");
        cb5=newCheckbox("CIVIL");

        add(cb1);
        add(cb2);
        add(cb3);
        add(cb4);
        add(cb5);

        cb1.addItemListener(this);
        cb2.addItemListener(this);
        cb3.addItemListener(this);
        cb4.addItemListener(this);
        cb5.addItemListener(this);

}

/*Whenever thestateofcheckboxischangedfromonestatetoanother
stateItemEventisgeneratedsoitemStateChanged()methodisinvokedbyitemListener*/
```

```
public

voiditemStateChanged(ItemEventie){ repain

t();//overwritetheStringinthe output

}

//AspartofAppletLifecyclemethods init()andpaint()methodwill be

//executedautomatically

public voidpaint(Graphics g){

    Stringstr1=cb1.getLabel()+cb1.getState();
    Stringstr2=cb2.getLabel()+cb2.getState();
    Stringstr3=cb3.getLabel()+cb3.getState();
    Stringstr4=cb4.getLabel()+cb4.getState();
    Stringstr5=cb5.getLabel()+cb5.getState();

    g.drawString(str1,100,100);
    g.drawString(str2,100,120);
    g.drawString(str3,100,140);
    g.drawString(str4,100,160);
    g.drawString(str5,100,180);

}
```

//multipleitemscanbeselected.Thelabelandthestateofeachcheckboxisprintedasoutput.
Note: 1 Whenthepaint() methodis invokedautomaticallyfor thefirsttimethestateof allthecheckboxes in the output is false.
2:Whenevertheitemstateischangedforany Checkboxtherepaint() methodiscalledfromthe itemStateChange() method then the String(Label and State)is overwritten in the output.
3:Note1 andNote2 areapplicableforalltheEventhandling Programs.



6. CheckboxGroup(radiobutton)

The object of CheckboxGroup class is used togroup together a set of Checkbox.At a timeonlyonecheckboxbuttonisallowed tobein"on" stateandremaining checkbox button in "off" state. It inherits the object class.

Constructor:

CheckboxGroup()

Methods

CheckboxgetSelectedCheckbox()

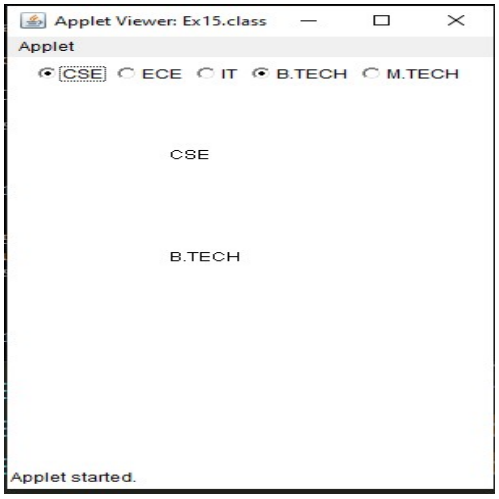
voidsetSelectedCheckbox(Checkboxwhich)

//ProgramonCheckBox

```
    import java.applet.*;
    import java.awt.*;
importjava.awt.event.*;

    /*<appletcode="CheckboxEx15.class"height=400width=300>
    </applet>*/

public class CheckboxEx15 extends Applet

    implementsItemListener{ CheckboxGroupcbg1,cbg2;
```

```
Checkboxcb1,cb2,cb3,cb4,cb5;

//Firstinit()andthenpaint()methodwill beexecutedandinitial outputisprintedon

theappletwindow.

//whenever thestateof anyCheckboxischangeditgoestoitemStateChangedmethod

public voidinit(){

        cbg1=newCheckboxGroup();
        cbg2=newCheckboxGroup();

        cb1=newCheckbox("CSE",cbg1,true);
        cb2=newCheckbox("ECE",cbg1,false);
        cb3=newCheckbox("IT",cbg1,false);

        cb4=newCheckbox("B.TECH",cbg2,true);
        cb5=newCheckbox("M.TECH",cbg2,false);

        add(cb1);
        add(cb2);
        add(cb3);
        add(cb4);
        add(cb5);

          cb1.addItemListener(this);
          cb2.addItemListener(this);
          cb3.addItemListener(this);
          cb4.addItemListener(this);
          cb5.addItemListener(this);

}

public

voiditemStateChanged(ItemEventie){ repaint();

}

public voidpaint(Graphicsg){

 Checkboxcb6=cbg1.getSelectedCheckbox();
 g.drawString(cb6.getLabel(),100,100);
 Stringstr=(cbg2.getSelectedCheckbox()).
 getLabel(); g.drawString(str,100,200);
```

```
    }

}
```



Note:Atatimefromagroupwecanselectoneitem

7. Scrollbars

TheobjectofScrollbarclassisused toaddhorizontaland verticalscrollbar.Scrollbaris a GUI component allows us to see invisible number of rows and columns.

Constructors:

Scrollbar()Scrollbar(in
tstyle)
Scrollbar(intstyle,intinitialValue,intthumbSize,intmin,intmax)

If style is **Scrollbar.VERTICAL**, averticalscrollbar is created.
Ifstyleis**Scrollbar.HORIZONTAL**,thescrollbarishorizontal.

Inthethird formof theconstructor,theinitialvalueof thescrollbarispassed in **initialValue**.The initialValue must be in between min and max values. **initialValue is the position of thethumb.**

Thenumberof unitsrepresentedbytheheightof the **thumb**ispassedin **thumbSize.**
The minimum and maximum values for the scroll bar are specified by **min and max. Methods of Scrollbar:**

voidsetValues(int initialValue, int thumbSize, int min, int max)
intgetValue() //positionof thethumbwhichistheinitialvalue
voidsetValue(int newValue)
intgetMinimum( )
intgetMaximum()
voidsetUnitIncrement(intnewIncr)//newIncr=3intialvalue+unit*newIncr 14710
void setBlockIncrement(intnewIncr)//newIncr=3intialvalue+Block*newIncr1316191

//Blockmeans10,unitmeans1,initial valueis 1

Event

AdjustmentEvent

Listener

AdjustmentListener

publicvoidadjustmentValueChanged(AdjustmentEventae)

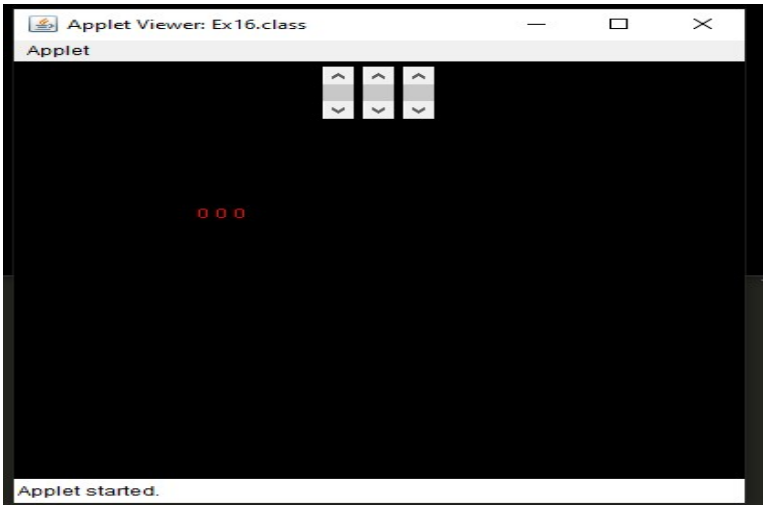
//ProgramonScrollbars

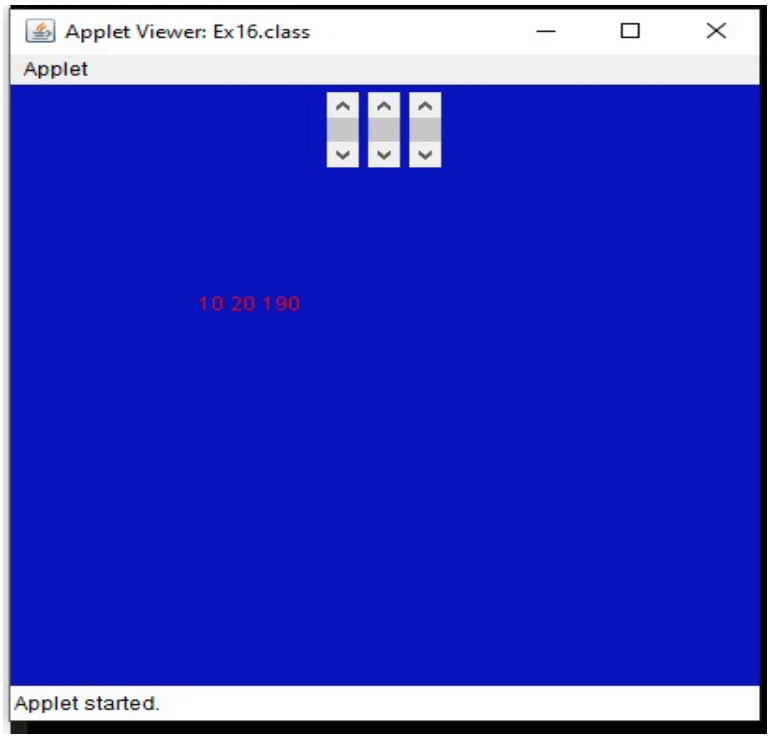
```
import t java.applet.*;
importjava.awt.*;
import java.awt.event.*;

/*<appletcode="ScrollbarEx16.class"height=400width=400>

</applet>*/

public

 classScrollbarEx16extendsAppletimplementsAdjustmentListener{ Scrollbar

 s1,s2,s3;
  Color c;
   int r=0,g=0,b=0;
 publicvoidinit(){

     // Scrollbar(intstyle,intinitialValue,intthumbSize,intmin,intmax)

     s1=newScrollbar(Scrollbar.VERTICAL,10,2,0,255);
     s2=newScrollbar(Scrollbar.VERTICAL,20,2,0,255);
     s3=newScrollbar(Scrollbar.VERTICAL,30,2,0,255);

     add(s1);
     add(s2);
     add(s3);

       s1.setUnitIncrement(10);//10,20,30,…
       s2.setUnitIncrement(10);//20,30,40,50---.
       s3.setUnitIncrement(10);// 30,40,50---

       s1.addAdjustmentListener(this);
       s2.addAdjustmentListener(this);
```

```
    }

    public voidadjustmentValueChanged(AdjustmentEventae){

        r=s1.getValue(); //givesthepositionofthethumbthatistheintialvalue
        g=s2.getValue();
        b=s3.getValue();
        repaint();

}

    public voidpaint(Graphicsg){

        c=new
        Color(r,g,b);setBackground(c)
        ; setForeground(Color.red);

        g.drawString(r+""+g+""+b,100,150);

    }

}
```

//Ifupwardarrowisclickedthevalueisdecreasedandifdownwardarrowisclickedthevalueisincreased.
//Whenthepaint()methodis invokedforthefirsttimethentheoutputis000

Note: When we click anyscrollbar using downward arrow for the first time then adjustmentValueChanged methodwill beinvokedsothevalues of r,g,barechangedtotheinitial values oftheScrollbarsexceptfor the Scrollbar we clicked.





8. TextArea

The object ofaTextAreaclassisamultilineregionthatdisplaystext.Itallowstheediting of multiple line text. It inherits TextComponent class.
Syntax:
Constructors:
TextArea()
TextArea(intnumLines,intnumChars)
//numLinesisheightandnumCharsisthewidthof
theTextAreaTextArea(String str)
TextArea(String str, int numLines, int numChars)
TextArea(Stringstr,intnumLines,intnumChars,intsBars)

Here,numLinesspecifiestheheight,in lines,of thetextarea,and numCharsspecifiesitswidth,in characters.Initialtextcan bespecifiedby str.In thefifthform,you can specify thescrollbarsthat you want the control to have. sBars must be one of these values:
SCROLLBARS_BOTH
SCROLLBARS_NONE
SCROLLBARS_HORIZONTAL_ONLY
SCROLLBARS_VERTICAL_ONLY

TextArea is a subclass of TextComponent. Therefore, it supports the getText( ), setText( ), getSelectedText(),select(),isEditable( ),and setEditable() methodsdescribed in theTextField.

TextAreaaddsthefollowingmethods:

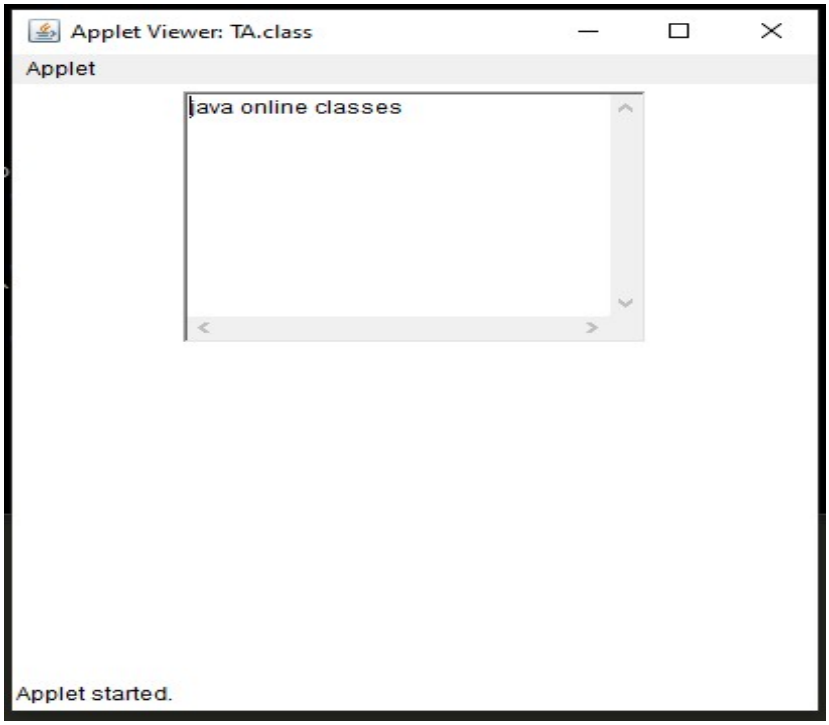
voidappend(Stringstr)

voidinsert(Stringstr,intindex)

voidreplaceRange(Stringstr,intstartIndex,intendIndex)

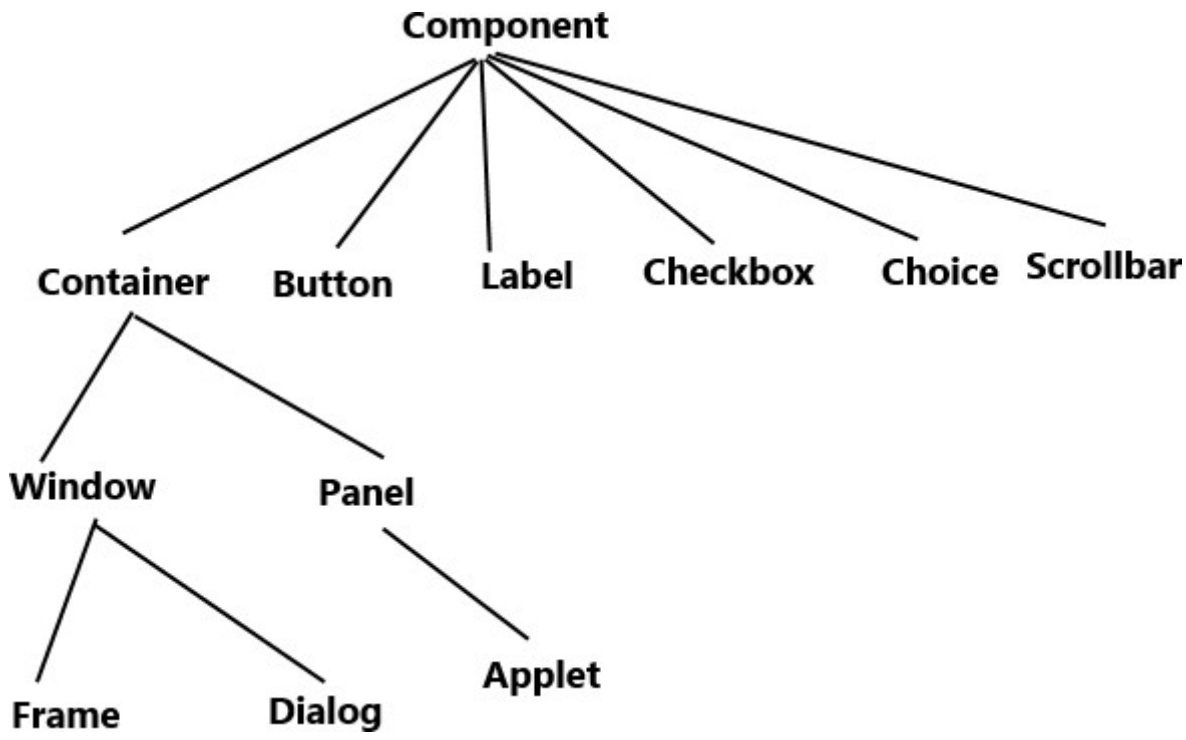
// Program on TextArea

```
importjava.awt.*;
importjava.applet.*;

/*<appletcode="TA.class"height=400width=400>
</applet>*/

public class TA extends Applet{

    public

    voidinit(){ TextAreata=newTextArea("javaonlineclasses

    ",10,30); add(ta);

    }

}
```



Source	Event	EventListener	MrthodinEventListener
Applet	MouseEvent	MouseListener	mouseClicked() mousePressed() mouseReleased() mouseEntered() mouseExited()
		MouseMotionListener	mouseMoved() mouseDragged()
	KeyEvent	KeyListener	keyPressed() keyReleased() keyTyped()
Label	-	-	-
TextField	-	-	-
Button	ActitonEvent	ActionListener	actionPerformed()
Choice List Checkbox CheckboxGroup	ItemEvent	ItemListener	itemStateChanged()
Scrollbar	AdjustmentEvent	AdjustmentListener	adjustmentValueChanged()
TextArea	-	-	-

Awtclasshierarchy



Container

The Container class is a subclass of Component. It has additional methods that allowotherComponentobjectstobenestedwithinit.OtherContainerobjectscan be stored inside of a Container (since they are themselves instances of Component). This makes for a multileveled containment system. A container is responsible for laying out (that is, positioning) any
components that it contains. It does this through the use of various layout managers.

Panel

APanelisawindowthatdoesnotcontainatitlebar,menubar,orborder. This is why you don't see these items when an applet is run inside a browser. When you run an applet using an applet viewer, the applet viewer provides the title and border

Window

TheWindowclasscreatesatop-levelwindow.Atop-levelwindowisnotcontained withinanyotherobject;itsitsdirectlyonthedesktop.Generally,youwon'tcreate Windowobjectsdirectly.Instead,youwilluseasubclass ofWindowcalledFrame, described next.

Frame

Frame encapsulateswhat is commonly thought of as a "window."It is asubclass of Window and has **a title bar, borders, and resizing corners**. If youcreate a Frame object from within an applet, it will contain a warning message, such as "Java Applet Window," to the user that an applet window has been created. This messagewarnsusers thatthewindowthey see wasstartedby an appletandnot by software running on their computer

Note:Framedoes not havestatusbar andmenu bar.

Canvas

Although it is not part of the hierarchy for applet or frame windows, there is one other type of window thatyou will findvaluable:Canvas. **Canvasencapsulates a blank window upon which you can draw.**

Constructors

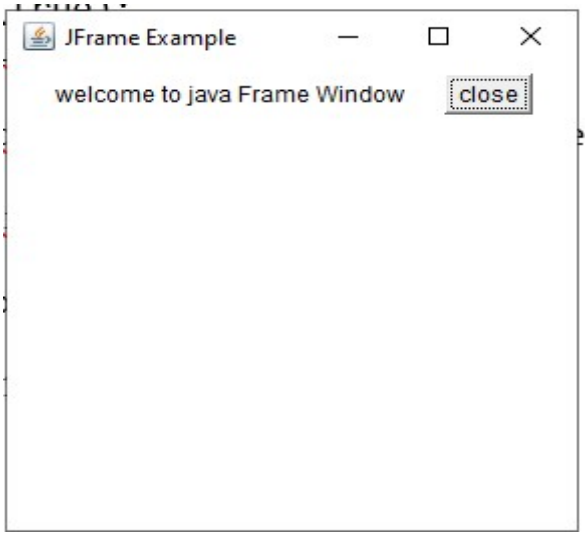
Frame()

Frame(Stringtitle)

//CreatingaFramewindowbyinstantiatingFrameclass

```
import java.awt.*;
importjava.awt.event.*;

public

        classFrameEx11implementsActionListener{ Fra

        me fm;
        FrameEx11(){

                fm=new Frame("JFrame Example");

                //Creatingaframe.

                Label lb = new Label("welcome to java Frame Window "); Button
                jb=new Button("close");

                fm.setLayout(new FlowLayout(FlowLayout.CENTER));
                fm.add(lb);//addinglabeltotheframe. fm.add(jb);

                fm.setSize(300, 300);//settingframesize.
                fm.setVisible(true);

                jb.addActionListener(this);

        }

        public

                voidactionPerformed(ActionEventae){ fm.
```

```
        public
                static void main(String args[]){ FrameEx1
                1ta=new FrameEx11();
        }
}
```



Adapterclasses

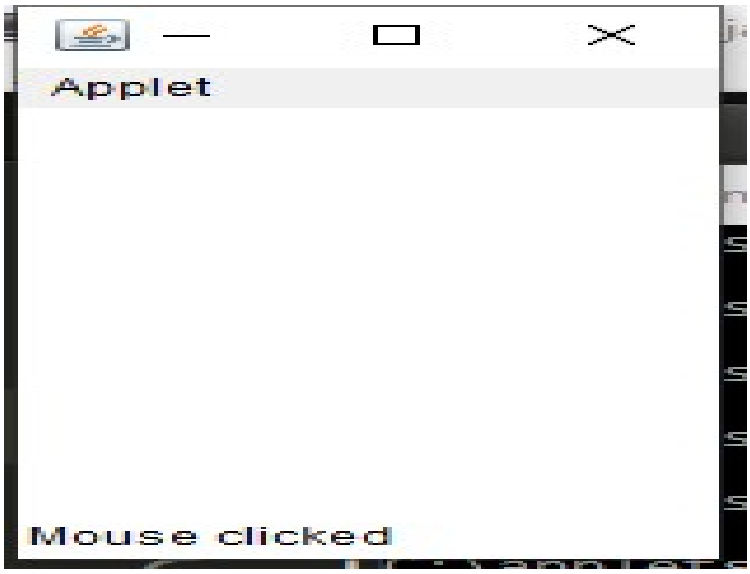
Java adapter classes provide the default implementation of listener *interfaces*. If weinherittheadapterclass,weenednotbeforcedtoprovidetheimplementation of all the methods of listener interfaces. So, it saves code.

Theadapterclassesarefoundin**java.awt.event**and**javax.swing.event** packages.

Note:1: IncaseofaddMouseListener(this),thisreferstothecurrentclassobject i.e., child class which extends Applet.
Note2: InterfaceswhichhavemorethanonemethodhascorrespondingAdapter classes.
ForExample:MouseListener–MouseAdapter
MouseMotionListener–
MouseMotionAdapterKeyListener–KeyAdapter

//ProgramonAdapterClasses

```
import java.awt.*;
importjava.awt.event.*;
import java.applet.*;

/*<appletcode="AdapterDemo.class"width=150height=200>
</applet>*/

public

        classAdapterDemoextendsApplet

        { public voidinit(){

                addMouseListener(newMyMouseAdapter(this));
                //eventsourceobject
                //thisreferstothecurrentclassobjectthatisAdapterDemo
          }

}

classMyMouseAdapterextendsMouseAdapter{

                AdapterDemoadapterDemo;

                publicMyMouseAdapter(AdapterDemoadapterDemo){ this.adapterDemo=adapt

                        erDemo;

                }

                public

                        voidmouseClicked(MouseEventme){ adapterDemo.showStatus("Mousecl

                        icked");
```



LayoutManager

A **Layout** Manager is useful to arrange components in a particular order in a frame or container. Container can be Applet, Window, Panel or a Frame. Components can be a Label, Button, TextField and so on. A layout manager is an instance of any class that implements the LayoutManager interface. The following classes represents the layout managers and available in java.awt package.

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

Note: When we add awt controls to Appletwindow the awt controls are added in the center of the Applet. The default layout is FlowLayout.

1. FlowLayout

The FlowLayout is used to **arrange the components in a line, one after another** (in a flow). FlowLayout is the default layout. It implements a simple layout style, which is similar to how words flow in a text editor.

Constructors of FlowLayout class

FlowLayout()
FlowLayout(inthow)
FlowLayout(inthow,inthorz,int vert)
//horzspacebetweenawtcontrolsw.r.ttwolinesandvertreferstothespacebetweenawtcontrolsinaline

Thefirstconstructorcreates thedefaultlayout,which centers components and leaves fivepixels of space between each component.

i.e.,FlowLayout(FlowLayout.CENTER,5,5)

Note: setLayout(new FlowLayout(FlowLayout.CENTER,5,5)) is by default available for the applet window. When we add awt controls to the applet window, the default layout is FlowLayout(FlowLayout.CENTER,5,5)

Thesecondconstructorletsyouspecifyhoweachlineisaligned.Validvaluesforhowareas follows:

```
public static finalint LEFT
public static finalint RIGHT
public static final int CENTER
public static final int LEADING
public static final intTRAILING
```

Note: LEADING meansleavingsomespaceinthe beginningand TRAILING means leaving some space at the end.

//ProgramforFlowLayout

```
importjava.applet.*;
importjava.awt.*;
importjava.awt.event.*;

/*<appletcode="FlowLayoutEx1.class"width=500height=400>
</applet>*/

public classFlowLayoutEx1extends
    AppletimplementsActionListener{ Buttonb1,b2,b3;
    public voidinit(){
//FlowLayout(inthow,inthorz,int vert)
        FlowLayout ob1=new FlowLayout(FlowLayout.RIGHT,10,10);
        setLayout(ob1);//theFlowLayoutissettotheAppletwindow

        b1=new Button("Red");
        b2=new Button("Yellow");
        b3=new Button("Green");
```

```
        b1.addActionListener(this);
        b2.addActionListener(this);
        b3.addActionListener(this);

        add(b1);
        add(b2);
        add(b3);
      }
    public
      voidactionPerformed(ActionEventae){ if(ae.getSource
      ()==b1){
        setBackground(Color.red);
}
    elseif(ae.getSource()==b2){ setBackground(Color.yellow
      );
     }
    else{ setBackground(Color.gree
      n);
}
}
}
```

2. BorderLayout

The Border Layout class implements a common layout style for top-level windows. It has four narrow, fixed-width components at the edges and one large area in the center. The four sides

are referred to as north, south, east and west. The middle area is called the center.

Constructors:

BorderLayout()
BorderLayout(inthorz,intvert)
Thefirst form creates adefaultborderlayout. Thesecond form allowsyouto specify thehorizontal and vertical space left between components in *horz*and *vert* respectively.

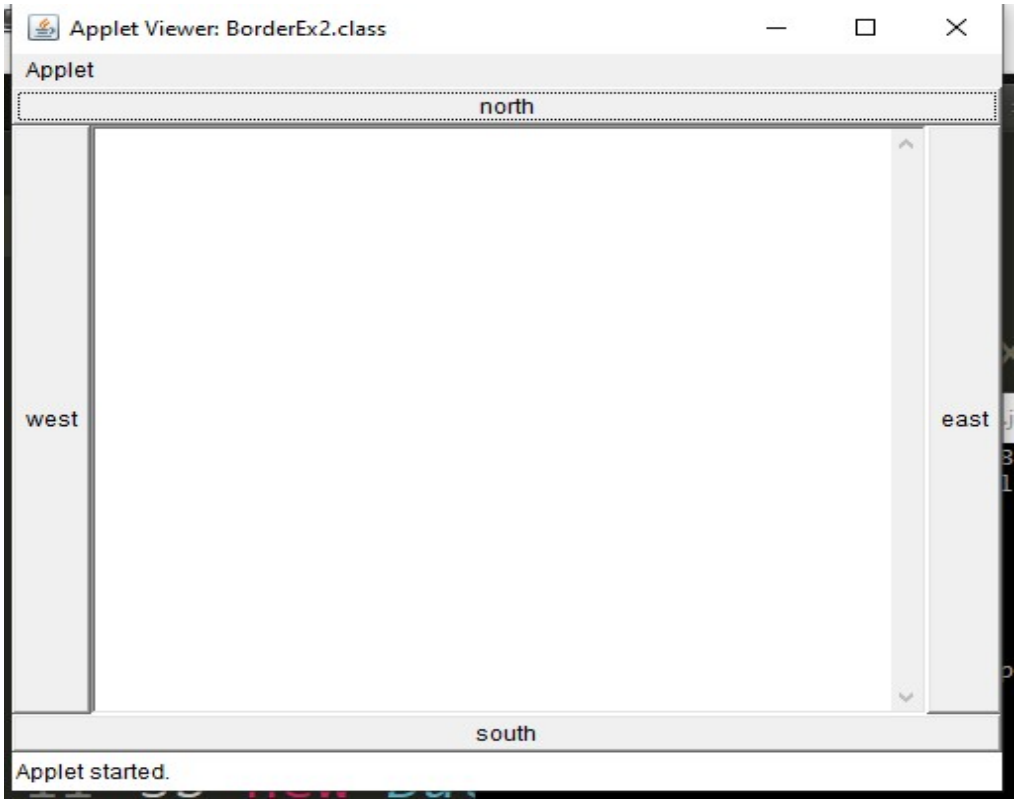
BorderLayoutdefinesthefollowingconstants(public,static,final)thatspecifytheregions:

BorderLayout.NORTHBorderLayout.SOUTHBorderLayout.EAST
BorderLayout.WESTBorderLayout.CENTER

//ProgramforBorderLayout

```
importjava.applet.*;
importjava.awt.*;
importjava.awt.event.*;
/*<appletcode="BorderLayoutEx2.class"width=500height=400>
</applet>*/
publicclassBorderLayoutEx2extendsAppletimplementsActionListener{
    TextAreata;
    Buttonnb,sb,eb,wb; public
    void init(){
    ta=new TextArea();
    nb=newButton("north");
    wb=new Button("west");
    sb=newButton("south");
    eb=new Button("east");

setLayout(newBorderLayout());
    nb.addActionListener(this);
    eb.addActionListener(this);
    sb.addActionListener(this);
    wb.addActionListener(this);

    add(nb,BorderLayout.NORTH);
    add(wb,BorderLayout.WEST);
    add(sb,BorderLayout.SOUTH);
```

```
        add(eb,BorderLayout.EAST);
        add(ta,BorderLayout.CENTER);
}
public
        voidactionPerformed(ActionEventae){ ta.setText(ae.getActionComm
        and());
}
}
```



3. GridLayout

GridLayoutlaysoutcomponentsinatwo-dimensionalgrid.

TheGridLayoutisusedtoarrangethecomponentsinrectangular grid.Onecomponentis displayed in each rectangle.

ConstructorsofGridLayoutclass

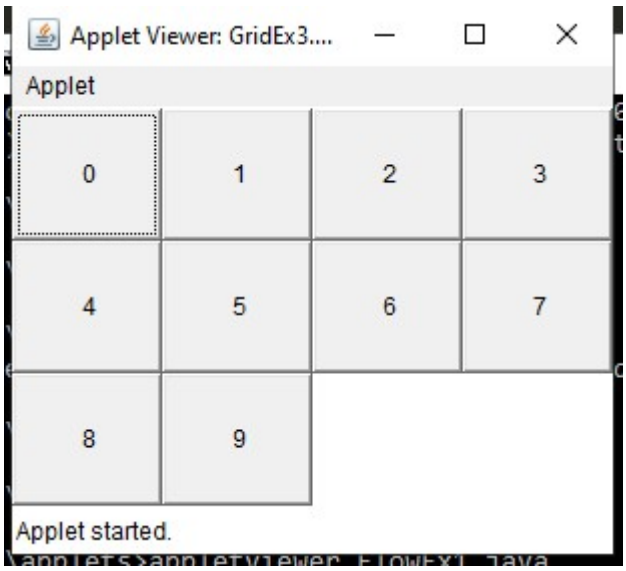
GridLayout():createsagridlayoutwithone columnpercomponent inarow.

GridLayout(introws,int columns):creates agrid layout withthegivenrows andcolumns but no gaps between the components.

GridLayout(introws, int columns, inthgap, intvgap):createsa gridlayout withthe given rows and columns along with given horizontal and vertical gaps.

//ProgramforGridLayout

```
    importjava.awt.*;
    importjava.applet.*;
    /*<appletcode="GridLayoutEx3.class"width=300height=200>
    </applet>*/
    public class GridLayoutEx3
    extendsApplet{ public voidinit(){
            setLayout(newGridLayout(3,4));
            for(int i=0;i<10;i++){
                    add(newButton(""+i));
}
    }

}
```



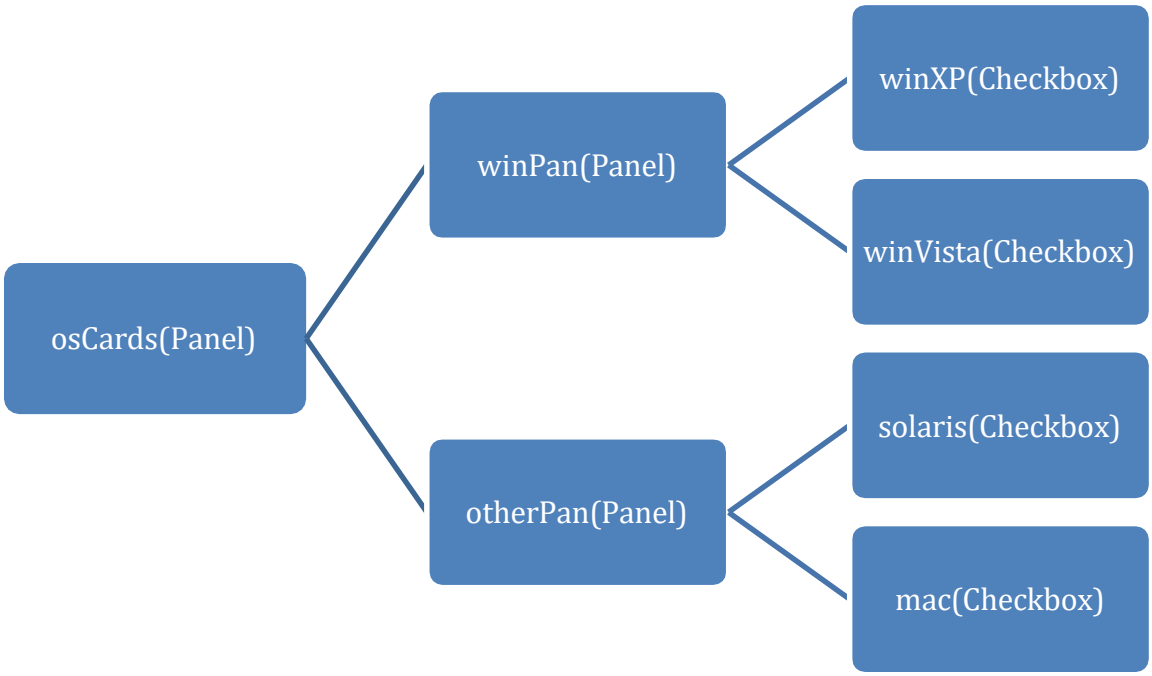
4. Card Layout

Constructors:

CardLayout()CardLayout(int
horz,intvert) **Methods:**
voidadd(ComponentpanelObj,StringcardName)

Note:cardNameisnotvisible. EachcardisonePanel. ThesecardNamesareusefulinselecting the random cards using the show() method.
voidfirst(Containerdeck)
void last(Container deck)
voidnext(Container deck)
voidprevious(Containerdeck)
voidshow(Containerdeck,StringcardName)//toselecttherandomcardwithcardName

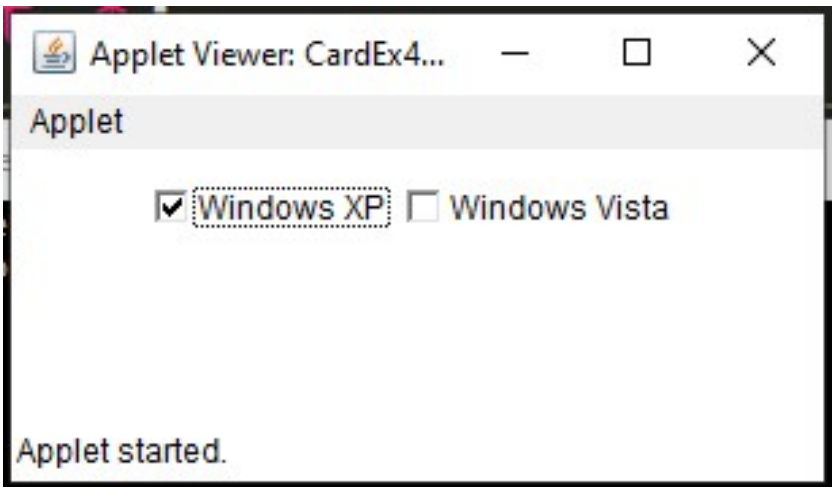
Note: In the next Program we are creating the following structure. osCards is a container deck whichholds 2cardswinPanandotherPan.winPanis aPanelwhichholds 2 checkboxes winXP and winVista. otherPan is aPanel which holds 2 checkboxes solaris and mac.



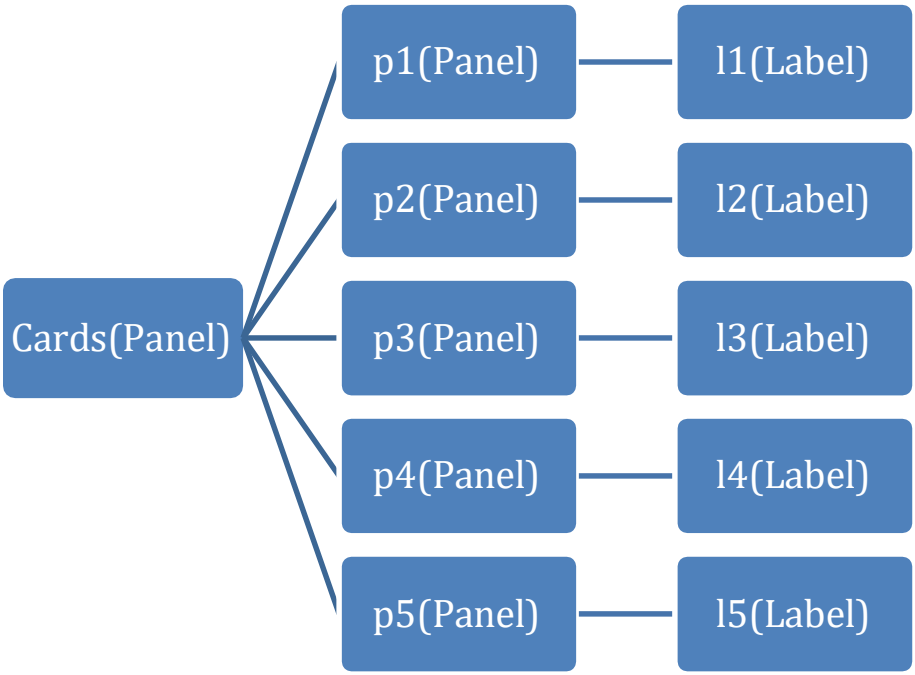
```
import java.awt.event.*;
importjava.awt.*;
importjava.applet.*;
/*<appletcode="CardEx3.class"width=300height=100>
</applet>*/
public class CardEx3 extends Applet implements
MouseListener{ CheckboxwinXP,winVista,solaris,mac; Panel osCards;
//Panel definition
CardLayoutcardLO;
public voidinit(){
        cardLO=newCardLayout();

        osCards= newPanel();
        osCards.setLayout(cardLO);//setcardlayouttocarddeckPanel

        winXP=newCheckbox("WindowsXP",null,true);
        winVista = new Checkbox("Windows Vista"); solaris
        = new Checkbox("Solaris");
        mac=newCheckbox("MacOS");

        //creating twoPanels
        PanelwinPan=newPanel();
        //add winXpandwinVistacheck boxestoa panel winPan.add(winXP);
        winPan.add(winVista);

        PanelotherPan=newPanel();
        //add solarisandmac checkboxestoapanel otherPan.add(solaris);
        otherPan.add(mac);

        // add panels to card deck panel
        osCards.add(winPan,"Windows");
        osCards.add(otherPan, "Other");

        //add card deck tomain appletwindow add(osCards);

// register mouse events
        addMouseListener(this);
        }
        //cyclethroughpanels
        public
                voidmousePressed(MouseEventme){ cardLO.next(o
                sCards);
        }
```

```
//Provideemptyimplementationsfortheother MouseListenermethods.
```

```
        public void mouseClicked(MouseEventme){
        }
        public voidmouseEntered(MouseEventme){
        }
        public voidmouseExited(MouseEventme){
        }
        public voidmouseReleased(MouseEventme){
        }
}
```

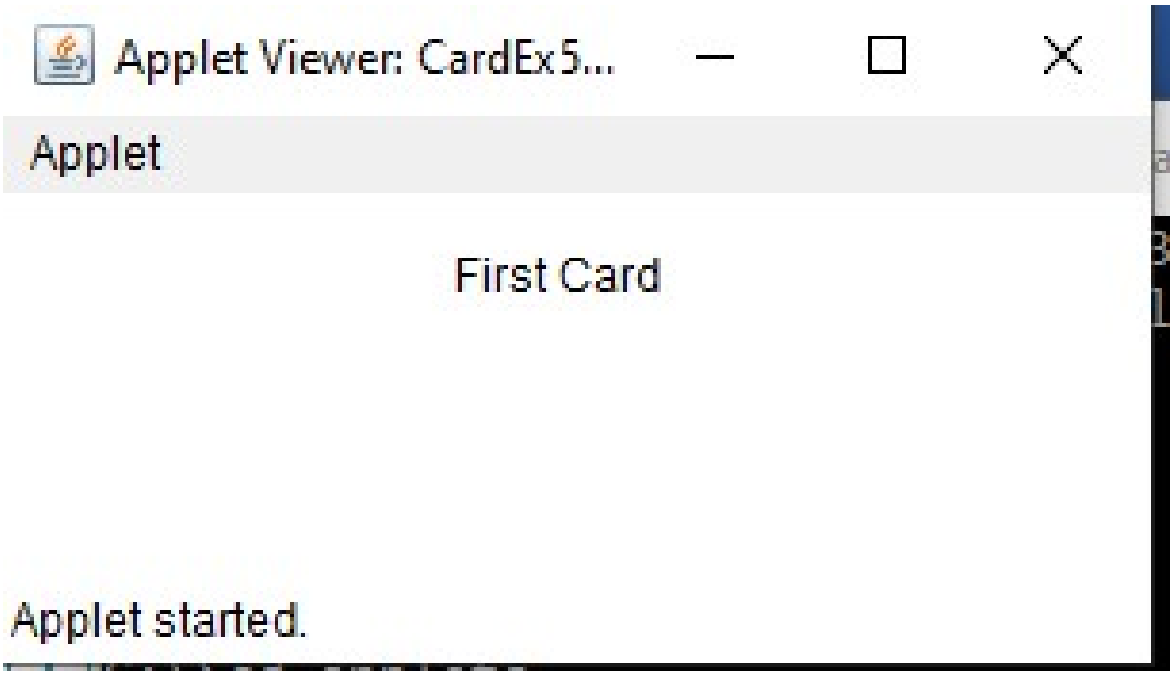


Program2:ImplementthefollowingstructureusingCardLayout




```
importjava.awt.event.*;
importjava.awt.*;
importjava.applet.*;
/*<appletcode="CardEx5.class"width=300height=100>
</applet>*/
publicclass CardEx5 extends Applet
        implementsMouseListener{ Labell1,l2,l3,l4,l5;
        Panel Cards;//Paneldefinition
        CardLayoutcardLO;
        public
            voidinit(){ cardLO=newCardLa
            yout(); Cards = new Panel();
            Cards.setLayout(cardLO);//setcardlayouttocarddeckPanel
            l1=new Label("FirstCard");
            l2=newLabel("SecondCard"); l3 =
            new Label("Third Card");l4 =new
            Label("Fourth Card");
            l5 = new Label("Fifth Card");

              // creating two Panels Panel
              p1=newPanel();
              p1.add(l1);
              Panelp2=newPanel();  p2.add(l2);
              Panelp3=newPanel();  p3.add(l3);
              Panelp4=newPanel(); p4.add(l4);
```

```
                Panelp5=newPanel(); p5.add(l5);

                //addpanelstocarddeckpanel
                Cards.add(p1, "First");
                Cards.add(p2,"Second");
                Cards.add(p3,"Third");
                Cards.add(p4,"Fourth");
                Cards.add(p5,"Fifth");

                //add carddecktomain appletwindow
                add(Cards);//registermouseevents
                addMouseListener(this);
    }
      //cyclethroughpanels
      public voidmousePressed(MouseEventme){
      }
      //Provideempty implementationsfor theotherMouseListenermethods.
      public
      voidmouseClicked(MouseEventme){ cardL
      O.next(Cards);
      }
      public
      voidmouseEntered(MouseEventme){ cardL
      O.last(Cards);
      }
      public
      voidmouseExited(MouseEventme){ cardLO.show(Cards,"Third")
      ;
      }
    public voidmouseReleased(MouseEventme){
```



5. GridBagLayout:

The key to the GridBag isthat eachcomponent can be adifferentsize, and each row in the grid can have a different no.of columns. That is why the layout is called a Grid Bag. It's a collection of small grids joined together.

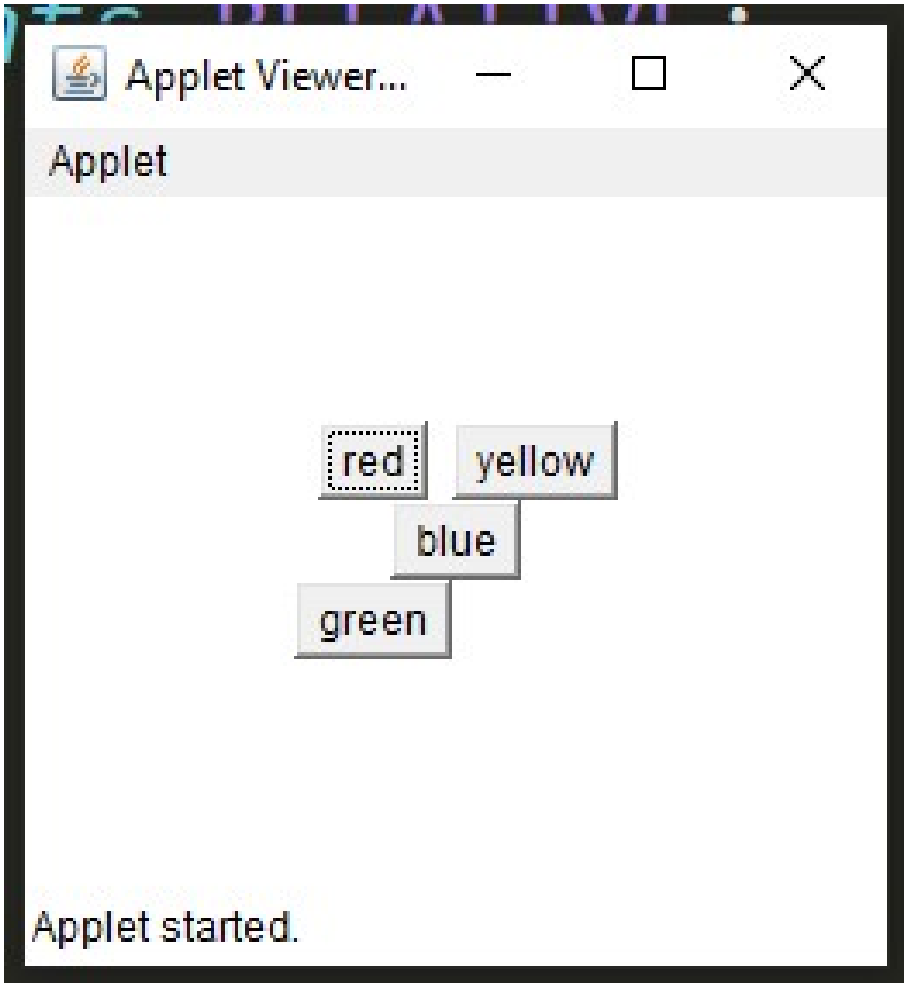
GridBagLayoutdefinesonlyone**Constuctor:** GridBabLayout()

Methods:

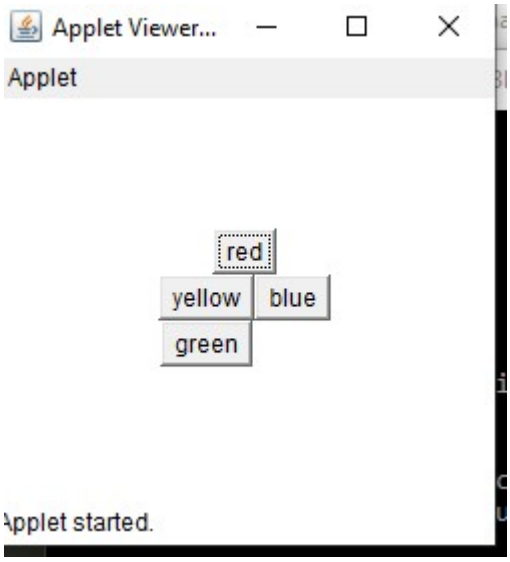
voidsetConstraints(Componentcomp,GridBagConstraintscons)

//DemostrationofGridBagLayout

```
import java.awt.event.*;
import java.awt.*;
import java.applet.*;
/*<appletcode="GBLEx6.class"width=250height=200>
</applet>*/
public class GBLEx6
    extendsApplet{ Buttonb1,b2,b3,b4;
public voidinit(){
        GridBagLayoutgbag=new GridBagLayout();
        GridBagConstraintsgbc=newGridBagConstraints();
        setLayout(gbag);
        b1=new Button("red");
        b2=newButton("yellow");
        b3=new Button("blue");
        b4=new
        Button("green");gbc.gridwidth=GridBagConstraints.
        RELATIVE;
    //RELATIVE means awtcontrolwilltakethespacerequired for itinarow
    gbag.setConstraints(b1,gbc);
    add(b1);
        gbc.gridwidth=GridBagConstraints.REMAINDER;
    //REMAINDERmeansawtcontrolwilloccupytheremainingspaceinarow
    gbag.setConstraints(b2,gbc);
    add(b2);
    //thegridwidthconstraintisnotupdatedforthebuttonb3sothegridwidthconstraint
ofb2 is applicable for
b3. gbag.setConstraints(b3,gbc);
add(b3);
        gbc.gridwidth=GridBagConstraints.RELATIVE;
        gbag.setConstraints(b4,gbc);
    add(b4);
    }
    }
```

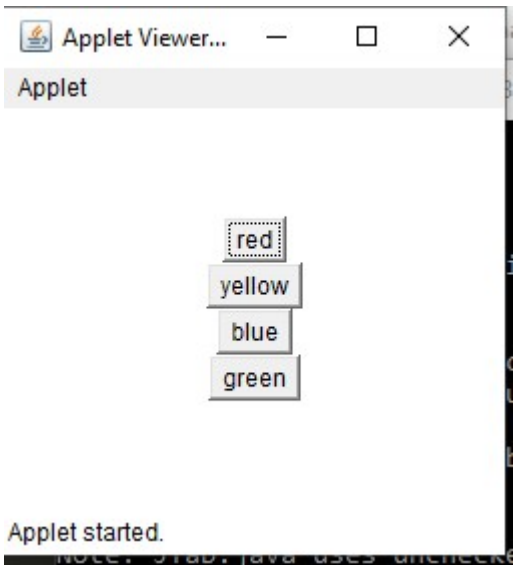


Program1:Writeanappletprogramtoobtainthefollowingoutput usingGridbagLayout



Hint:setgridwidthfor redremainder, yellowrelative,blueremainderandforgreen relative.

Program2:Writeanappletprogramtoobtainthefollowingoutput usingGridbagLayout



Hint:if wewant toarrange onebuttonin onelinethensetgridwidthtoremainder forred button. For the remaining buttons if we willnot change gridwidth then remainder willbe carried.

Swings

JavaSwingisusedtocreatewindow-basedapplicationsorgraphicaluserinterface. It is built on the topof AWT (Abstract WindowingToolkit) API andentirely **written in java**. Unlike AWT, Java Swing provides platform-independent and lightweight components. It is build on top of AWT API and acts as a replacement of AWT API, since ithas almostevery control corresponding to AWT controls. Swingcomponent follows a Model-View-Controller architecture, where **model** represents data, **view** represents presentation and **controller** acts as an interface between model and view

AWTcontrolslike Button, TextFieldandsoonare developedin differentlanguages that are not platform independent. Swing components are developed using Java. so they are written as JButton, JTextField and so on.

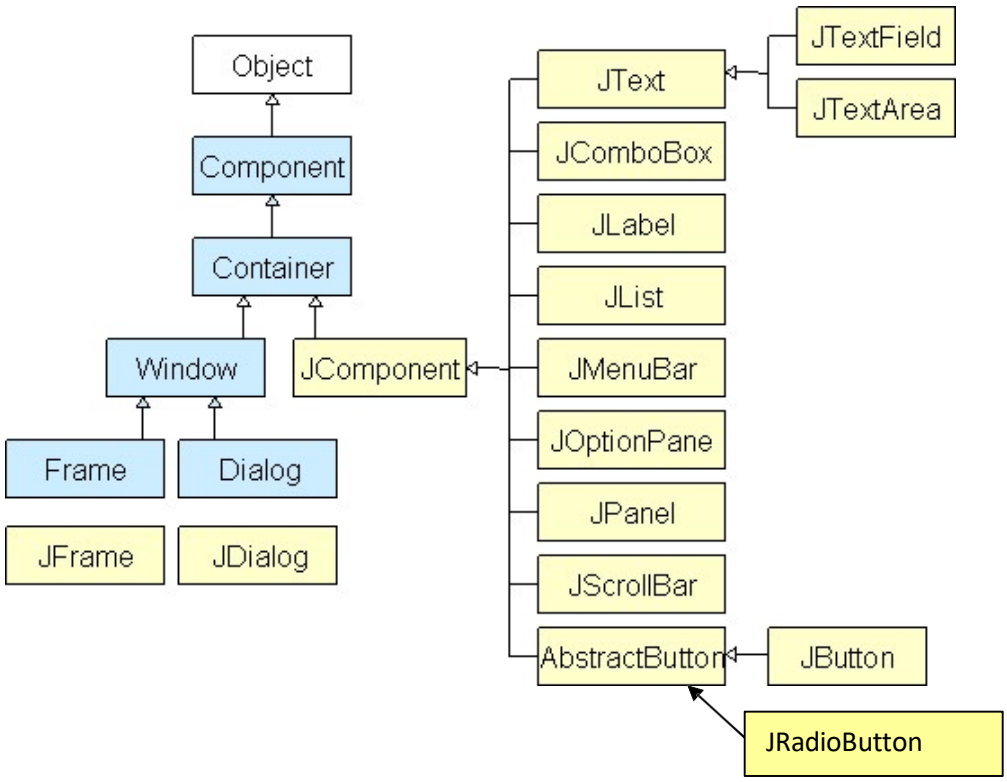
Thejavax.swingpackageprovidesclassesforjavaswingAPIsuchas JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu,etc.

DifferencebetweenAWTandSwing:-

No.	JavaAWT	Java Swing
1)	AWTcomponentsare **platform-dependent**	Javaswingcomponentsare **platform-independent**.
2)	AWTcomponents are **heavyweight**.	Swingcomponentsare**lightweight**.
3)	AWT **doesn't support pluggablelookandfeel**.	Swing**supportspluggablelookand feel**.
4)	AWT provides **less componentsthan**Swing.	Swing provides **more powerful componentssuch**astables,lists, scrollpanes, colorchooser, tabbedpaneetc.
5)	AWT**doesn'tfollows MVC**(ModelView Controller).	Swing**followsMVC**.

ExploringSwing:

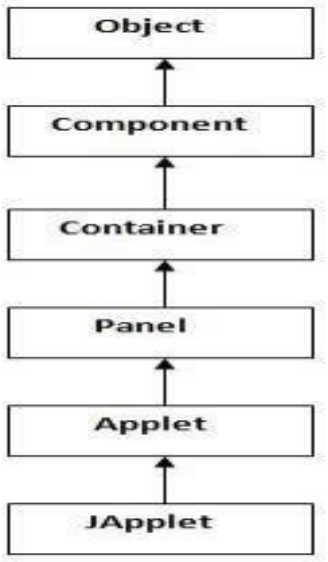
ThehierarchyofJAVASWINGclass



JApplet:-

JAppletis a class which extends **Applet** class. JApplet class has all the controls of swing. The JApplet class extends the Applet class. The javax.swing.**JApplet**class defines the core functionality of an **applet**. (**java**.awt.**Applet**is the older, AWT- based form)

HierarchyofApplet



JComponent

TheJComponentclass is the base class of all Swingcomponents except top-levelcontainers.Swingcomponentswhosenamesbeginwith"J"are descendants of the JComponent class. For example, JButton, JScrollPane, JPanel, JTable etc. But, JFrame and JDialog don't inherit JComponent class because they are the child of top-level containers.

The JComponent class extends the Container class which itself extends Component. The Container class has support for addingcomponents to the container.

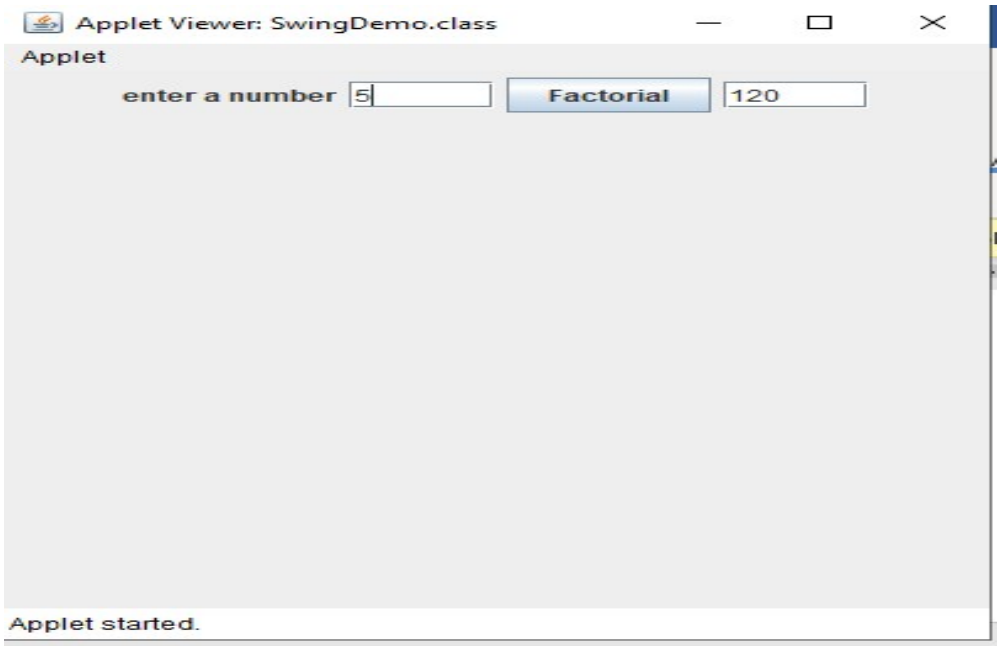
Constructor:

JComponent()

//JavaSwingProgram

```
importjavax.swing.*;
importjava.awt.*;
importjava.awt.event.*;

/*<appletcode="SwingDemo.class"height=400width=400></applet>*/

public

        classSwingDemoextendsJAppletimplementsActionListener{
```

```
        JLabel
        l1;JTextField t1,t2;

        public voidinit(){

            // These two statements are to be added for all swing programs

            Container c=getContentPane();

            c.setLayout(new FlowLayout());// here FlowLayout is not default layout

            b1=newJButton("Factorial");
            t1=new JTextField("",5);
            t2=new JTextField("",5);
            l1=newJLabel("enteranumber");

            // instead of adding awt controls to Applet Window in Applet programs,
            //for swing programs we have to add to container

            c.add(l1);
            c.add(t1);
            c.add(b1);
            c.add(t2);

            b1.addActionListener(this);

}

public

voidactionPerformed(ActionEventae){ Stringstr=t

1.getText(); intnum=Integer.parseInt(str);

int fact=1;

for(inti=1;i<=num;i++){ act=fac

        t*i;

}

t2.setText(fact+"");

}

}
```



2. RadioButton:

Constructor:

JRadioButton(Stringstr)

Method:

voidadd(AbstractButtonab)

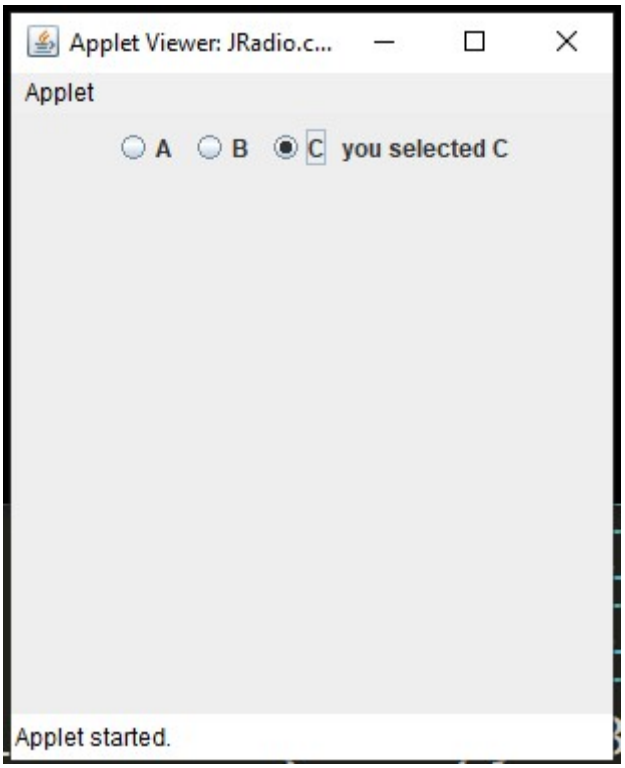
Note: AftercreatingtheJRadioButtonweneedtoGrouptheseButtons.For this we use ButtonGroup class

ButtonGroup()

//ProgramonRadiobuttonsusingjavaswing

```
importjavax.swing.*;
importjava.awt.event.*;
importjava.awt.*;

/*<appletcode="JRadioButtonDemo.class"height=300width=300>
</applet>*/

publicclassJRadioButtonDemoextendsJAppletimplementsActionListener{ JL
abeljl;
```

```
public voidinit(){
        Container c=getContentPane();
        c.setLayout(new FlowLayout());

        JRadioButtonb1=newJRadioButton("A");
        JRadioButtonb2=newJRadioButton("B");
        JRadioButtonb3=newJRadioButton("C");

 ButtonGroupbg=new ButtonGroup(); //adding JRadioButtons to ButtonGroup

 bg.add(b1);
 bg.add(b2);
        bg.add(b3);

        b1.addActionListener(this);
        b2.addActionListener(this);
        b3.addActionListener(this);

        c.add(b1);
        c.add(b2);
        c.add(b3);
        jl=newJLabel("selectone");
        c.add(jl);

 }
   public
   voidactionPerformed(ActionEventae){ jl.setText("youselected"+ae.getActionCommand());
   }
}
```



3. JComboBox://inAppletsitisChoice

JComboBox:

SwingprovidesacomboboxthroughtheJComboBoxclass.Acombobox normally displaysoneentry,butitwillalsodisplayadrop-downlistthat allows a user to select a different entry.

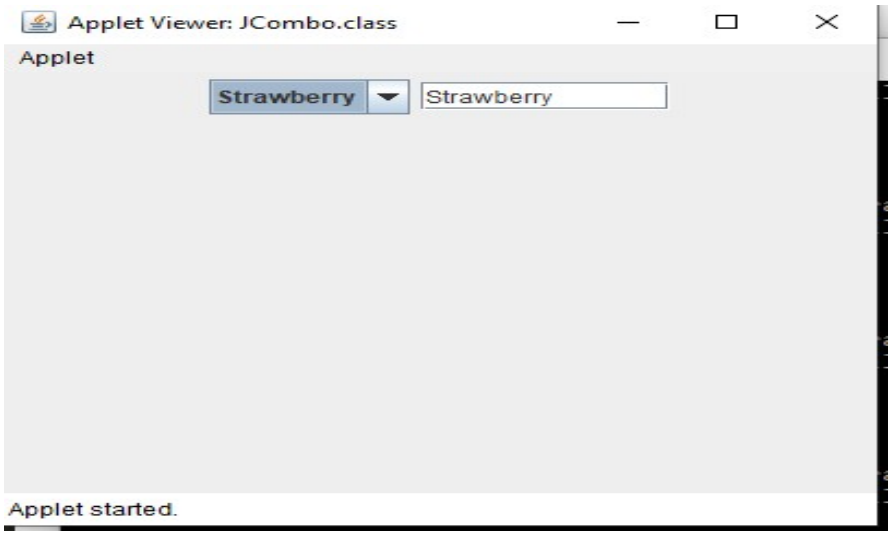
Constructor:

JComboBox(Object[]items)

Methods:
voidaddItem(Objectobj)
ObjectgetSelectedItem()

//DemonstrationonJCombobox

```
importjavax.swing.*;
importjava.awt.*;
importjava.awt.event.*;

/*<appletcode="JCombo.class"height=300width=400></applet>*/
publicclassJComboextendsJAppletimplementsItemListener{ JCombo
        Boxjcb;
        JTextFieldtf;

        public voidinit(){
            Container c=getContentPane();
            c.setLayout(newFlowLayout());

            jcb=new JComboBox();
            jcb.addItem("Vanilla");
            jcb.addItem("Choclate");
            jcb.addItem("Strawberry");

            c.add(jcb);
            tf=newJTextField("",10);
            c.add(tf);
            jcb.addItemListener(this);
        }
public
voiditemStateChanged(ItemEventie){ tf.setText(ie.getItem()+"");
        }
}
```



4. JTabbedPane:

JTabbedPaneencapsulatesatabbedpane.JTabbedPane defines by using below constructor.

JTabbedPane()

Methods:

voidaddTab(Stringname,Componentcomp)

Procedure:

Thegeneralproceduretouseatabbedpaneisoutlinedhere:

1) createaninstanceofJTabbedPane.

2) Addeach tabbycallingaddTab()

3) Addthetabbedpanetothecontentpane.

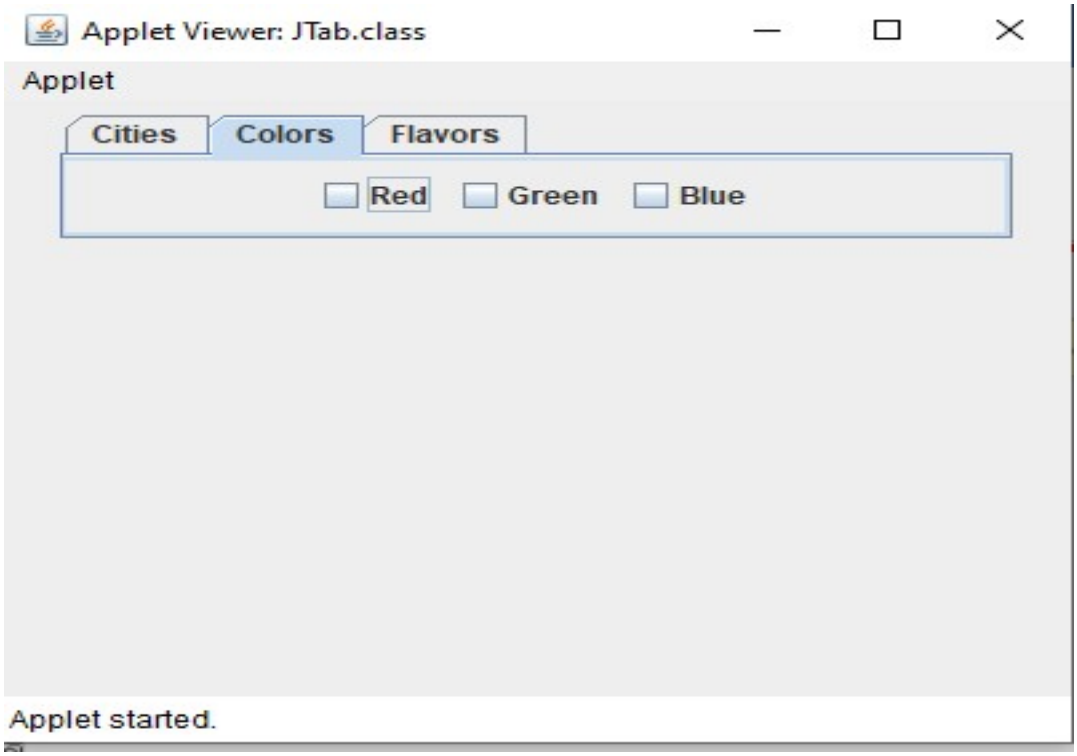
//DemonstrationofJTabbedPane

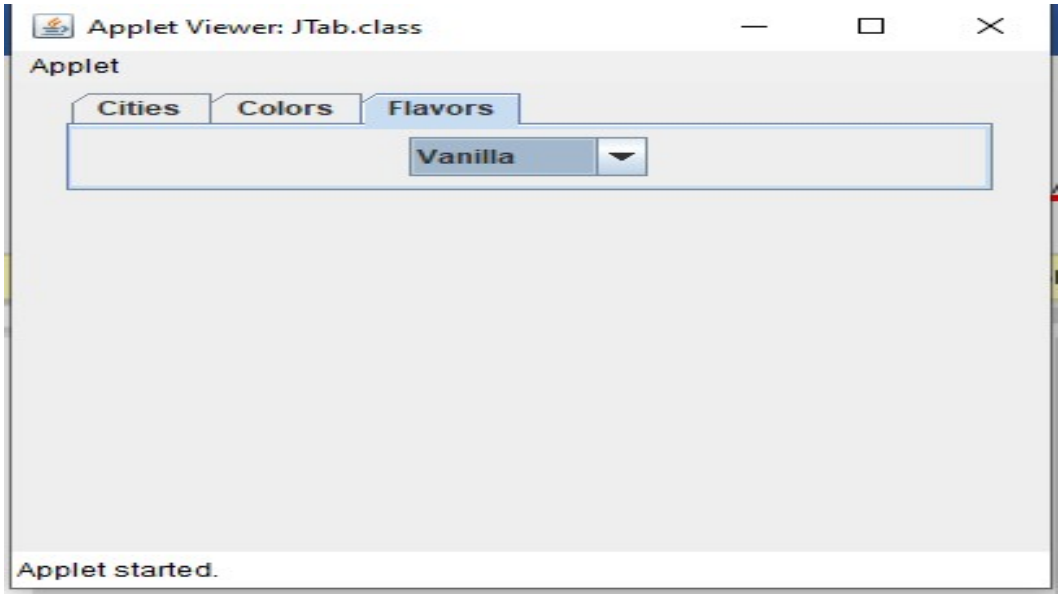
```
importjavax.swing.*;
importjava.awt.*;

/*<appletcode="JTabbedPaneDemo.class"height=300width=400>
</applet>*/

publicclassJTabbedPaneDemoextendsJApplet{ publicv
oidinit(){
        Container c=getContentPane();
```

```
                c.setLayout(new FlowLayout());
                JTabbedPanejtp=new JTabbedPane();

                jtp.addTab("Cities", new CitiesPanel());
                //whentheaddTab()methodisinvokeditgoes  toCitiesPanel()constructor
                //andtheCities NewYork,London,so onareaddedto theCities Tab jtp.addTab("Colors",
                new ColorsPanel());
                jtp.addTab("Flavors",newFlavorsPanel());

                c.add(jtp);
        }
 }
 classCitiesPanelextendsJPanel{ public
CitiesPanel(){
        JButtonb1=newJButton("NewYork");
        add(b1);
        JButtonb2=newJButton("London");
        add(b2);
        JButtonb3=newJButton("HongKong");
        add(b3);
        JButtonb4=newJButton("Tokyo");
        add(b4);
  }
 }
 classColorsPanelextendsJPanel{ public
 ColorsPanel(){
            JCheckBox  cb1=new  JCheckBox("Red");
            JCheckBoxcb2=newCheckBox("Green");
            JCheckBox cb3=new JCheckBox("Blue");

            add(cb1);
            add(cb2);
            add(cb3);
      }
 }
 classFlavorsPanelextendsJPanel{ publi
 cFlavorsPanel(){
            JComboBoxjcb=newJComboBox();

            jcb.addItem("Vanilla");
            jcb.addItem("Choclate");
            jcb.addItem("Strawberry");
            add(jcb);
    }
```

```
}
```


Applet Viewer: JTab.class
Applet
Cities
Colors
Flavors
New York
London
Hong Kong
Tokyo
Applet started.


Applet Viewer: JTab.class
Applet
Cities
Colors
Flavors
Red
Green
Blue
Applet started.



JScrollPane:

Constructor:

JScrollPane(Componentcomp)

Procedure

1. Createthecomponenttobescrolled.(ex:-Panel)
2. Create aninstanceofJScrollPane, passthecomponent(Panel)tobe scrolled.
3. Addthescrollpanetothecontentpane (Container).

//DemonstrationofJSCrollPane

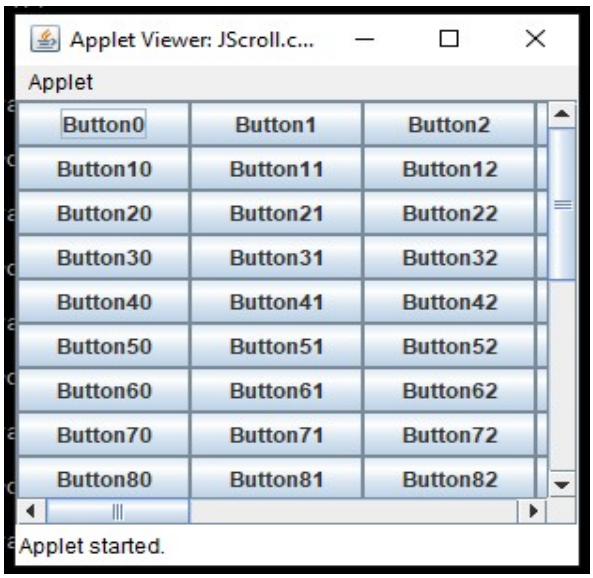
```
importjavax.swing.*;
importjava.awt.*;

 /*<appletcode="JScrollPaneDemo.class"height=250 width=300>

 </applet>*/

publicclassJScrollPaneDemoextendsJApplet{//CreatinganApplet

publicvoidinit(){

        Container c=getContentPane();

        c.setLayout(newBorderLayout());

        JPaneljp=new JPanel();
```

```
            //1. Create the componenttobescrolled. jp.setLayout(new GridLayout(20,20));
for(inti=0;i<200;i++){
            jp.add(newJButton("Button"+i));
}
   //2.CreateaninstanceofJScrollPane,passthecomponenttobescrolled.
    JScrollPanejsp=new JScrollPane(jp);//
   //3.Add thescrollpanetothecontentpane.
    c.add(jsp,BorderLayout.CENTER);
  }
  }
```

Note 1: Create the JPanel add the Buttons to the JPanel. Add the JScrollPane to the JPanel. Then add the JScrollPane to the ContentPane.

Note 2: If you add less number of Buttons to the JPanel then the Scroll Bars are not visible evenifyoucreate JScrollPane.So, addmore number of Buttons to visualize the JScrollPane.

Note3: VerticalScrollBarisautomaticallycreatedifweaddmorenumber of rows and horizontalScroolBar is automaticallycreated if we addmore number of columns



JTree

A tree is a component that presents a hierarchical view of data.Trees are implemented in swing by the JTree class.

Constructors:

JTree(TreeNodetn)

TheTreenodeinterfacedeclaresmethodsthatobtaininformationabouta tree node.

The mutable TreeNode interface extends TreeNode. It declares methods that can insert and remove child nodes or change the parent node. The DefaultMutableTreeNodeclassimplementstheMutableTreeNodeinterface. It represents a node in a tree.

oneofitsconstructorsisshownhere: DefaultMutableTreeNode(Object obj)
Tocreate ahierarchy oftreenodes,theadd()method of DefaultMutableTreeNode can be used.

Itssignatureisshownhere:

voidadd(MutableTreeNodechild)

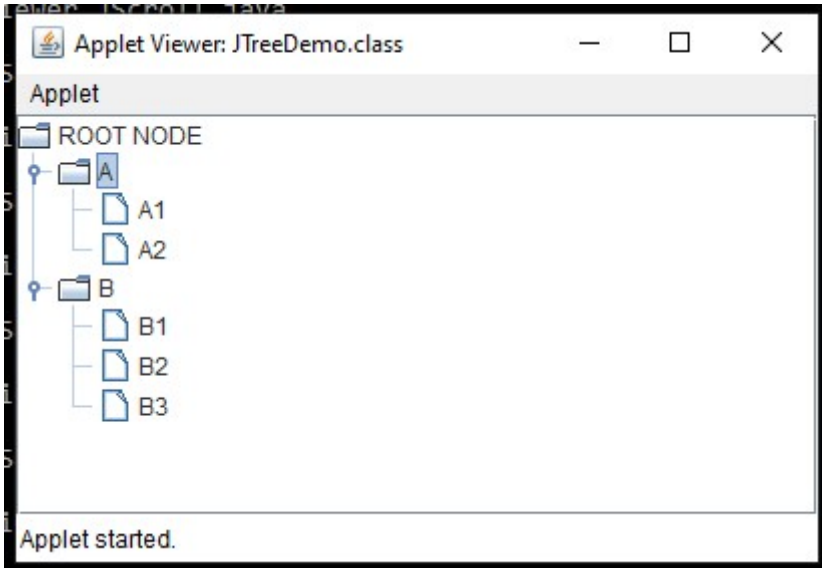
herearethestepstofollowtouseofJTree.

1. Createaninstanceof JTree.
2. CreateJScrollPaneandpassthetreeas theobjecttobescrolled. (AddJScrollpane to the Tree)
3. AddtheJScrollpanetothecontentpane.

//DemonstrationonTrees

```
importjavax.swing.*;
importjava.awt.*;
importjavax.swing.tree.*;

/*<appletcode="JTreeDemo.class"height=200 width=400>
</applet>*/

publicclassJTreeDemoextendsJApplet{
 publicvoidinit(){

        Container c=getContentPane();
        c.setLayout(newBorderLayout());
```

```
        DefaultMutableTreeNodetop=newDefaultMutableTreeNode("ROOT
        NODE");
        DefaultMutableTreeNodea=newDefaultMutableTreeNode("A");
        top.add(a);
        DefaultMutableTreeNodea1=newDefaultMutableTreeNode("A1");
        DefaultMutableTreeNodea2=newDefaultMutableTreeNode("A2");

        a.add(a1);
        a.add(a2);
        DefaultMutableTreeNodeb=newDefaultMutableTreeNode("B");
        top.add(b);
        DefaultMutableTreeNodeb1=newDefaultMutableTreeNode("B1");
        DefaultMutableTreeNodeb2=newDefaultMutableTreeNode("B2");
        DefaultMutableTreeNodeb3=newDefaultMutableTreeNode("B3");
        b.add(b1);
        b.add(b2);
        b.add(b3);

        //addtherootnodetotheJTree

        JTree tree=new JTree(top); // 1. Create an instance of JTree.

        JScrollPanejsp=newJScrollPane(tree);//2.Addingscrollpanetotree

        c.add(jsp, BorderLayout.NORTH); //3. adding scroll pane to container
}

}
```



Note:IfucreatemorenumberofnodesthentheJScrollPaneisvisible

JTable

JTableis acomponentthatdisplaysrows andcolumns of data.JTable supplies several constructors.

Constructor:

JTable(Objectdata[][],ObjectcolHead[])

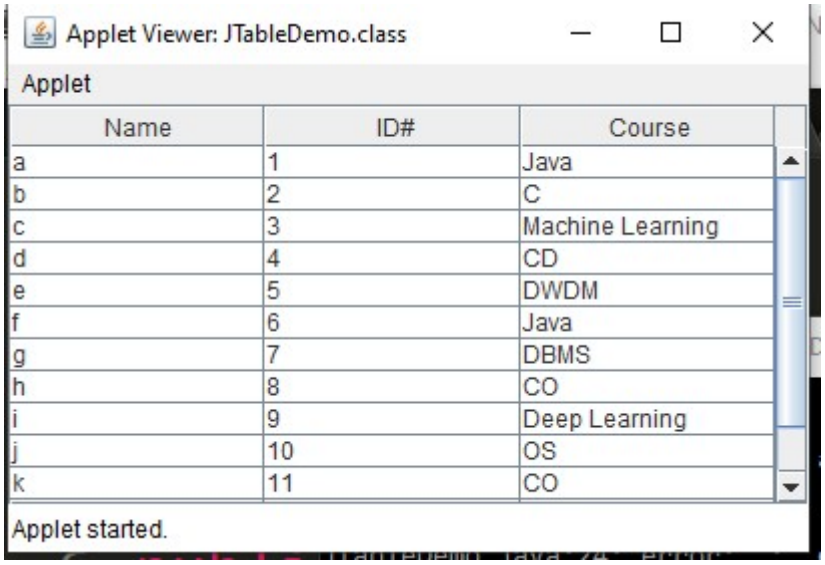
Here are thestepsrequired tosetupa simpleJTable thatcanbeused to display the data.

1. Createaninstanceof JTable.
2. CreateJScrollPaneand passthetableas the objecttobescrolled.(Add the table to the scrollpane)
3. Addthescrollpanetothecontent pane.

//DemonstrationonJTable

```
importjavax.swing.*;
importjava.awt.*;

/*<appletcode="JTableDemo.class" height=200width=400></applet>*/

publicclassJTableDemoextendsJApplet{ publicvoidinit(

){

        Container c=getContentPane();
        c.setLayout(newBorderLayout());
```

```
        String[]colHeads={"Name","ID#","Course"};
      Object[][] data={
      {"a","1","Java"},
      {"b","2","C"},
      {"c","3","MachineLearning"},
      {"d","4","CD"},
      {"e","5","DWDM"},
      {"f","6","Java"},
      {"g","7","DBMS"},
      {"h","8","CO"},
      {"i","9","Deep Learning"},
      {"j","10","OS"},
      {"k","11","CO"},
      {"l","12","FLAT"},
      {"m","13","DAA"}
};
      JTabletable=newJTable(data,colHeads);//1.Createaninstanceof JTable.
      JScrollPanejsp=new JScrollPane(table);//2.Add thetabletothe scrollpane
      c.add(jsp,BorderLayout.CENTER);//3. Addthescrollpane tothecontent pane.
  }
 }
```



Applet	**JApplet**
Label	JLabel
Button	JButton
TextField	JTextField
TextArea	JTextArea
Checkbox	JCheckbox
CheckboxGroup	JButtonGroup
Choice	JCombobox
List	JList
	JTabbedPane
	JScrollPane
	JTable
	JTree

JFrame

Thejavax.swing.JFrameclass is atypeofcontainerwhichinherits the java.awt.Frame class. JFrame works like the main window where components like labels, buttons, textfields are added to create a GUI.

Unlike Frame, JFrame has the option to hide or close the window with the help of setDefaultCloseOperation(int) method.

Constructors

Constructor	Description
JFrame()	Itconstructsanewframethatisinitially invisible.
JFrame(GraphicsConfigurationgc)	It creates a Frame in the specified GraphicsConfigurationofascreen deviceandablanktitle.
JFrame(Stringtitle)	Itcreatesanew,initiallyinvisibleFrame with the specified title.
JFrame(String title, GraphicsConfigurationgc)	It createsaJFrame withthespecified title and the specified GraphicsConfiguration of ascreen device.

Note:Withoutcreatingapplettagwecancreateawindowwhich is a Frame.

JFrame

```
importjava.awt.*;
importjavax.swing.*;

publicclassFrameEx11{ FrameEx11(

){

        JFramefm=newJFrame("JFrameExample");

        //CreatingaframewithTitle.

        JLabellb=newJLabel("welcome tojavaFrameWindow ");

        //Creatingalabel

        fm.setLayout(newFlowLayout(FlowLayout.CENTER));
        fm.add(lb);          //adding label to the frame.
fm.setSize(300, 300);//setting frame size.
```

```
fm.setVisible(true);//set frame visibilty true.


fm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 //For closing the Frame

 }
 publicstaticvoidmain(String

    args[]){ FrameEx11ta=new

    FrameEx11();

}
```

Compilation:javacFrameEx11.java

Exectuion: java FrameEx11