Event handling

What is an Event?

Change in the state of an object is known as event i.e. event describes the change in state of source. Events are generated as result of user interaction with the graphical user interface components. For example, clicking on a button, moving the mouse, entering a character through keyboard,selecting an item from list, scrolling the page are the activities that causes an event to happen.

Types of Event

The events can be broadly classified into two categories:

- **Foreground Events** - Those events which require the direct interaction of user.They are generated as consequences of a person interacting with the graphical components in Graphical User Interface. For example, clicking on a button, moving the mouse, entering a character through keyboard,selecting an item from list, scrolling the page etc.
- **Background Events** - Those events that require the interaction of end user are known as background events. Operating system interrupts, hardware or software failure, timer expires, an operation completion are the example of background events.

What is Event Handling?

Event Handling is the mechanism that controls the event and decides what should happen if an event occurs. This mechanism have the code which is known as event handler that is executed when an event occurs. Java Uses the Delegation Event Model to handle the events. This model defines the standard mechanism to generate and handle the events.Let's have a brief introduction to this model.

The Delegation Event Model has the following key participants namely:

- **Source** - The source is an object on which event occurs. Source is responsible for providing information of the occurred event to it's handler. Java provide as with classes for source object.
- **Listener** - It is also known as event handler. Listener is responsible for generating response to an event. From java implementation point of view the listener is also an object. Listener waits until it receives an event. Once the event is received , the listener process the event an then returns.

The benefit of this approach is that the user interface logic is completely separated from the logic that generates the event. The user interface element is able to delegate the processing of an event to the separate piece of code. In this model ,Listener needs to be registered with the source object so that the listener can receive the event notification. This is an efficient way of handling the event because the event notifications are sent only to those listener that want to receive them.

Steps involved in event handling

- The User clicks the button and the event is generated.
- Now the object of concerned event class is created automatically and information about the source and the event get populated with in same object.
- Event object is forwarded to the method of registered listener class.
- the method is now get executed and returns.

Points to remember about listener

- In order to design a listener class we have to develop some listener interfaces.These Listener interfaces forecast some public abstract callback methods which must be implemented by the listener class.

- If you do not implement the any if the predefined interfaces then your class can not act as a listener class for a source object.

Callback Methods

These are the methods that are provided by API provider and are defined by the application programmer and invoked by the application developer. Here the callback methods represents an event method. In response to an event java jre will fire callback method. All such callback methods are provided in listener interfaces.

If a component wants some listener will listen to it's events the the source must register itself to the listener.

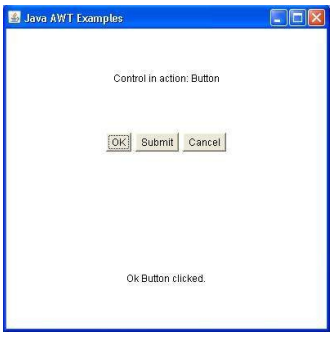
Event Handling Example

AwtControlDemo.java

```
package com.tutorialspoint.gui;

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


public class AwtControlDemo {
// declaring components
   private Frame mainFrame;
   private Label headerLabel;
   private Label statusLabel;
   private Panel controlPanel;

   public AwtControlDemo(){
      prepareGUI();
   }

   public static void main(String[] args){
      AwtControlDemo  awtControlDemo = new AwtControlDemo();
      awtControlDemo.showEventDemo();
   }
// creating components
   private void prepareGUI(){
      mainFrame = new Frame("Java AWT Examples");
      mainFrame.setSize(400,400);
      mainFrame.setLayout(new GridLayout(3, 1));
      mainFrame.addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent windowEvent){
            System.exit(0);
         }
      });
      headerLabel = new Label();
      headerLabel.setAlignment(Label.CENTER);
      statusLabel = new Label();
      statusLabel.setAlignment(Label.CENTER);
      statusLabel.setSize(350,100);

      controlPanel = new Panel();
      controlPanel.setLayout(new FlowLayout());

      mainFrame.add(headerLabel);
      mainFrame.add(controlPanel);
```

```
      mainFrame.add(statusLabel);
      mainFrame.setVisible(true);
   }
// showing the components which were declared and created
   private void showEventDemo(){
      headerLabel.setText("Control in action: Button");

      Button okButton = new Button("OK");
      Button submitButton = new Button("Submit");
      Button cancelButton = new Button("Cancel");

      okButton.setActionCommand("OK");
      submitButton.setActionCommand("Submit");
      cancelButton.setActionCommand("Cancel");

      okButton.addActionListener(new ButtonClickListener());
      submitButton.addActionListener(new ButtonClickListener());
      cancelButton.addActionListener(new ButtonClickListener());

      controlPanel.add(okButton);
      controlPanel.add(submitButton);
      controlPanel.add(cancelButton);

      mainFrame.setVisible(true);
   }

// it is defined for activating ActionListener interface
   private class ButtonClickListener implements ActionListener{
      public void actionPerformed(ActionEvent e) {
         String command = e.getActionCommand();
         if( command.equals( "OK" ))  {
            statusLabel.setText("Ok Button clicked.");
         }
         else if( command.equals( "Submit" ) )  {
            statusLabel.setText("Submit Button clicked.");
         }
         else  {
            statusLabel.setText("Cancel Button clicked.");
         }
      }
   }
}
```

Verify the following output



Event and Listener (Java Event Handling)

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

Java Event classes and Listener interfaces

Event Classes	Listener Interfaces
ActionEvent	ActionListener
MouseEvent	MouseListener and MouseMotionListener
MouseWheelEvent	MouseWheelListener
KeyEvent	KeyListener
ItemEvent	ItemListener
TextEvent	TextListener
AdjustmentEvent	AdjustmentListener
WindowEvent	WindowListener
ComponentEvent	ComponentListener
ContainerEvent	ContainerListener
FocusEvent	FocusListener

Steps to perform Event Handling

Following steps are required to perform event handling:

1. Register the component with the Listener

Registration Methods

For registering the component with the Listener, many classes provide the registration methods. For example:

- **Button**
  - public void addActionListener(ActionListener a){}
- **MenuItem**
  - public void addActionListener(ActionListener a){}
- **TextField**
  - public void addActionListener(ActionListener a){}
  - public void addTextListener(TextListener a){}
- **TextArea**
  - public void addTextListener(TextListener a){}
- **Checkbox**
  - public void addItemListener(ItemListener a){}
- **Choice**
  - public void addItemListener(ItemListener a){}
- **List**
  - public void addActionListener(ActionListener a){}
  - public void addItemListener(ItemListener a){}

Java Event Handling Code

We can put the event handling code into one of the following places:

1. Within class
2. Other class

1 java event handling by implementing ActionListener

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

class AEvent extends Frame implements ActionListener{
TextField tf;

AEvent(){

//create components
tf=new TextField();
tf.setBounds(60,50,170,20);
Button b=new Button("click me");
b.setBounds(100,120,80,30);

//register listener
b.addActionListener(this);//passing current instance

//add components and set size, layout and visibility
add(b);add(tf);
setSize(300,300);
setLayout(null);
setVisible(true);
}
public void actionPerformed(ActionEvent e){
tf.setText("Welcome");
}
public static void main(String args[]){
new AEvent();
}
}
```

public void setBounds(int xaxis, int yaxis, int width, int height); have been used in the above example that sets the position of the component it may be button, textfield etc.

2) Java event handling by outer class

```
import java.awt.*;
import java.awt.event.*;
class AEvent2 extends Frame{
TextField tf;
AEvent2(){
//create components
tf=new TextField();
tf.setBounds(60,50,170,20);
```

```
Button b=new Button("click me");
b.setBounds(100,120,80,30);
//register listener
Outer o=new Outer(this);
b.addActionListener(o);//passing outer class instance
//add components and set size, layout and visibility
add(b);add(tf);
setSize(300,300);
setLayout(null);
setVisible(true);
}
public static void main(String args[]){
new AEvent2();
}
}
import java.awt.event.*;
class Outer implements ActionListener{
AEvent2 obj;
Outer(AEvent2 obj){
this.obj=obj;
}
public void actionPerformed(ActionEvent e){
obj.tf.setText("welcome");
}
}
```

Java Adapter Classes

Java adapter classes *provide the default implementation of listener interfaces*. If you inherit the adapter class, you will not be forced to provide the implementation of all the methods of listener interfaces. So it *saves code*.

The adapter classes are found in **java.awt.event**, **java.awt.dnd** and **javax.swing.event** packages. The Adapter classes with their corresponding listener interfaces are given below.

java.awt.event Adapter classes

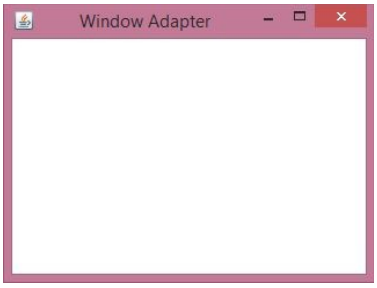
Adapter class	Listener interface
WindowAdapter	WindowListener
KeyAdapter	KeyListener
MouseAdapter	MouseListener
MouseMotionAdapter	MouseMotionListener
FocusAdapter	FocusListener
ComponentAdapter	ComponentListener
ContainerAdapter	ContainerListener
HierarchyBoundsAdapter	HierarchyBoundsListener

Java WindowAdapter Example

```
import java.awt.*;
import java.awt.event.*;
public class AdapterExample{
   Frame f;
   AdapterExample(){
```

```
        f=new Frame("Window Adapter");
        f.addWindowListener(new WindowAdapter(){
           public void windowClosing(WindowEvent e) {
              f.dispose();
           }
        });

        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
  public static void main(String[] args) {
    new AdapterExample();
  }
  }
```

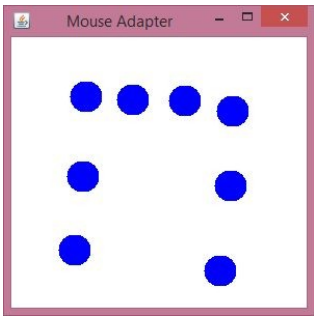
Output:



Java MouseAdapter Example

```
  import java.awt.*;
  import java.awt.event.*;
  public class MouseAdapterExample extends MouseAdapter{
    Frame f;
    MouseAdapterExample(){
        f=new Frame("Mouse Adapter");
        f.addMouseListener(this);

        f.setSize(300,300);
        f.setLayout(null);
        f.setVisible(true);
    }
    public void mouseClicked(MouseEvent e) {
        Graphics g=f.getGraphics();
        g.setColor(Color.BLUE);
        g.fillOval(e.getX(),e.getY(),30,30);
    }

  public static void main(String[] args) {
    new MouseAdapterExample();
  }
  }
```

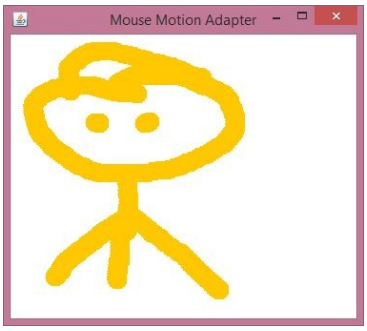
Output:



Java MouseMotionAdapter Example

```
import java.awt.*;
import java.awt.event.*;
public class MouseMotionAdapterExample extends MouseMotionAdapter{
   Frame f;
   MouseMotionAdapterExample(){
      f=new Frame("Mouse Motion Adapter");
      f.addMouseMotionListener(this);

      f.setSize(300,300);
      f.setLayout(null);
      f.setVisible(true);
   }
public void mouseDragged(MouseEvent e) {
   Graphics g=f.getGraphics();
   g.setColor(Color.ORANGE);
   g.fillOval(e.getX(),e.getY(),20,20);
}
public static void main(String[] args) {
   new MouseMotionAdapterExample();
}
}
```

Output:

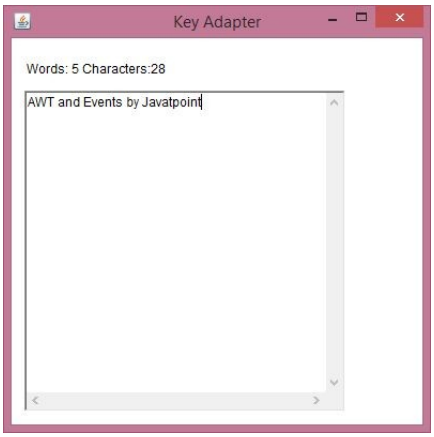


Java KeyAdapter Example

```
import java.awt.*;
import java.awt.event.*;
public class KeyAdapterExample extends KeyAdapter{
   Label l;
   TextArea area;
   Frame f;
   KeyAdapterExample(){
      f=new Frame("Key Adapter");
      l=new Label();
      l.setBounds(20,50,200,20);
```

```
        area=new TextArea();
        area.setBounds(20,80,300, 300);
        area.addKeyListener(this);

        f.add(l);f.add(area);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
    public void keyReleased(KeyEvent e) {
        String text=area.getText();
        String words[]=text.split("\\s");
        l.setText("Words: "+words.length+" Characters:"+text.length());
    }

    public static void main(String[] args) {
        new KeyAdapterExample();
    }
}
```

Output:



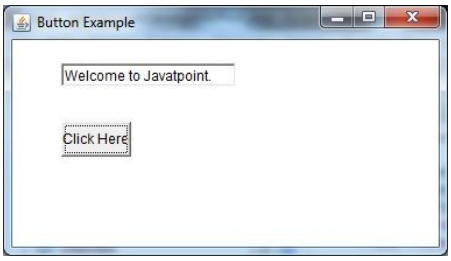
Content beyond syllabus

Java AWT Button Example with ActionListener

```
import java.awt.*;
import java.awt.event.*;
public class ButtonExample {
public static void main(String[] args) {
    Frame f=new Frame("Button Example");
    final TextField tf=new TextField();
    tf.setBounds(50,50, 150,20);
    Button b=new Button("Click Here");
    b.setBounds(50,100,60,30);
    b.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
            tf.setText("Welcome to AIET.");
        }
    });
    f.add(b);f.add(tf);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
}
```

}

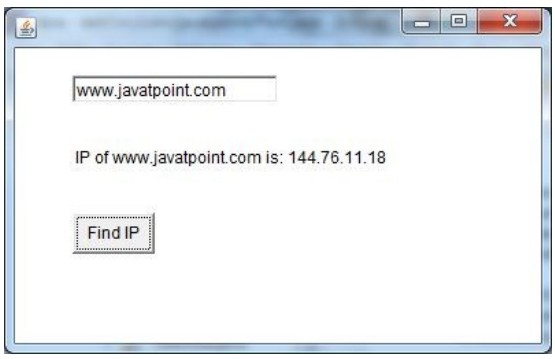
Output:



Java AWT Label Example with ActionListener

```
import java.awt.*;
import java.awt.event.*;
public class LabelExample extends Frame implements ActionListener{
    TextField tf; Label l; Button b;
    LabelExample(){
        tf=new TextField();
        tf.setBounds(50,50, 150,20);
        l=new Label();
        l.setBounds(50,100, 250,20);
        b=new Button("Find IP");
        b.setBounds(50,150,60,30);
        b.addActionListener(this);
        add(b);add(tf);add(l);
        setSize(400,400);
        setLayout(null);
        setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        try{
        String host=tf.getText();
        String ip=java.net.InetAddress.getByName(host).getHostAddress();
        l.setText("IP of "+host+" is: "+ip);
        }catch(Exception ex){System.out.println(ex);}
    }
    public static void main(String[] args) {
        new LabelExample();
    }
}
```

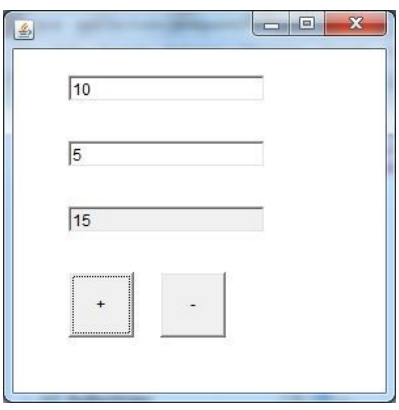
Output:



Java AWT TextField Example with ActionListener

```
import java.awt.*;
import java.awt.event.*;
public class TextFieldExample extends Frame implements ActionListener{
    TextField tf1,tf2,tf3;
    Button b1,b2;
    TextFieldExample(){
        tf1=new TextField();
        tf1.setBounds(50,50,150,20);
        tf2=new TextField();
        tf2.setBounds(50,100,150,20);
        tf3=new TextField();
        tf3.setBounds(50,150,150,20);
        tf3.setEditable(false);
        b1=new Button("+");
        b1.setBounds(50,200,50,50);
        b2=new Button("-");
        b2.setBounds(120,200,50,50);
        b1.addActionListener(this);
        b2.addActionListener(this);
        add(tf1);add(tf2);add(tf3);add(b1);add(b2);
        setSize(300,300);
        setLayout(null);
        setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        String s1=tf1.getText();
        String s2=tf2.getText();
        int a=Integer.parseInt(s1);
        int b=Integer.parseInt(s2);
        int c=0;
        if(e.getSource()==b1){
            c=a+b;
        }else if(e.getSource()==b2){
            c=a-b;
        }
        String result=String.valueOf(c);
        tf3.setText(result);
    }
public static void main(String[] args) {
    new TextFieldExample();
}
}
```

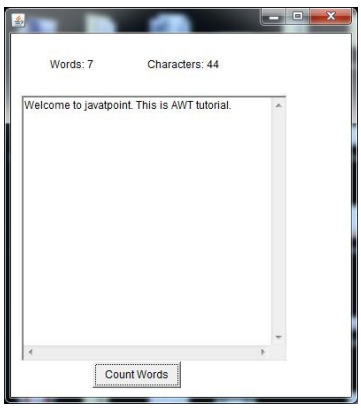
Output:



Java AWT TextArea Example with ActionListener

```
import java.awt.*;
import java.awt.event.*;
public class TextAreaExample extends Frame implements ActionListener{
Label l1,l2;
TextArea area;
Button b;
TextAreaExample(){
   l1=new Label();
   l1.setBounds(50,50,100,30);
   l2=new Label();
   l2.setBounds(160,50,100,30);
   area=new TextArea();
   area.setBounds(20,100,300,300);
   b=new Button("Count Words");
   b.setBounds(100,400,100,30);
   b.addActionListener(this);
   add(l1);add(l2);add(area);add(b);
   setSize(400,450);
   setLayout(null);
   setVisible(true);
}
public void actionPerformed(ActionEvent e){
   String text=area.getText();
   String words[]=text.split("\\s");
   l1.setText("Words: "+words.length);
   l2.setText("Characters: "+text.length());
}
public static void main(String[] args) {
   new TextAreaExample();
}
}
```

Output:

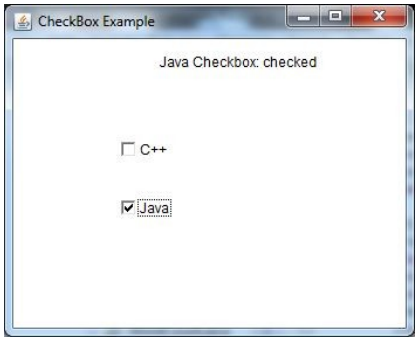


Java AWT Checkbox Example with ItemListener

```
import java.awt.*;
import java.awt.event.*;
public class CheckboxExample
{
    CheckboxExample(){
```

```
        Frame f= new Frame("CheckBox Example");
        final Label label = new Label();
        label.setAlignment(Label.CENTER);
        label.setSize(400,100);
        Checkbox checkbox1 = new Checkbox("C++");
        checkbox1.setBounds(100,100, 50,50);
        Checkbox checkbox2 = new Checkbox("Java");
        checkbox2.setBounds(100,150, 50,50);
        f.add(checkbox1); f.add(checkbox2); f.add(label);
        checkbox1.addItemListener(new ItemListener() {
            public void itemStateChanged(ItemEvent e) {
              label.setText("C++ Checkbox: "
              + (e.getStateChange()==1?"checked":"unchecked"));
            }
          });
        checkbox2.addItemListener(new ItemListener() {
            public void itemStateChanged(ItemEvent e) {
              label.setText("Java Checkbox: "
              + (e.getStateChange()==1?"checked":"unchecked"));
            }
          });
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
public static void main(String args[])
{
   new CheckboxExample();
}
}
```

Output:

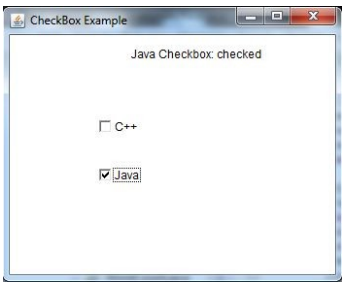


Java AWT Checkbox Example with ItemListener

```
import java.awt.*;
import java.awt.event.*;
public class CheckboxExample
{
    CheckboxExample(){
        Frame f= new Frame("CheckBox Example");
        final Label label = new Label();
        label.setAlignment(Label.CENTER);
        label.setSize(400,100);
        Checkbox checkbox1 = new Checkbox("C++");
        checkbox1.setBounds(100,100, 50,50);
        Checkbox checkbox2 = new Checkbox("Java");
```

```
        checkbox2.setBounds(100,150, 50,50);
        f.add(checkbox1); f.add(checkbox2); f.add(label);
        checkbox1.addItemListener(new ItemListener() {
            public void itemStateChanged(ItemEvent e) {
              label.setText("C++ Checkbox: "
              + (e.getStateChange()==1?"checked":"unchecked"));
            }
          });
        checkbox2.addItemListener(new ItemListener() {
            public void itemStateChanged(ItemEvent e) {
              label.setText("Java Checkbox: "
              + (e.getStateChange()==1?"checked":"unchecked"));
            }
          });
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
public static void main(String args[])
{
   new CheckboxExample();
}
}
```

Output:



Java AWT Choice

The object of Choice class is used to show popup menu of choices. Choice selected by user is shown on the top of a menu. It inherits Component class.

AWT Choice Class Declaration

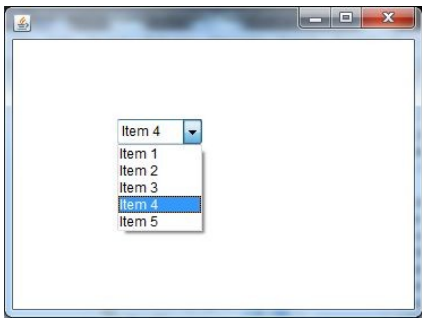
1. public class Choice extends Component implements ItemSelectable, Accessible

Java AWT Choice Example

```
import java.awt.*;
public class ChoiceExample
{
      ChoiceExample(){
      Frame f= new Frame();
      Choice c=new Choice();
      c.setBounds(100,100, 75,75);
      c.add("Item 1");
      c.add("Item 2");
      c.add("Item 3");
      c.add("Item 4");
      c.add("Item 5");
```

```
        f.add(c);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
public static void main(String args[])
{
  new ChoiceExample();
}
}
```

Output:



Java AWT List

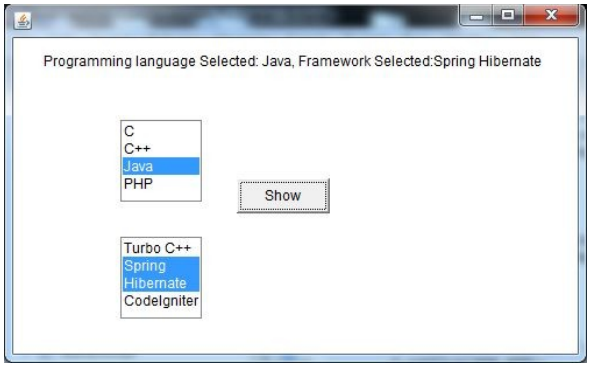
The object of List class represents a list of text items. By the help of list, user can choose either one item or multiple items. It inherits Component class.

Java AWT List Example with ActionListener

```
import java.awt.*;
import java.awt.event.*;
public class ListExample
{
    ListExample(){
      Frame f= new Frame();
      final Label label = new Label();
      label.setAlignment(Label.CENTER);
      label.setSize(500,100);
      Button b=new Button("Show");
      b.setBounds(200,150,80,30);
      final List l1=new List(4, false);
      l1.setBounds(100,100, 70,70);
      l1.add("C");
      l1.add("C++");
      l1.add("Java");
      l1.add("PHP");
      final List l2=new List(4, true);
      l2.setBounds(100,200, 70,70);
      l2.add("Turbo C++");
      l2.add("Spring");
      l2.add("Hibernate");
      l2.add("CodeIgniter");
      f.add(l1); f.add(l2); f.add(label); f.add(b);
      f.setSize(450,450);
      f.setLayout(null);
      f.setVisible(true);
      b.addActionListener(new ActionListener() {
```

```
        public void actionPerformed(ActionEvent e) {
         String data = "Programming language Selected: "+l1.getItem(l1.getSelectedIndex());
         data += ", Framework Selected:";
         for(String frame:l2.getSelectedItems()){
                data += frame + " ";
         }
         label.setText(data);
         }
        });
}
public static void main(String args[])
{
  new ListExample();
}
}
```

Output:



Java AWT Canvas

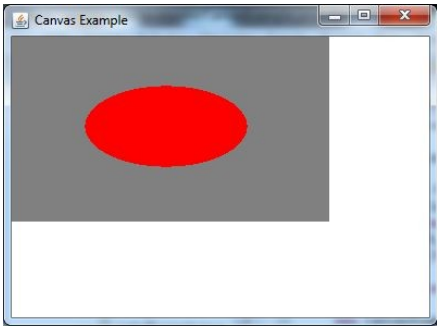
The Canvas control represents a blank rectangular area where the application can draw or trap input events from the user. It inherits the Component class.

Java AWT Canvas Example

```
import java.awt.*;
public class CanvasExample
{
 public CanvasExample()
 {
  Frame f= new Frame("Canvas Example");
  f.add(new MyCanvas());
  f.setLayout(null);
  f.setSize(400, 400);
  f.setVisible(true);
 }
 public static void main(String args[])
 {
  new CanvasExample();
 }
}
class MyCanvas extends Canvas
{
      public MyCanvas() {
      setBackground (Color.GRAY);
      setSize(300, 200);
    }
```

```
 public void paint(Graphics g)
 {
  g.setColor(Color.red);
  g.fillOval(75, 75, 150, 75);
 }
}
```

Output:



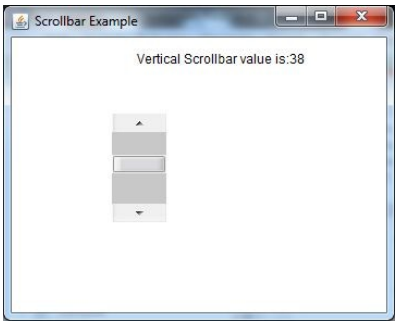
Java AWT Scrollbar

The object of Scrollbar class is used to add horizontal and vertical scrollbar. Scrollbar is a GUI component allows us to see invisible number of rows and columns.

Java AWT Scrollbar Example with AdjustmentListener

```
import java.awt.*;
import java.awt.event.*;
class ScrollbarExample{
    ScrollbarExample(){
        Frame f= new Frame("Scrollbar Example");
        final Label label = new Label();
        label.setAlignment(Label.CENTER);
        label.setSize(400,100);
        final Scrollbar s=new Scrollbar();
        s.setBounds(100,100, 50,100);
        f.add(s);f.add(label);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
        s.addAdjustmentListener(new AdjustmentListener() {
           public void adjustmentValueChanged(AdjustmentEvent e) {
             label.setText("Vertical Scrollbar value is:"+ s.getValue());
           }
        });
      }
public static void main(String args[]){
new ScrollbarExample();
}
}
```

Output:



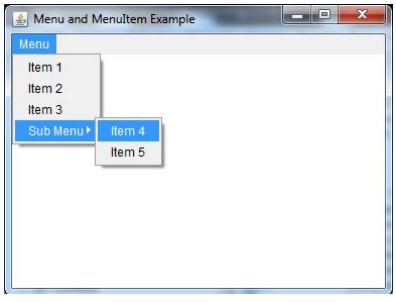
Java AWT MenuItem and Menu

The object of MenuItem class adds a simple labeled menu item on menu. The items used in a menu must belong to the MenuItem or any of its subclass.
The object of Menu class is a pull down menu component which is displayed on the menu bar. It inherits the MenuItem class.

Java AWT MenuItem and Menu Example

```
import java.awt.*;
class MenuExample
{
    MenuExample(){
        Frame f= new Frame("Menu and MenuItem Example");
        MenuBar mb=new MenuBar();
        Menu menu=new Menu("Menu");
        Menu submenu=new Menu("Sub Menu");
        MenuItem i1=new MenuItem("Item 1");
        MenuItem i2=new MenuItem("Item 2");
        MenuItem i3=new MenuItem("Item 3");
        MenuItem i4=new MenuItem("Item 4");
        MenuItem i5=new MenuItem("Item 5");
        menu.add(i1);
        menu.add(i2);
        menu.add(i3);
        submenu.add(i4);
        submenu.add(i5);
        menu.add(submenu);
        mb.add(menu);
        f.setMenuBar(mb);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
}
public static void main(String args[])
{
new MenuExample();
}
}
```

Output:



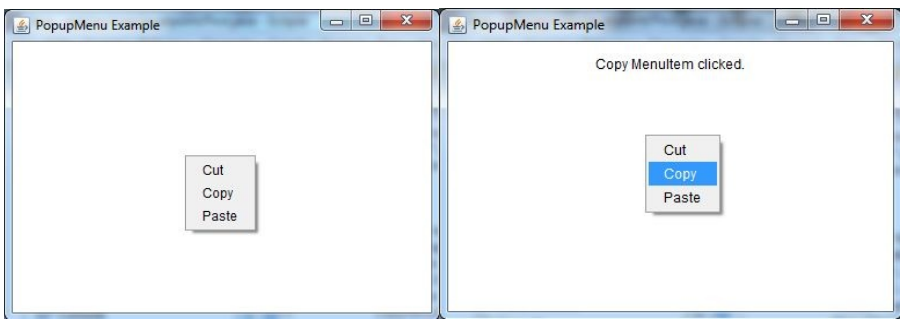
Java AWT PopupMenu

PopupMenu can be dynamically popped up at specific position within a component. It inherits the Menu class.

Java AWT PopupMenu Example

```
import java.awt.*;
import java.awt.event.*;
class PopupMenuExample
{
    PopupMenuExample(){
        final Frame f= new Frame("PopupMenu Example");
        final PopupMenu popupmenu = new PopupMenu("Edit");
        MenuItem cut = new MenuItem("Cut");
        cut.setActionCommand("Cut");
        MenuItem copy = new MenuItem("Copy");
        copy.setActionCommand("Copy");
        MenuItem paste = new MenuItem("Paste");
        paste.setActionCommand("Paste");
        popupmenu.add(cut);
        popupmenu.add(copy);
        popupmenu.add(paste);
        f.addMouseListener(new MouseAdapter() {
          public void mouseClicked(MouseEvent e) {
             popupmenu.show(f , e.getX(), e.getY());
          }
        });
        f.add(popupmenu);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
public static void main(String args[])
{
      new PopupMenuExample();
}
}
```

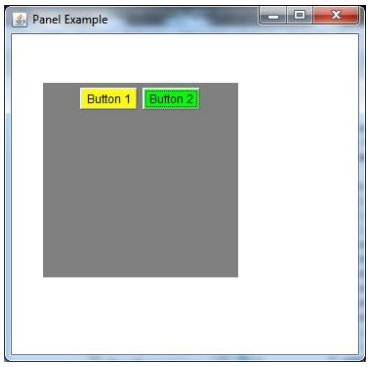
Output:



Java AWT Panel Example

```
import java.awt.*;
public class PanelExample {
    PanelExample()
      {
      Frame f= new Frame("Panel Example");
      Panel panel=new Panel();
      panel.setBounds(40,80,200,200);
      panel.setBackground(Color.gray);
      Button b1=new Button("Button 1");
      b1.setBounds(50,100,80,30);
      b1.setBackground(Color.yellow);
      Button b2=new Button("Button 2");
      b2.setBounds(100,100,80,30);
      b2.setBackground(Color.green);
      panel.add(b1); panel.add(b2);
      f.add(panel);
      f.setSize(400,400);
      f.setLayout(null);
      f.setVisible(true);
      }
      public static void main(String args[])
      {
      new PanelExample();
      }
}
```

Output:



Java AWT Dialog

The Dialog control represents a top level window with a border and a title used to take some form of input from the user. It inherits the Window class.

Unlike Frame, it doesn't have maximize and minimize buttons.

Java AWT Dialog Example

```
import java.awt.*;
import java.awt.event.*;
public class DialogExample {
   private static Dialog d;
   DialogExample() {
      Frame f= new Frame();
      d = new Dialog(f , "Dialog Example", true);
```

```
        d.setLayout( new FlowLayout() );
        Button b = new Button ("OK");
        b.addActionListener ( new ActionListener()
        {
            public void actionPerformed( ActionEvent e )
            {
                DialogExample.d.setVisible(false);
            }
        });
        d.add( new Label ("Click button to continue."));
        d.add(b);
        d.setSize(300,300);
        d.setVisible(true);
    }
    public static void main(String args[])
    {
        new DialogExample();
    }
}
```

Output: