Java AWT Tutorial

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.

Java AWT Example

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

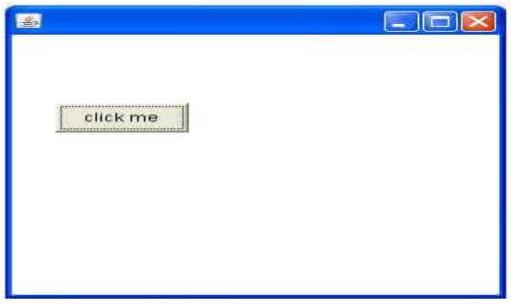
- By extending Frame class (inheritance)
- By creating the object of Frame class (association)

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.

```
import java.awt.*;
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 manager
setVisible(true);//now frame will be visible, by default not visible
}
public static void main(String args[]){
First f=new First();
} }
```

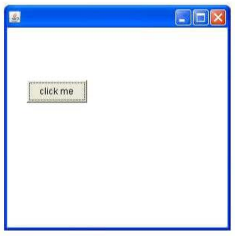
The setBounds(int x axis, int yaxis, int width, int height) method is used in the above example that sets the position of the awt button.



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();
}}
```



AWT Controls:

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.

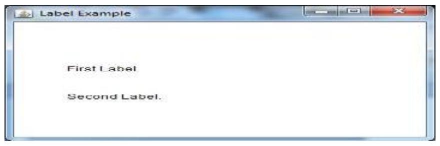
AWT Label Class Declaration

```
public class Label extends Component implements Accessible
```

Java Label Example

```
import java.awt.*;
class LabelExample{
public static void main(String args[]){
    Frame f= new Frame("Label Example");
    Label l1,l2;
    l1=new Label("First Label.");
    l1.setBounds(50,100, 100,30);
    l2=new Label("Second Label.");
    l2.setBounds(50,150, 100,30);
    f.add(l1); f.add(l2);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
}
}
```

Output:



Java 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.

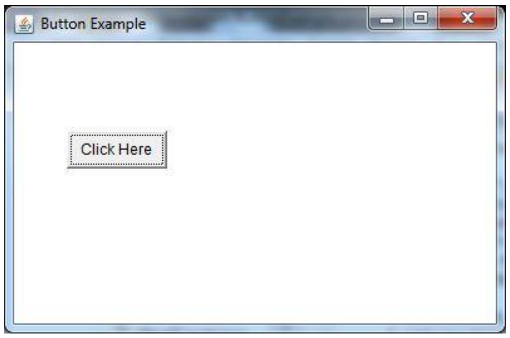
AWT Button Class declaration

```
public class Button extends Component implements Accessible
```

Java AWT Button Example

```
import java.awt.*;
public class ButtonExample {
public static void main(String[] args) {
    Frame f=new Frame("Button Example");
    Button b=new Button("Click Here");
    b.setBounds(50,100,80,30);
    f.add(b);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
}
}
```

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.

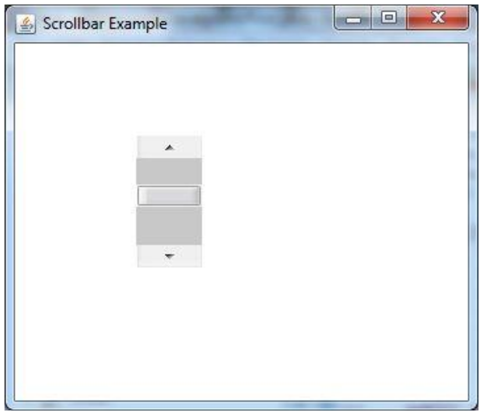
AWT Scrollbar class declaration

```
public class Scrollbar extends Component implements Adjustable, Accessible
```

Java 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:



Text Components:

Java 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.

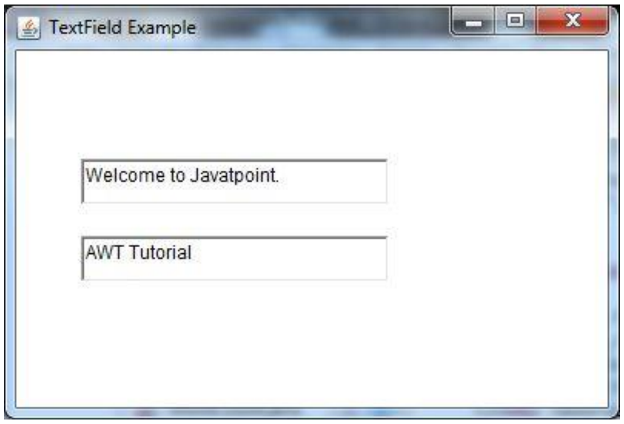
AWT TextField Class Declaration

```
public class TextField extends TextComponent
```

Java AWT TextField Example

```
import java.awt.*;
class TextFieldExample{
public static void main(String args[]){
    Frame f= new Frame("TextField Example");
    TextField t1,t2;
    t1=new TextField("Welcome to Javatpoint.");
    t1.setBounds(50,100, 200,30);
    t2=new TextField("AWT Tutorial");
    t2.setBounds(50,150, 200,30);
    f.add(t1); f.add(t2);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
}
}
```

Output:



Welcome to Javatpoint.

AWT Tutorial

Java 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

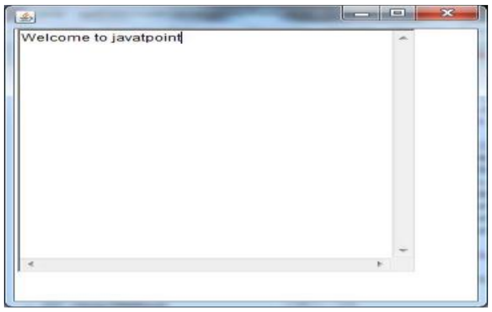
```
public class TextArea extends TextComponent
```

Java AWT TextArea Example

```
import java.awt.*;
public class TextAreaExample
{
    TextAreaExample(){
      Frame f= new Frame();
          TextArea area=new TextArea("Welcome to javatpoint");
      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 TextAreaExample();
}
}
```

Output:



```
Welcome to javatpoint
```

Java 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".

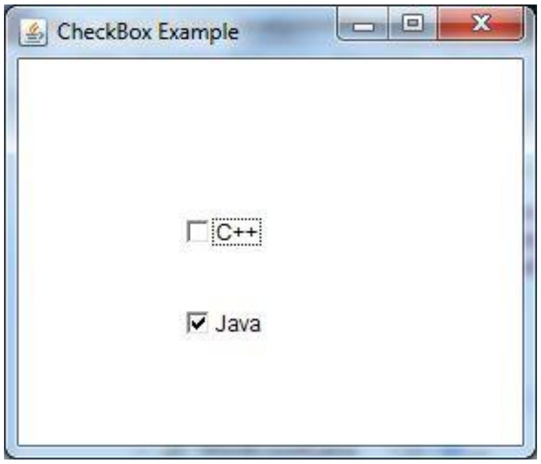
AWT Checkbox Class Declaration

```
public class Checkbox extends Component implements ItemSelectable, Accessible
```

Java AWT Checkbox Example

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

Output:



Java 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.

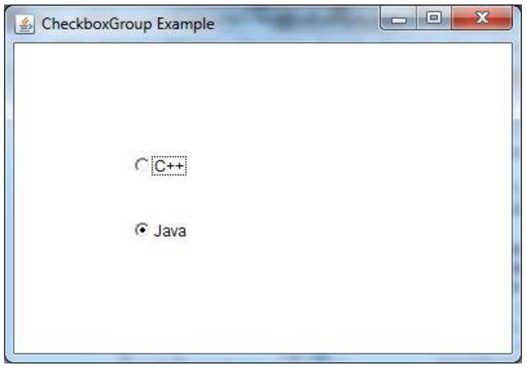
AWT CheckboxGroup Class Declaration

```
public class CheckboxGroup extends Object implements Serializable
```

Java AWT CheckboxGroup Example

```
import java.awt.*;
public class CheckboxGroupExample
{
       CheckboxGroupExample(){
       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 CheckboxGroupExample();
}
}
```

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.

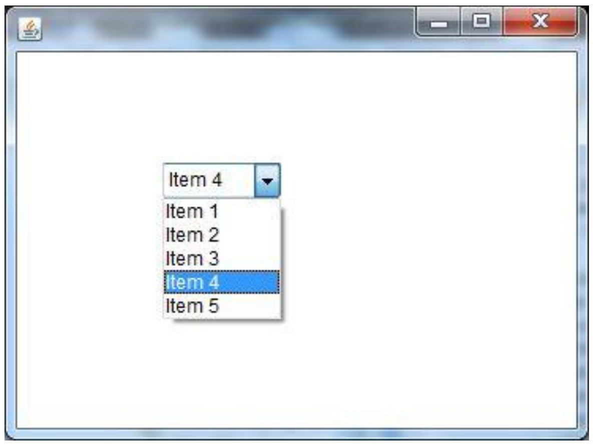
AWT Choice Class Declaration

```
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.

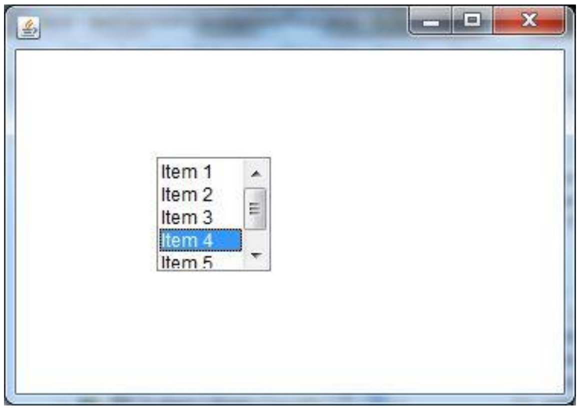
AWT List class Declaration

```
public class List extends Component implements ItemSelectable, Accessible
```

Java AWT List Example

```
import java.awt.*;
public class ListExample
{
    ListExample(){
        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 ListExample();
}
}
```

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.

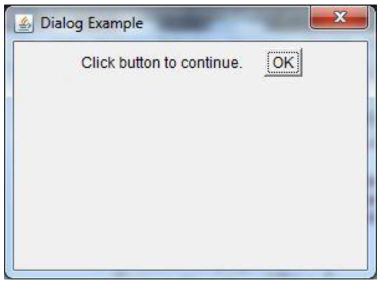
AWT Dialog class declaration

```
public class Dialog extends Window
```

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:



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.

AWT MenuItem class declaration

```
public class MenuItem extends MenuComponent implements Accessible
```

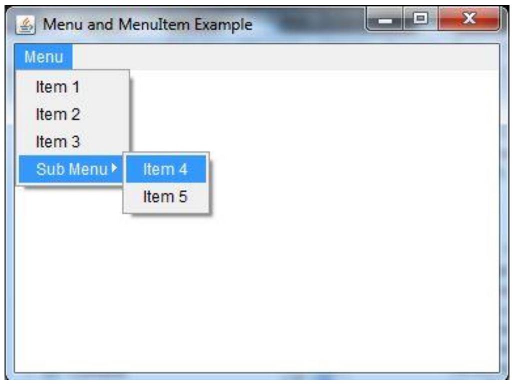
AWT Menu class declaration

```
public class Menu extends MenuItem implements MenuContainer, Accessible
```

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:



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 by implementing ActionListener

```
import java.awt.*;
import java.awt.event.*;
class AEvent extends Frame implements ActionListener{
TextField tf;
AEvent()
{
```

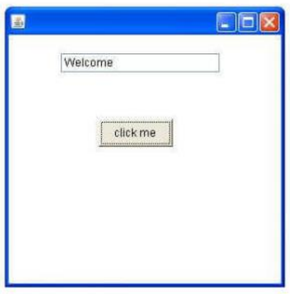
```
//create components
tf=new TextField();
tf.setBounds(60,50,170,20);
Button b=new Button("click me");
b.setBounds(100,120,80,30);

//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.

Output:



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:

```
public abstract void mouseClicked(MouseEvent e);
public abstract void mouseEntered(MouseEvent e);
public abstract void mouseExited(MouseEvent e);
public abstract void mousePressed(MouseEvent e);
public abstract void mouseReleased(MouseEvent e);
```

Java MouseListener

Example 1

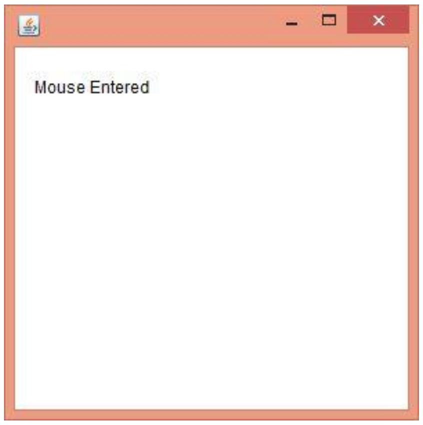
```
import java.awt.*;
import java.awt.event.*;
public class MouseListenerExample extends Frame implements MouseListener{
    Label l;
```

```
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:



Mouse Entered

Java MouseListener

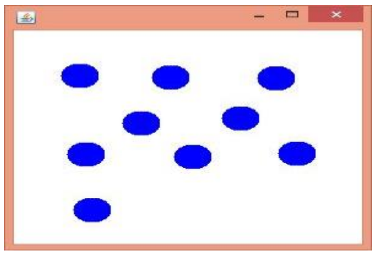
Example 2

```
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:



Java 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:

1. **public abstract void** keyPressed(KeyEvent e);
2. **public abstract void** keyReleased(KeyEvent e);

```
public abstract void keyTyped(KeyEvent e);
```

Java KeyListener

Example 1

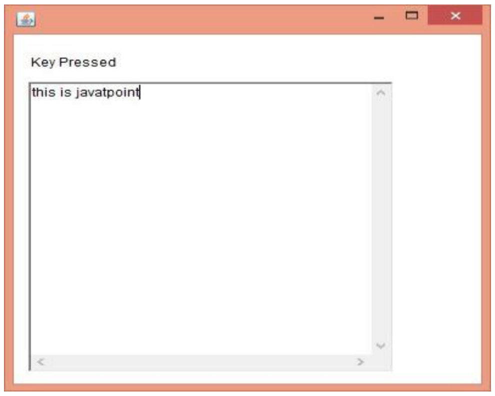
```
import java.awt.*;
import java.awt.event.*;
public class KeyListenerExample extends Frame implements KeyListener{
    Label l;
    TextArea area;
    KeyListenerExample(){

        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:



```
Key Pressed
this is javatpoint
```

Java KeyListener

Example 2: Count Words & Characters

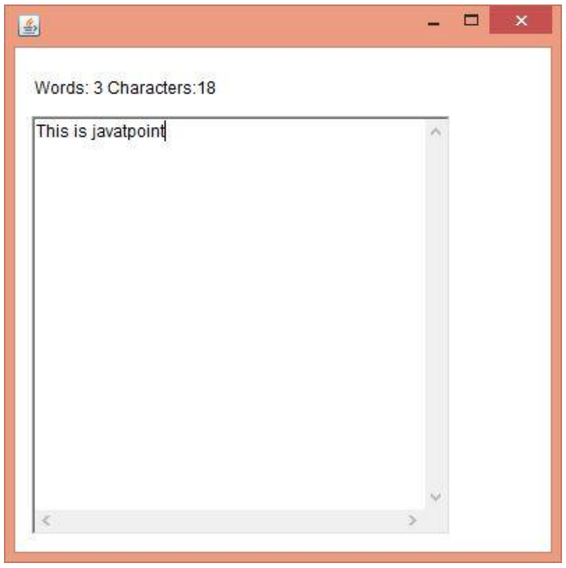
```
import java.awt.*;
import java.awt.event.*;
public class KeyListenerExample extends Frame implements KeyListener{
    Label l;
    TextArea area;
KeyListenerExample(){

    l=new Label();
    l.setBounds(20,50,200,20);
    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) {}
public void keyReleased(KeyEvent e) {
    String text=area.getText();
    String words[]=text.split("\\s");
    l.setText("Words: "+words.length+" Characters:"+text.length());
}
public void keyTyped(KeyEvent e) {}

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

Output:


```
Words: 3 Characters:18
This is javatpoint
```

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** package. 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.awt.dnd Adapter classes

Adapter class	Listener interface
DragSourceAdapter	DragSourceListener
DragTargetAdapter	DragTargetListener

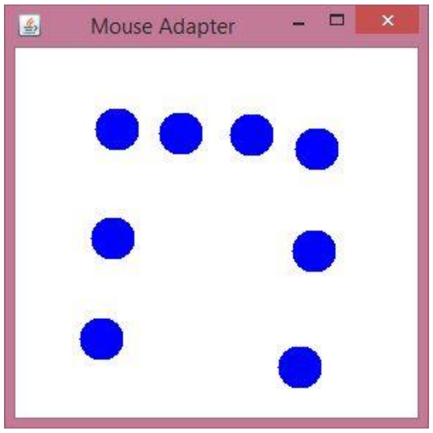
Java MouseAdapter Example

```
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:

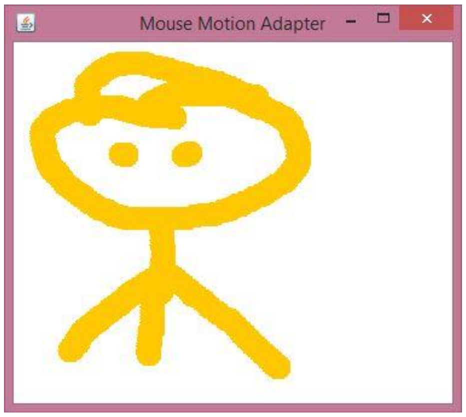


Java MouseMotionAdapter Example

```java
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:



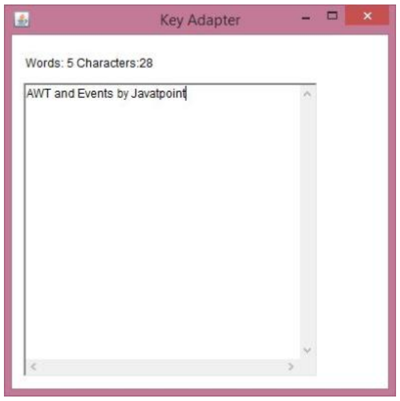
Java KeyAdapter Example

```java
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:



```
Words: 5 Characters:28
AWT and Events by Javatpoint
```