Java AWT and Event Handling

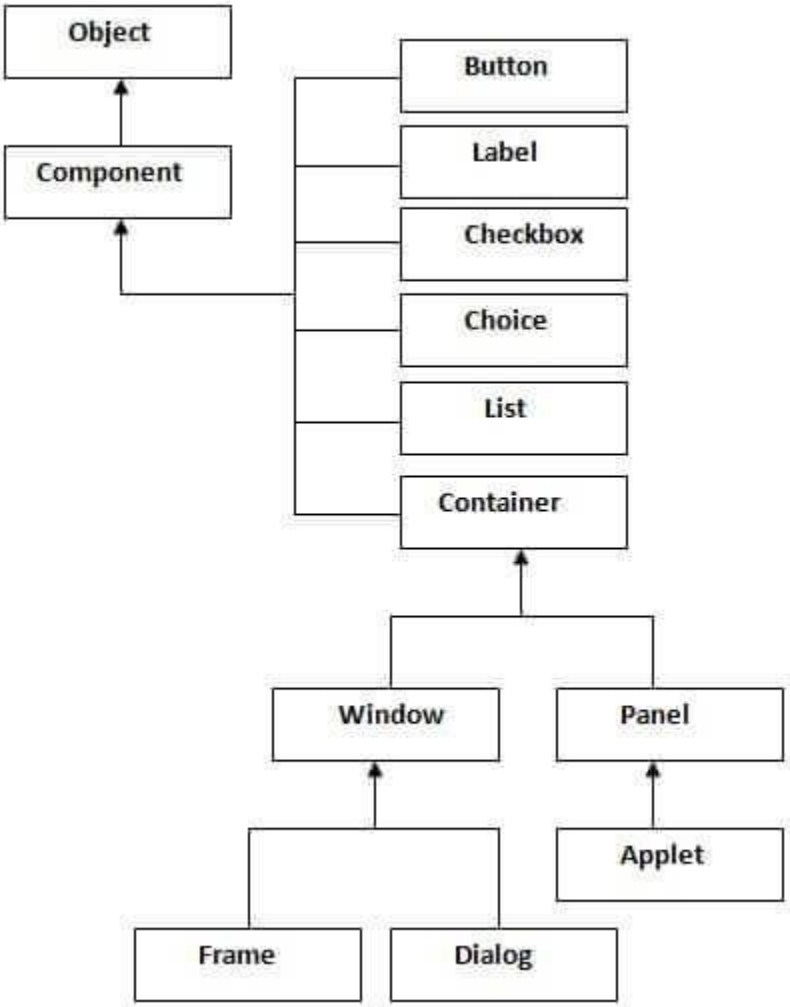
Java AWT (Abstract Window Toolkit) is *an API to develop GUI or window-based applications* in java.

Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavyweight i.e. its components are using the resources of OS.

The java.awt package provides classes for AWT api such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.

Java AWT Hierarchy

The hierarchy of Java AWT classes are given below.



Container

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

Window

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

Panel

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

Frame

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

Useful Methods of Component class

Method	Description
public void add(Component c)	inserts a component on this component.
public void setSize(int width,int height)	sets the size (width and height) of the component.
public void setLayout(LayoutManager m)	defines the layout manager for the component.
public void setVisible(boolean status)	changes the visibility of the component, by default false.

AWT Example

To create simple awt example, you need a frame. There are two ways to create a frame in AWT.

- By extending Frame class (inheritance)
- By creating the object of Frame class (association)

AWT Example by Inheritance

Let's see a simple example of AWT where we are inheriting Frame class. Here, we are showing Button component on the Frame.

```
class First extends Frame
{
        First()
        {
                Button b=new Button("click me");
                b.setBounds(30,100,80,30);// setting button position
                add(b);//adding button into frame
                setSize(300,300);//frame size 300 width and 300 height
                setLayout(null);//no layout now bydefault BorderLayout
                setVisible(true);//now frame willbe visible, bydefault not visible
        }
        public static void main(String args[])
        {
                First f=new First();

        }
```

```
}
```

AWT Example by Association

Let's see a simple example of AWT where we are creating instance of Frame class. Here, we are showing Button component on the Frame.

```
import java.awt.*;
class First2
{
    First2()
    {
        Frame f=new Frame();
        Button b=new Button("click me");
        b.setBounds(30,50,80,30);
        f.add(b);
        f.setSize(300,300);
        f.setLayout(null);
        f.setVisible(true);
    }
    public static void main(String args[])
    {
        First2 f=new First2();
    }
}
```

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

Java ActionListener Interface

The Java ActionListener is notified whenever you click on the button or menu item. It is notified against ActionEvent. The ActionListener interface is found in java.awt.event package. It has only one method: actionPerformed().

actionPerformed() method

The actionPerformed() method is invoked automatically whenever you click on the registered component.

public abstract void actionPerformed(ActionEvent e);

How to write ActionListener

The common approach is to implement the ActionListener. If you implement the ActionListener class, you need to follow 3 steps:

1) Implement the ActionListener interface in the class:
   public class ActionListenerExample Implements ActionListener
2) Register the component with the Listener:
   component.addActionListener(instanceOfListenerclass);

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){}

3) Override the actionPerformed() method:

```
void actionPerformed(ActionEvent e){
        //Write the code here
}
```

Event Handling Code

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

1. Within class
2. Other class
3. Anonymous class

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);
		/*addWindowListener(new WindowAdapter()
		{
			public void windowClosing(WindowEvent e)
			{
				dispose();
			}
		}); */
		setSize(300,300);
		setLayout(null);
		setVisible(true);
	}
	public void actionPerformed(ActionEvent e)
	{
		tf.setText("Welcome");
	}
	public static void main(String args[])
	{
		new AEvent();
	}
}
```

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");
   }
}
```

3) <u>Java event handling by anonymous class</u>

```
import java.awt.*;
import java.awt.event.*;
class AEvent3 extends Frame
```

```
{
	TextField tf;
	AEvent3()
	{
		tf=new TextField();
		tf.setBounds(60,50,170,20);
		Button b=new Button("click me");
		b.setBounds(50,120,80,30);

		b.addActionListener(new ActionListener()
		{
			public void actionPerformed(ActionEvent e)
			{
				tf.setText("hello");
			}
		});
		add(b);add(tf);
		setSize(300,300);
		setLayout(null);
		setVisible(true);
	}
	public static void main(String args[])
	{
		new AEvent3();
	}
}
```

Java MouseListener Interface

The Java MouseListener is notified whenever you change the state of mouse. It is notified against MouseEvent. The MouseListener interface is found in java.awt.event package. It has five methods.

Methods of MouseListener interface

The signature of 5 methods found in MouseListener interface are given below:

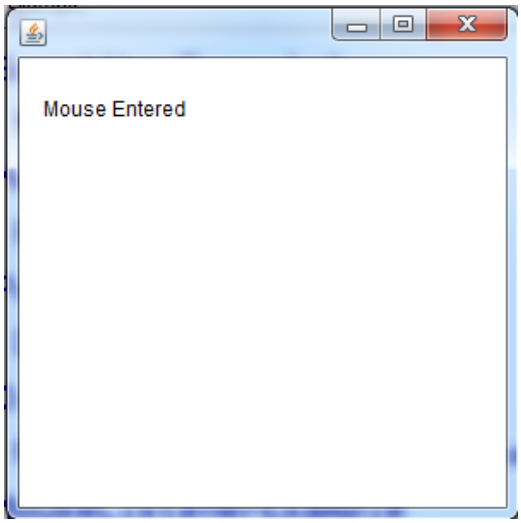
1. public abstract void mouseClicked(MouseEvent e);
2. public abstract void mouseEntered(MouseEvent e);
3. public abstract void mouseExited(MouseEvent e);
4. public abstract void mousePressed(MouseEvent e);
5. public abstract void mouseReleased(MouseEvent e);

MouseListener Example

```
import java.awt.*;
import java.awt.event.*;
public class MouseListenerExample extends Frame implements MouseListener
```

```
{
        Label l;
        MouseListenerExample()
        {
                addMouseListener(this);
                 l=new Label();
                l.setBounds(20,50,100,20);
                add(l);
                setSize(300,300);
                setLayout(null);
                setVisible(true);
         }
         public void mouseClicked(MouseEvent e)
        {
                l.setText("Mouse Clicked");
         }
        public void mouseEntered(MouseEvent e)
        {
                l.setText("Mouse Entered");
         }
         public void mouseExited(MouseEvent e)
        {
                 l.setText("Mouse Exited");
         }
        public void mousePressed(MouseEvent e)
        {
                 l.setText("Mouse Pressed");
         }
        public void mouseReleased(MouseEvent e)
        {
                l.setText("Mouse Released");
         }
        public static void main(String[] args)
        {
                new MouseListenerExample();
        }
}
```

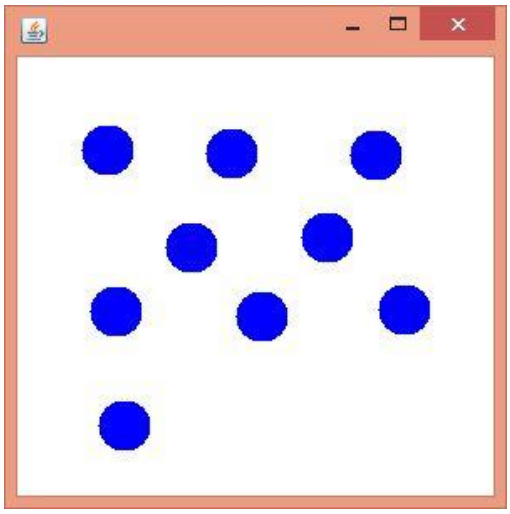
Output:



MouseListener Example 2

```
import java.awt.*;
import java.awt.event.*;
public class MouseListenerExample2 extends Frame implements MouseListener
{
        MouseListenerExample2()
        {
                addMouseListener(this);
                setSize(300,300);
                setLayout(null);
                setVisible(true);
        }
        public void mouseClicked(MouseEvent e)
        {
                Graphics g=getGraphics();
                g.setColor(Color.BLUE);
                g.fillOval(e.getX(),e.getY(),30,30);
        }
        public void mouseEntered(MouseEvent e) {}
        public void mouseExited(MouseEvent e) {}
         public void mousePressed(MouseEvent e) {}
        public void mouseReleased(MouseEvent e) {}
         public static void main(String[] args)
        {
                new MouseListenerExample2();
        }
}
```

Output:



MouseMotionListener Interface

The Java MouseMotionListener is notified whenever you move or drag mouse. It is notified against MouseEvent. The MouseMotionListener interface is found in java.awt.event package. It has two methods.

Methods of MouseMotionListener interface

The signature of 2 methods found in MouseMotionListener interface are given below:

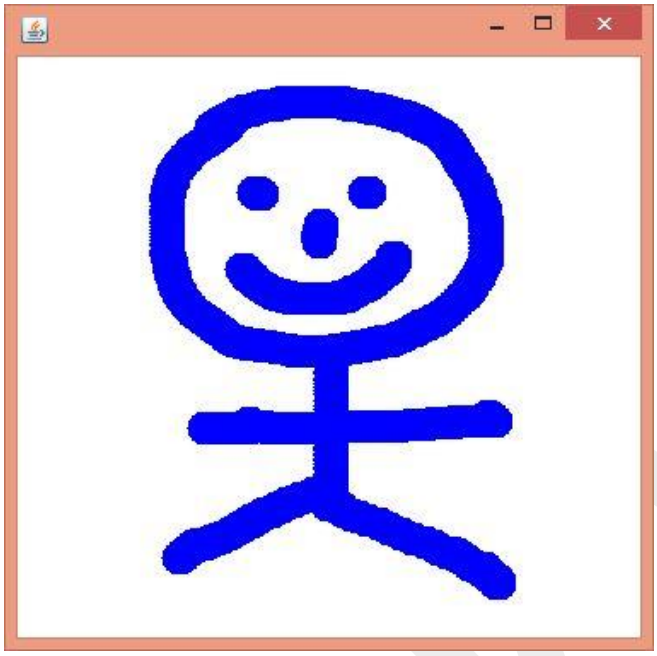
1. public abstract void mouseDragged(MouseEvent e);
2. public abstract void mouseMoved(MouseEvent e);

MouseMotionListener Example

```
import java.awt.*;
import java.awt.event.*;
public class MouseMotionListenerExample extends Frame implements
MouseMotionListener
{
        MouseMotionListenerExample()
        {
                addMouseMotionListener(this);
                setSize(500,500);
                setLayout(null);
                setVisible(true);
        }
        public void mouseDragged(MouseEvent e)
        {
                Graphics g=getGraphics();
                g.setColor(Color.BLUE);
                g.fillOval(e.getX(),e.getY(),20,20);
        }
        public void mouseMoved(MouseEvent e) {}
        public static void main(String[] args)
```

```
        {
                new MouseMotionListenerExample();
        }
}
```

Output:



AWT Controls:

AWT Button

The button class is used to create a labeled button that has platform independent implementation. The application result in some action when the button is pushed.

AWT Button Class declaration

public class Button extends Component implements Accessible

AWT Label

The object of Label class is a component for placing text in a container. It is used to display a single line of read only text. The text can be changed by an application but a user cannot edit it directly.

AWT Label Class Declaration

public class Label extends Component implements Accessible

AWT TextField

The object of a TextField class is a text component that allows the editing of a single line text. It inherits TextComponent class.

AWT TextField Class Declaration

public class TextField extends TextComponent

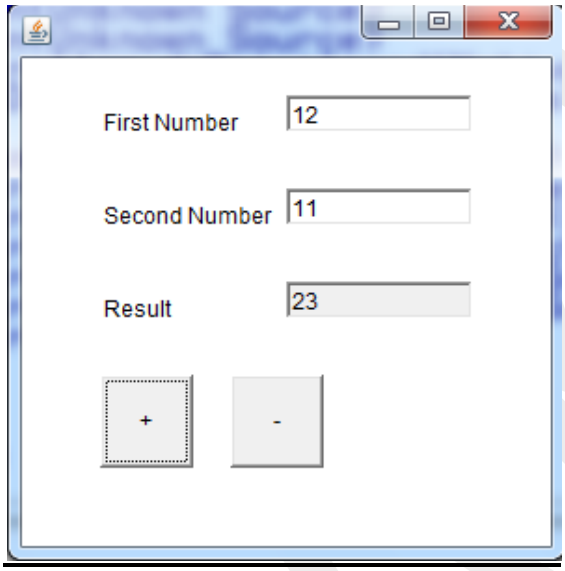
Example

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

```
public class TextFieldExample extends Frame implements ActionListener
{
	TextField tf1,tf2,tf3;
	Button b1,b2;
	Label l1,l2,l3;
	TextFieldExample()
	{
		l1=new Label("First Number");
		l1.setBounds(50,50,100,30);
		tf1=new TextField();
		tf1.setBounds(150,50,100,20);
		l2=new Label("Second Number");
		l2.setBounds(50,100, 100,30);
		tf2=new TextField();
		tf2.setBounds(150,100,100,20);
		l3=new Label("Result");
		l3.setBounds(50,150,100,30);
		tf3=new TextField();
		tf3.setBounds(150,150,100,20);
		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);add(l1);add(l2);add(l3);
		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:



AWT TextArea

The object of a TextArea class is a multi line region that displays text. It allows the editing of multiple line text. It inherits TextComponent class.

AWT TextArea Class Declaration

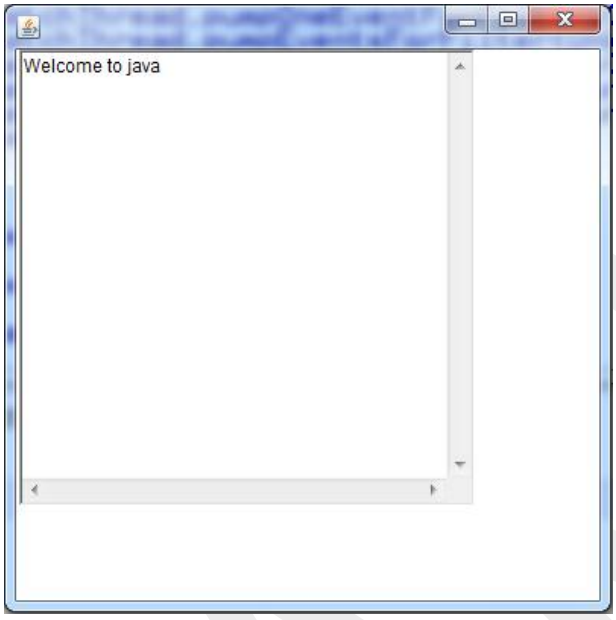
public class TextArea extends TextComponent

Example

```
import java.awt.*;
public class TextAreaExample1
{
      TextAreaExample1()
      {
            Frame f= new Frame();
            TextArea area=new TextArea("Welcome to java");
            area.setBounds(10,30, 300,300);
            f.add(area);
            f.setSize(400,400);
```

```
            f.setLayout(null);
            f.setVisible(true);
        }
        public static void main(String args[])
        {
            new TextAreaExample1();
        }
}
```

Output:



AWT Checkbox

The Checkbox class is used to create a checkbox. It is used to turn an option on (true) or off (false). Clicking on a Checkbox changes its state from "on" to "off" or from "off" to "on".

AWT Checkbox Class Declaration

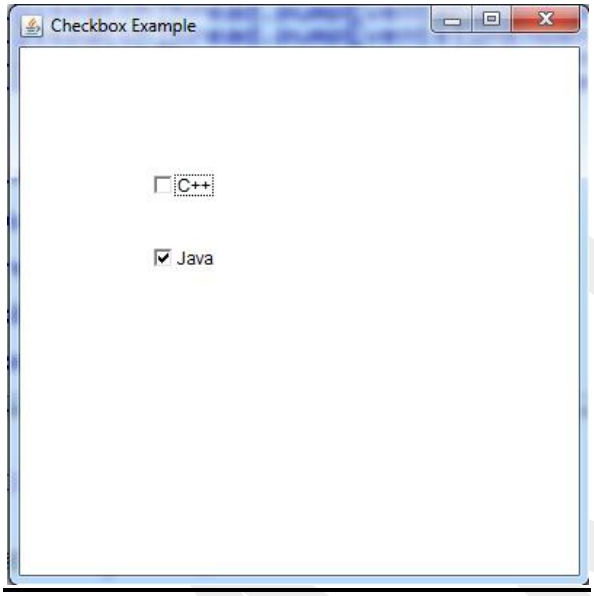
public class Checkbox extends Component implements ItemSelectable, Accessible

Example 1

```
import java.awt.*;
public class CheckboxExample1
{
        CheckboxExample1()
        {
            Frame f= new Frame("Checkbox Example");
            Checkbox checkbox1 = new Checkbox("C++");
            checkbox1.setBounds(100,100, 50,50);
            Checkbox checkbox2 = new Checkbox("Java", true);
             checkbox2.setBounds(100,150, 50,50);
             f.add(checkbox1);
             f.add(checkbox2);
             f.setSize(400,400);
```

```
            f.setLayout(null);
            f.setVisible(true);
        }
        public static void main(String args[])
        {
            new CheckboxExample1();
        }
}
```

Output:

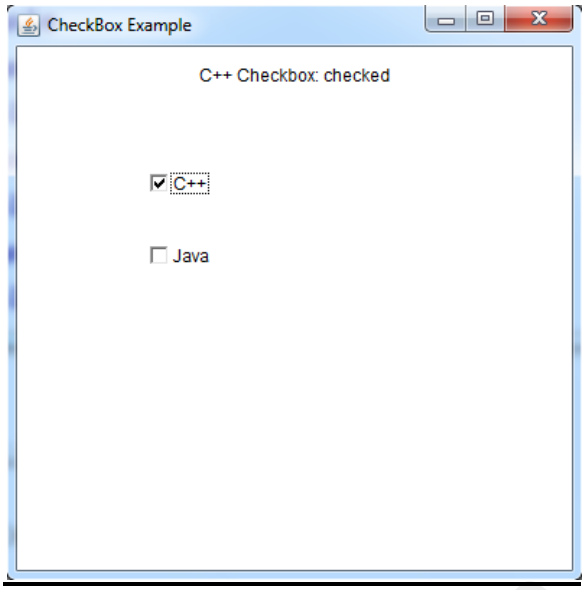


Example 2

```
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);
        f.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent e)
```

```
                {
                        System.exit(0);//dispose();
                }
        });
        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:



AWT CheckboxGroup

The object of CheckboxGroup class is used to group together a set of Checkbox. At a time only one check box button is allowed to be in "on" state and remaining check box button in "off" state. It inherits the object class.

Note: CheckboxGroup enables you to create radio buttons in AWT. There is no special control for creating radio buttons in AWT.

AWT CheckboxGroup Class Declaration

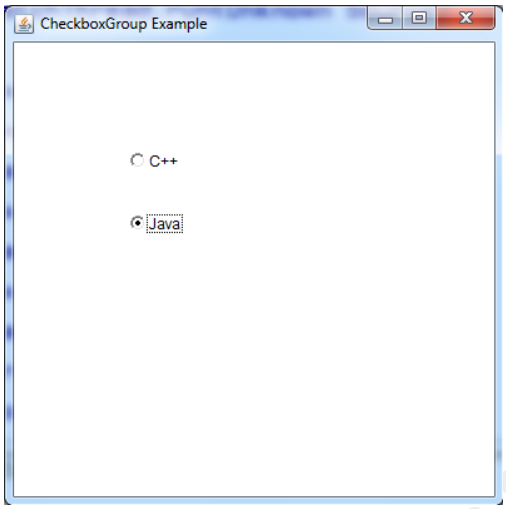
public class CheckboxGroup extends Object implements Serializable

Example 1

```
import java.awt.*;
public class CheckboxGroupExample1
{
        CheckboxGroupExample1()
        {
                Frame f= new Frame("CheckboxGroup Example");
                CheckboxGroup cbg = new CheckboxGroup();
                Checkbox checkBox1 = new Checkbox("C++", cbg, false);
                checkBox1.setBounds(100,100, 50,50);
                Checkbox checkBox2 = new Checkbox("Java", cbg, true);
                checkBox2.setBounds(100,150, 50,50);
                f.add(checkBox1);
                f.add(checkBox2);
                f.setSize(400,400);
                f.setLayout(null);
                f.setVisible(true);
        }
        public static void main(String args[])
        {
                new CheckboxGroupExample1();
```

```
        }
}
```

Output:

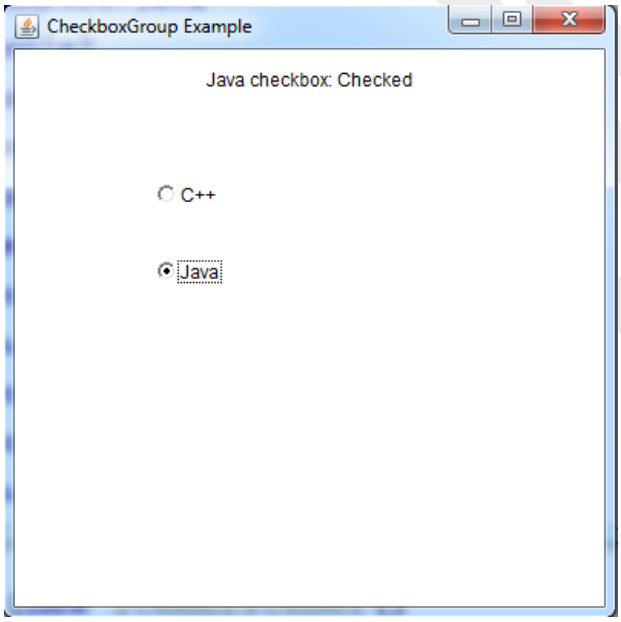


Example 2

```
import java.awt.*;
import java.awt.event.*;
public class CheckBoxGroupExample
{
        CheckBoxGroupExample()
        {
                Frame f= new Frame("CheckboxGroup Example");
                final Label label = new Label();
                label.setAlignment(Label.CENTER);
                label.setSize(400,100);
                CheckboxGroup cbg = new CheckboxGroup();
                Checkbox checkBox1 = new Checkbox("C++", cbg, false);
                checkBox1.setBounds(100,100, 50,50);
                Checkbox checkBox2 = new Checkbox("Java", cbg, false);
                checkBox2.setBounds(100,150, 50,50);
                f.add(checkBox1); f.add(checkBox2); f.add(label);
                f.setSize(400,400);
                f.setLayout(null);
                f.setVisible(true);
                f.addWindowListener(new WindowAdapter()
                {
                        public void windowClosing(WindowEvent e)
                        {
                                System.exit(0);//dispose();
                        }
                });
                checkBox1.addItemListener(new ItemListener()
```

```
        {
            public void itemStateChanged(ItemEvent e)
            {
                label.setText("C++ checkbox: Checked");
            }
        });
        checkBox2.addItemListener(new ItemListener()
        {
            public void itemStateChanged(ItemEvent e)
            {
                label.setText("Java checkbox: Checked");
            }
        });
    }
    public static void main(String args[])
    {
        new CheckBoxGroupExample();
    }
}
```

Output:



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

public class Choice extends Component implements ItemSelectable, Accessible

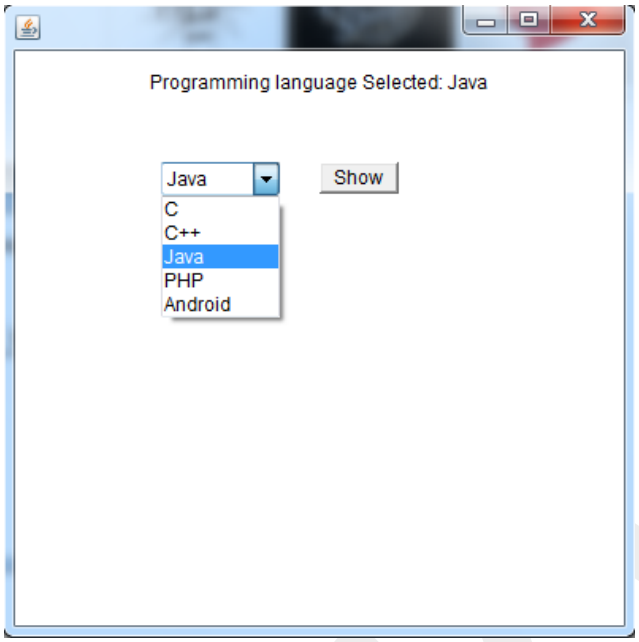
AWT Choice Example 1

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

```
import java.awt.event.*;
public class ChoiceExample
{
	ChoiceExample()
	{
		Frame f= new Frame();
		final Label label = new Label();
		label.setAlignment(Label.CENTER);
		label.setSize(400,100);
		Button b=new Button("Show");
		b.setBounds(200,100,50,20);
		final Choice c=new Choice();
		c.setBounds(100,100, 75,75);
		c.add("C");
		c.add("C++");
		c.add("Java");
		c.add("PHP");
		c.add("Android");
		f.add(c);f.add(label); f.add(b);
		f.setSize(400,400);
		f.setLayout(null);
		f.setVisible(true);
		f.addWindowListener(new WindowAdapter()
		{
			public void windowClosing(WindowEvent e)
			{
				System.exit(0);//dispose();
			}
		});
		b.addActionListener(new ActionListener()
		{
			public void actionPerformed(ActionEvent e)
			{
				String data = "Programming language Selected: "+
c.getItem(c.getSelectedIndex());
				label.setText(data);
			}
		});
	}
	public static void main(String args[])
	{
		new ChoiceExample();
```

```
        }
}
```

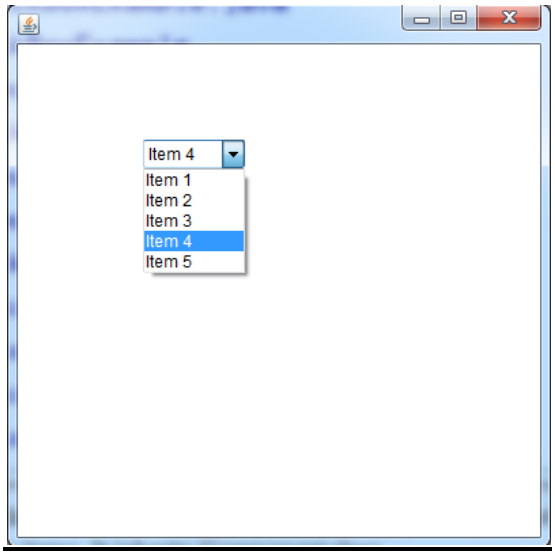
Output:



AWT Choice Example 2

```
public class ChoiceExample1
{
        ChoiceExample1()
        {
                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 ChoiceExample1();
        }
}
```

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.

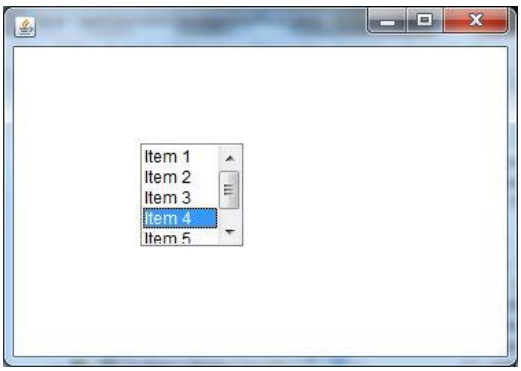
AWT List class Declaration

public class List extends Component implements ItemSelectable, Accessible

AWT List Example 1

```
import java.awt.*;
public class ListExample1
{
        ListExample1()
        {
                Frame f= new Frame();
                List l1=new List(5);
                l1.setBounds(100,100, 75,75);
                l1.add("Item 1");
                l1.add("Item 2");
                l1.add("Item 3");
                l1.add("Item 4");
                l1.add("Item 5");
                f.add(l1);
                f.setSize(400,400);
                f.setLayout(null);
                f.setVisible(true);
        }
        public static void main(String args[])
        {
                new ListExample1();
        }
}
```

Output:

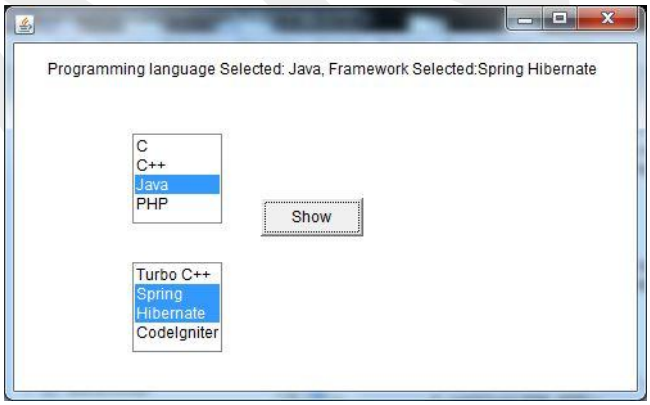


AWT List Example 2

```
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);
                f.addWindowListener(new WindowAdapter()
                {
                        public void windowClosing(WindowEvent e)
                        {
```

```
                    System.exit(0);
                }
            });
            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



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.

AWT Canvas class Declaration

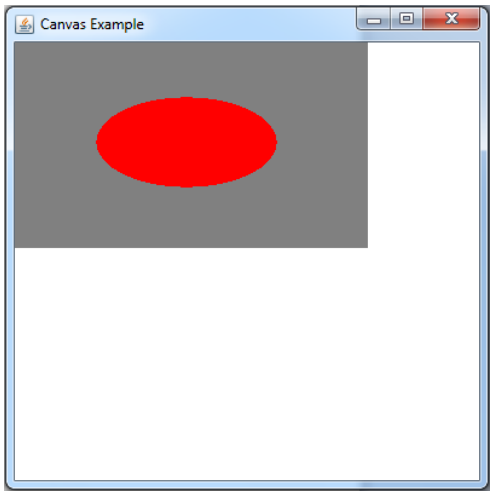
public class Canvas extends Component implements Accessible

AWT Canvas Example

```
import java.awt.*;
import java.awt.event.*;
public class CanvasExample
```

```
{
	public CanvasExample()
	{
		Frame f= new Frame("Canvas Example");
		f.add(new MyCanvas());
		f.setLayout(null);
		f.addWindowListener(new WindowAdapter()
		{
			public void windowClosing(WindowEvent e)
			{
				System.exit(0);
			}
		});
		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:



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.

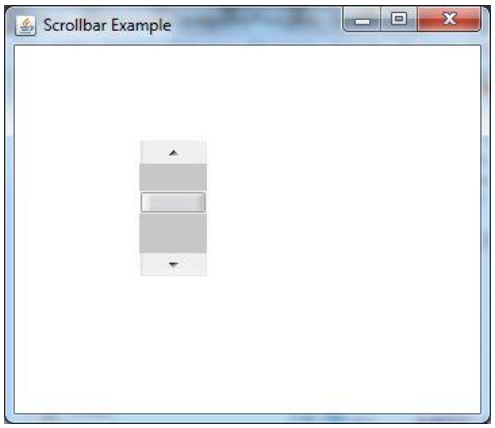
AWT Scrollbar class declaration

public class Scrollbar extends Component implements Adjustable, Accessible

AWT Scrollbar Example

```
import java.awt.*;
class ScrollbarExample
{
        ScrollbarExample()
        {
                Frame f= new Frame("Scrollbar Example");
                Scrollbar s=new Scrollbar();
                s.setBounds(100,100, 50,100);
                f.add(s);
                f.setSize(400,400);
                f.setLayout(null);
                f.setVisible(true);
        }
        public static void main(String args[])
        {
                new ScrollbarExample();
        }
}
```

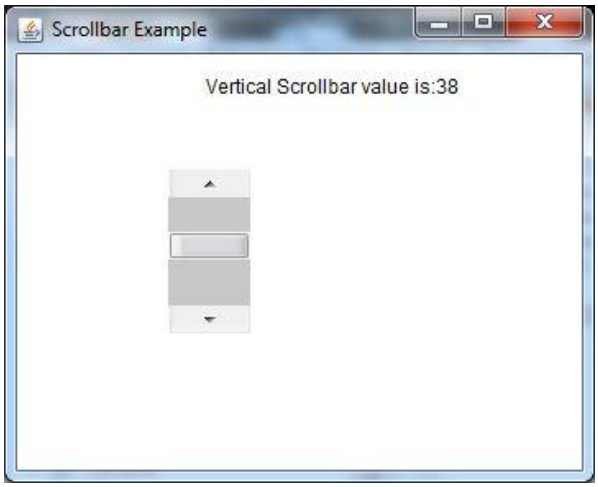
Output:



AWT Scrollbar Example with AdjustmentListener

```
import java.awt.*;
import java.awt.event.*;
class ScrollbarExample1
{
        ScrollbarExample1()
        {
                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 ScrollbarExample1();
        }
}
```

Output:



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.

AWT MenuItem class declaration

public class MenuItem extends MenuComponent implements Accessible

AWT Menu class declaration

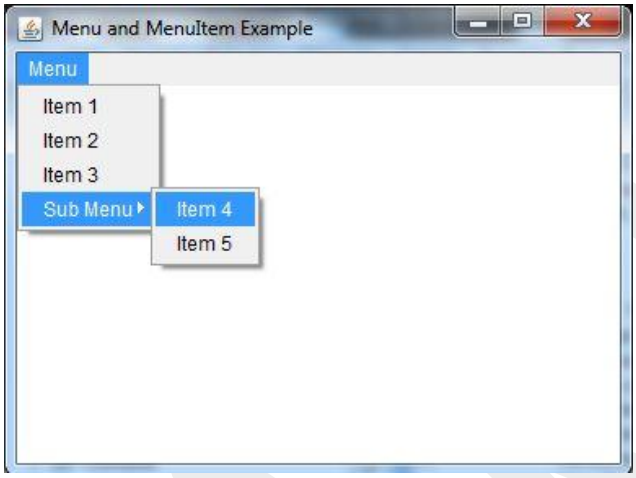
public class Menu extends MenuItem implements MenuContainer, Accessible

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:



AWT Panel

The Panel is a simplest container class. It provides space in which an application can attach any other component. It inherits the Container class.

It doesn't have title bar.

AWT Panel class declaration

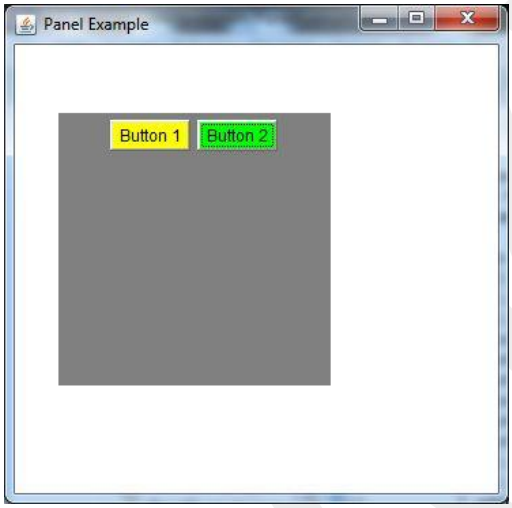
public class Panel extends Container implements Accessible

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.

Frame vs Dialog

Frame and Dialog both inherits Window class. Frame has maximize and minimize buttons but Dialog doesn't have.

AWT Dialog class declaration

public class Dialog extends Window

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:

How to close AWT Window in Java

We can close the AWT Window or Frame by calling *dispose()* or *System.exit()* inside windowClosing() method. The windowClosing() method is found in **WindowListener** interface and **WindowAdapter** class.

The WindowAdapter class implements WindowListener interfaces. It provides the default implementation of all the 7 methods of WindowListener interface. To override the windowClosing() method, you can either use WindowAdapter class or WindowListener interface.

If you implement the WindowListener interface, you will be forced to override all the 7 methods of WindowListener interface. So it is better to use WindowAdapter class.

Close AWT Window Example

```
import java.awt.*;
import java.awt.event.*;
public class WindowClosingExample extends Frame
{
	WindowClosingExample()
	{
		addWindowListener(new WindowAdapter()
		{
			public void windowClosing(WindowEvent e)
			{
				dispose();
			}
		});
		setSize(400,400);
		setLayout(null);
		setVisible(true);
	}
	public static void main(String[] args)
	{
		new WindowClosingExample();
	}
}
```

KeyListener Interface

The Java KeyListener is notified whenever you change the state of key. It is notified against KeyEvent. The KeyListener interface is found in java.awt.event package. It has three methods.

Methods of KeyListener interface

The signature of 3 methods found in KeyListener interface are given below:

- publicabstract void keyPressed(KeyEvent e);
- publicabstract void keyReleased(KeyEvent e);
- publicabstract void keyTyped(KeyEvent e);

KeyListener Example

```
import java.awt.*;
import java.awt.event.*;
public class KeyListenerExample extends Frame implements KeyListener
{
	Label l;
	TextArea area;
	KeyListenerExample()
	{
		l=new Label();
```

```
            l.setBounds(20,50,100,20);
            area=new TextArea();
            area.setBounds(20,80,300, 300);
            area.addKeyListener(this);
            add(l);add(area);
            setSize(400,400);
            setLayout(null);
            setVisible(true);
        }
        public void keyPressed(KeyEvent e)
        {
            l.setText("Key Pressed");
        }
        public void keyReleased(KeyEvent e)
        {
            l.setText("Key Released");
        }
        public void keyTyped(KeyEvent e)
        {
            l.setText("Key Typed");
        }
        public static void main(String[] args)
        {
            new KeyListenerExample();
        }
}
```

Output: