

awt control button added in frame and applet with example

```
import java.awt.*;

public class myframe extends Frame
{
    public static void main(String[] args)
    {
        myframe f=new myframe();
        Button b1=new Button("Click Here");
        b1.setBounds(120,70,80,80);
        f.add(b1);
        Button b2=new Button("submit");
        b2.setBounds(60,35,80,80);
        f.add(b2);
        f.setSize(400,400);
        f.setVisible(true);
    }
}
```

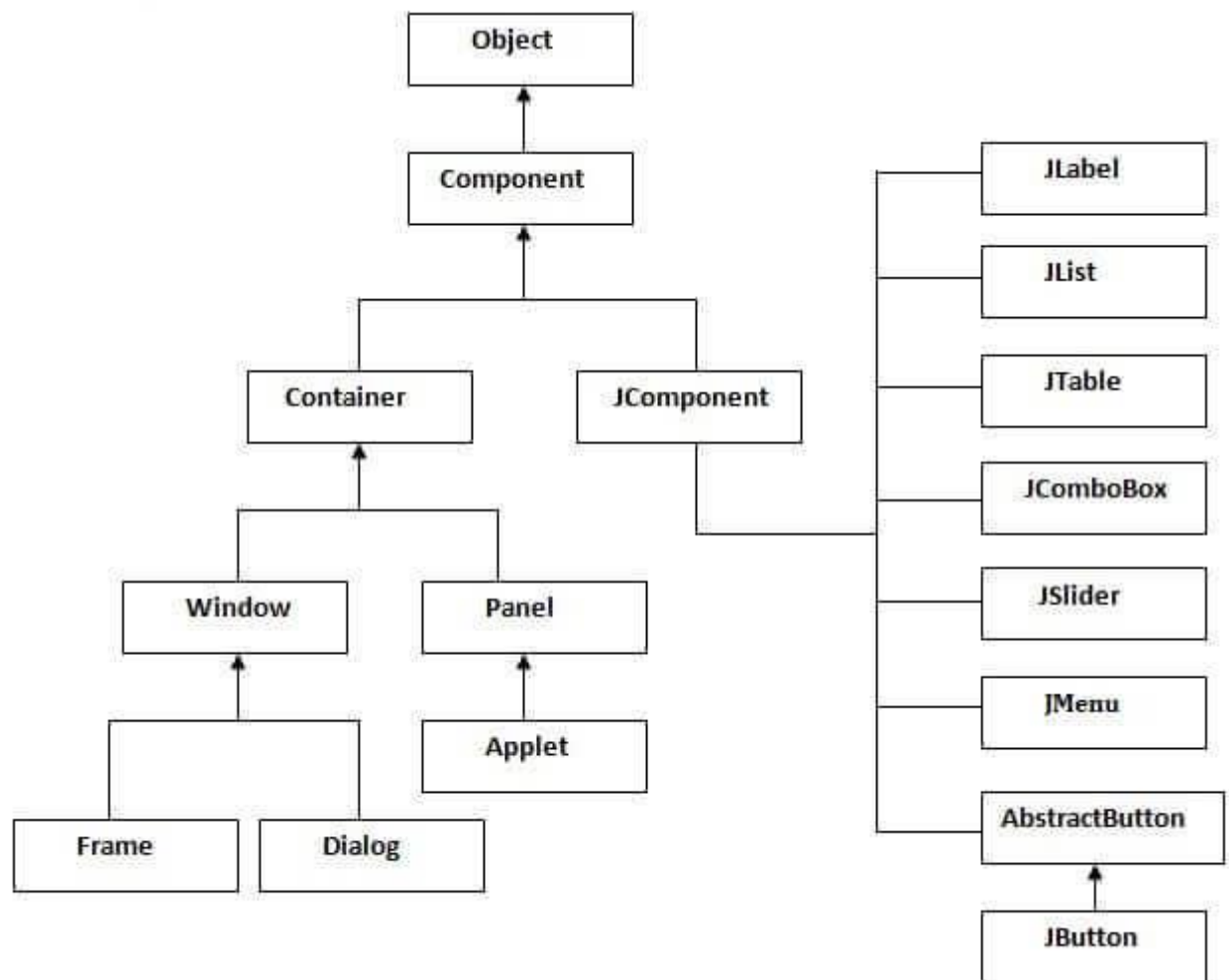
```
import java.awt.*;

import java.applet.*;

public class ButtonExample extends Applet
{
    public void init()
    {
```

```
    Button b1=new Button();  
    Button b2=new Button("eee");  
    Button b3=new Button("it");  
    add(b1);  
    add(b2);  
    add(b3);  
}  
}  
/*  
<applet code="ButtonExample.class" width=200 height=200>  
</applet>  
*/
```

Hierarchy of Java Swing classes



JLabel
JTextField
JButton
JToggleButton
JCheckBox
JRadioButton
JTabbedPane
JScrollPane
JList
JComboBox
swing Menus
Dialogs

JButton Example

```

import javax.swing.*;
public class FirstSwingExample {
public static void main(String[] args) {
JFrame f=new JFrame();           //creating instance of JFrame

JButton b=new JButton("click");   //creating instance of JButton
b.setBounds(130,100,100, 40);      //x axis, y axis, width, height

f.add(b);//adding button in JFrame
JButton b1=new JButton("click hello");
b1.setBounds(70,50,100,40);
f.add(b1);
f.setSize(400,500);               //400 width and 500 height
f.setLayout(null);                //using no layout managers
f.setVisible(true);               //making the frame visible
}
}

```

second method of use of frame

```

import javax.swing.*;
import java.awt.*;
public class myframe extends JFrame
{
public static void main(String[] args)
{
myframe f=new myframe();
JButton b1=new JButton("Click Here");
b1.setBounds(120,70,80,80);
f.add(b1);
JButton b2=new JButton("submit");
b2.setBounds(60,35,80,80);
f.add(b2);
f.setSize(400,400);
f.setVisible(true);
}
}

```

JLabel example

```
import javax.swing.*;
class LabelExample
{
    public static void main(String args[])
    {
        JFrame f= new JFrame("Label Example");
        JLabel l1,l2;
        l1=new JLabel("First Label.");
        l1.setBounds(50,50, 100,30);
        l2=new JLabel("Second Label.");
        l2.setBounds(50,100, 100,30);
        f.add(l1); f.add(l2);
        f.setSize(300,300);
        f.setLayout(null);
        f.setVisible(true);
    } }
```

JTextField example:

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

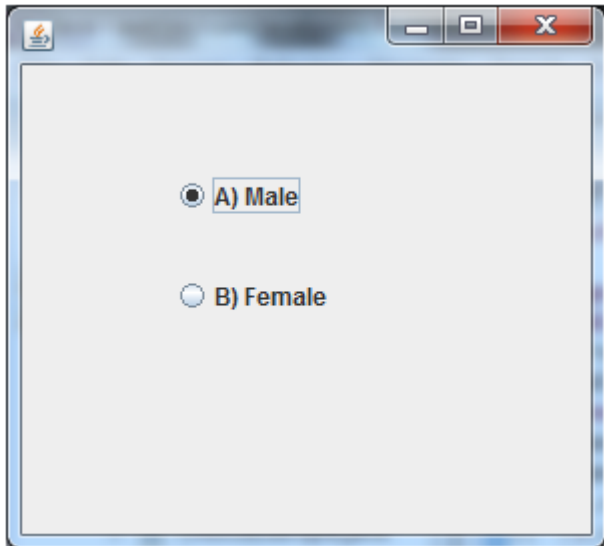
```
}
```

Java JRadioButton

The JRadioButton class is used to create a radio button. It is used to choose one option from multiple options. It is widely used in exam systems or quiz.

```
import javax.swing.*;
public class RadioButtonExample {
    JFrame f;
    RadioButtonExample(){
        f=new JFrame();
        JRadioButton r1=new JRadioButton("A) Male");
        JRadioButton r2=new JRadioButton("B) Female");
        r1.setBounds(75,50,100,30);
        r2.setBounds(75,100,100,30);
        ButtonGroup bg=new ButtonGroup();
        bg.add(r1);bg.add(r2);
        f.add(r1);f.add(r2);
        f.setSize(300,300);
        f.setLayout(null);
        f.setVisible(true);
    }
    public static void main(String[] args) {
        new RadioButtonExample();
    }
}
```

Output:

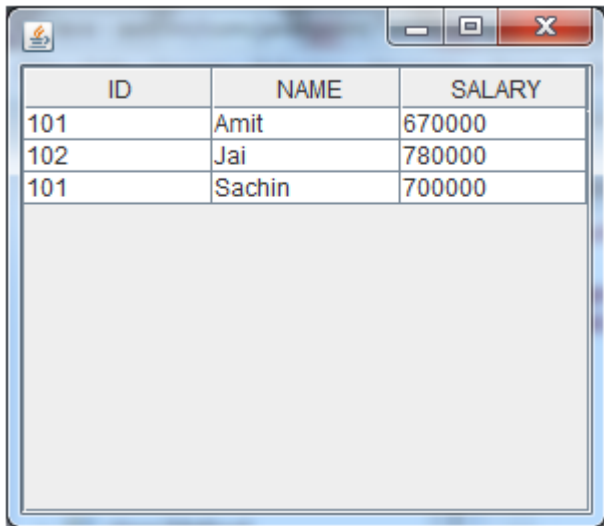


Java JTable

The JTable class is used to display data in tabular form. It is composed of rows and columns.

```
import javax.swing.*;
public class TableExample {
    JFrame f;
    TableExample(){
        f=new JFrame();
        String data[][]={ {"101","Amit","670000"},
                           {"102","Jai","780000"},
                           {"101","Sachin","700000"} };
        String column[]={ "ID","NAME","SALARY" };
        JTable jt=new JTable(data,column);
        jt.setBounds(30,40,200,300);
        JScrollPane sp=new JScrollPane(jt);
        f.add(sp);
        f.setSize(300,400);
        f.setVisible(true);
    }
    public static void main(String[] args) {
        new TableExample();
    }
}
```

Output:



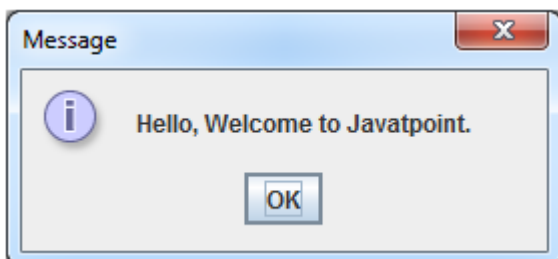
ID	NAME	SALARY
101	Amit	670000
102	Jai	780000
101	Sachin	700000

Java JOptionPane

The JOptionPane class is used to provide standard dialog boxes such as message dialog box, confirm dialog box and input dialog box. These dialog boxes are used to display information or get input from the user. The JOptionPane class inherits JComponent class.

```
import javax.swing.*;
public class OptionPaneExample {
    JFrame f;
    OptionPaneExample(){
        f=new JFrame();
        JOptionPane.showMessageDialog(f,"Hello, Welcome to Javatpoint.");
    }
    public static void main(String[] args) {
        new OptionPaneExample();
    }
}
```

Output:



Java JMenuBar, JMenu and JMenuItem

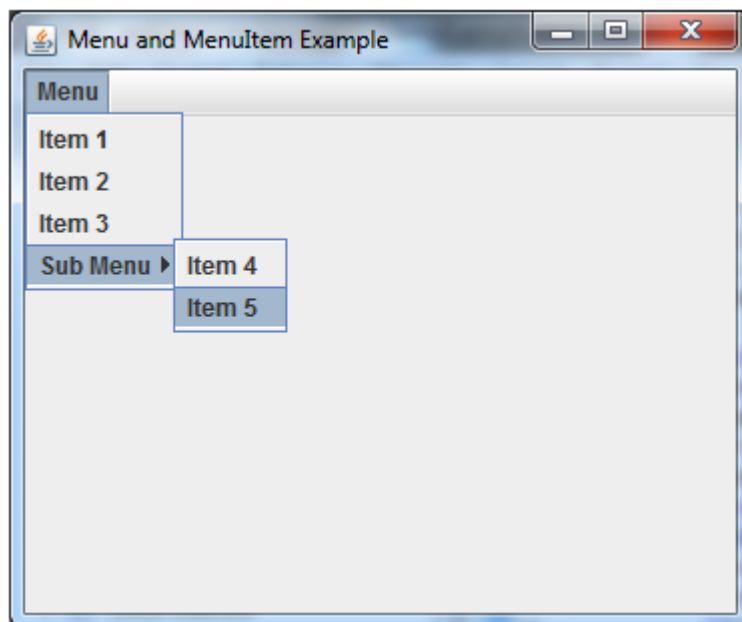
The JMenuBar class is used to display menubar on the window or frame. It may have several menus.

The object of JMenu class is a pull down menu component which is displayed from the menu bar. It inherits the JMenuItem class.

The object of JMenuItem class adds a simple labeled menu item. The items used in a menu must belong to the JMenuItem or any of its subclass.

```
1.      import javax.swing.*;
2.      class MenuExample
3.      {
4.          JMenu menu, submenu;
5.          JMenuItem i1, i2, i3, i4, i5;
6.          MenuExample(){
7.              JFrame f= new JFrame("Menu and MenuItem Example");
8.              JMenuBar mb=new JMenuBar();
9.              menu=new JMenu("Menu");
10.             submenu=new JMenu("Sub Menu");
11.             i1=new JMenuItem("Item 1");
12.             i2=new JMenuItem("Item 2");
13.             i3=new JMenuItem("Item 3");
14.             i4=new JMenuItem("Item 4");
15.             i5=new JMenuItem("Item 5");
16.             menu.add(i1); menu.add(i2); menu.add(i3);
17.             submenu.add(i4); submenu.add(i5);
18.             menu.add(submenu);
19.             mb.add(menu);
20.             f.setJMenuBar(mb);
21.             f.setSize(400,400);
22.             f.setLayout(null);
23.             f.setVisible(true);
24.         }
25.         public static void main(String args[])
26.         {
27.             new MenuExample();
28.         }}
```

Output:



Java JToggleButton

JToggleButton is used to create toggle button, it is two-states button to switch on or off.

JToggleButton(Icon icon, boolean selected)	It creates a toggle button with the specified image and selection state, but no text.
JToggleButton(String text)	It creates an unselected toggle button with the specified text.
JToggleButton(String text, boolean selected)	It creates a toggle button with the specified text and selection state.
JToggleButton(String text, Icon icon)	It creates a toggle button that has the specified text and image, and that is initially unselected.
JToggleButton(String text, Icon icon, boolean selected)	It creates a toggle button with the specified text, image, and selection state.

Methods

Modifier and Type	Method	Description
AccessibleContext	getAccessibleContext()	It gets the AccessibleContext associated with this JToggleButton.
String	getUIClassID()	It returns a string that specifies the name of the l&f class that renders this component.

protected String	paramString()	It returns a string representation of this JToggleButton.
void	updateUI()	It resets the UI property to a value from the current look and feel.

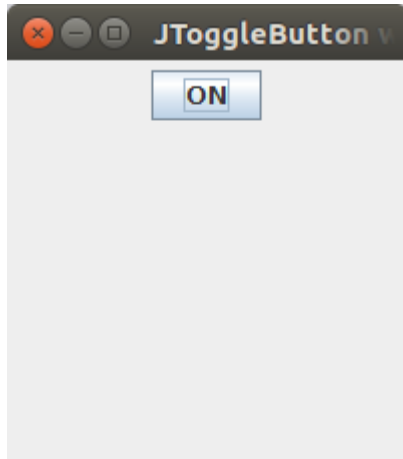
JToggleButton Example

```

1.      import java.awt.FlowLayout;
2.      import java.awt.event.ItemEvent;
3.      import java.awt.event.ItemListener;
4.      import javax.swing.JFrame;
5.      import javax.swing.JToggleButton;
6.
7.      public class JToggleButtonExample extends JFrame implements ItemLi
stener {
8.          public static void main(String[] args) {
9.              new JToggleButtonExample();
10.         }
11.         private JToggleButton button;
12.         JToggleButtonExample() {
13.             setTitle("JToggleButton with ItemListener Example");
14.             setLayout(new FlowLayout());
15.             setJToggleButton();
16.             setAction();
17.             setSize(200, 200);
18.             setVisible(true);
19.             setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20.         }
21.         private void setJToggleButton() {
22.             button = new JToggleButton("ON");
23.             add(button);
24.         }
25.         private void setAction() {
26.             button.addItemListener(this);
27.         }
28.         public void itemStateChanged(ItemEvent eve) {
29.             if (button.isSelected())
30.                 button.setText("OFF");
31.             else
32.                 button.setText("ON");
33.         }
34.     }

```

Output



Java JTabbedPane

The JTabbedPane class is used to switch between a group of components by clicking on a tab with a given title or icon. It inherits JComponent class.

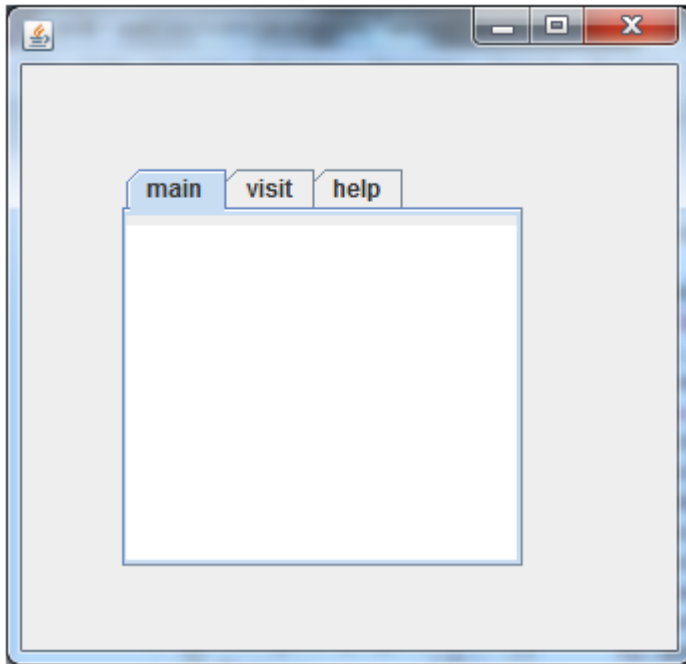
```
1.      import javax.swing.*;
2.      public class TabbedPaneExample {
3.          JFrame f;
4.          TabbedPaneExample(){
5.              f=new JFrame();
6.              JTextArea ta=new JTextArea(200,200);
7.              JPanel p1=new JPanel();
8.              p1.add(ta);
9.              JPanel p2=new JPanel();
10.             JPanel p3=new JPanel();
11.             JTabbedPane tp=new JTabbedPane();
12.             tp.setBounds(50,50,200,200);
13.             tp.add("main",p1);
14.             tp.add("visit",p2);
15.             tp.add("help",p3);
16.             f.add(tp);
17.             f.setSize(400,400);
18.             f.setLayout(null);
```

```

19.         f.setVisible(true);
20.     }
21.     public static void main(String[] args) {
22.         new TabbedPaneExample();
23.     }}

```

Output:



Java JDialog

The JDialog 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 Dialog class.

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

```

1.     import javax.swing.*;
2.     import java.awt.*;
3.     import java.awt.event.*;
4.     public class DialogExample {
5.         private static JDialog d;
6.         DialogExample() {
7.             JFrame f= new JFrame();
8.             d = new JDialog(f, "Dialog Example", true);
9.             d.setLayout( new FlowLayout() );

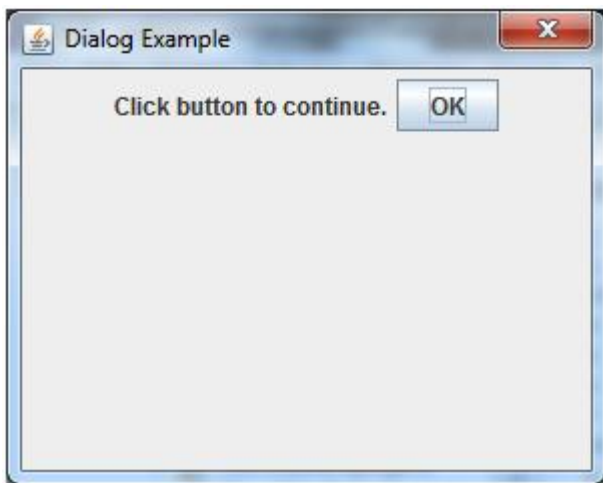
```

```

10.      JButton b = new JButton ("OK");
11.      b.addActionListener ( new ActionListener()
12.      {
13.          public void actionPerformed((ActionEvent e)
14.          {
15.              DialogExample.d.setVisible(false);
16.          }
17.      });
18.      d.add( new JLabel ("Click button to continue.));
19.      d.add(b);
20.      d.setSize(300,300);
21.      d.setVisible(true);
22.  }
23.  public static void main(String args[])
24.  {
25.      new DialogExample();
26.  }
27.  }

```

Output:

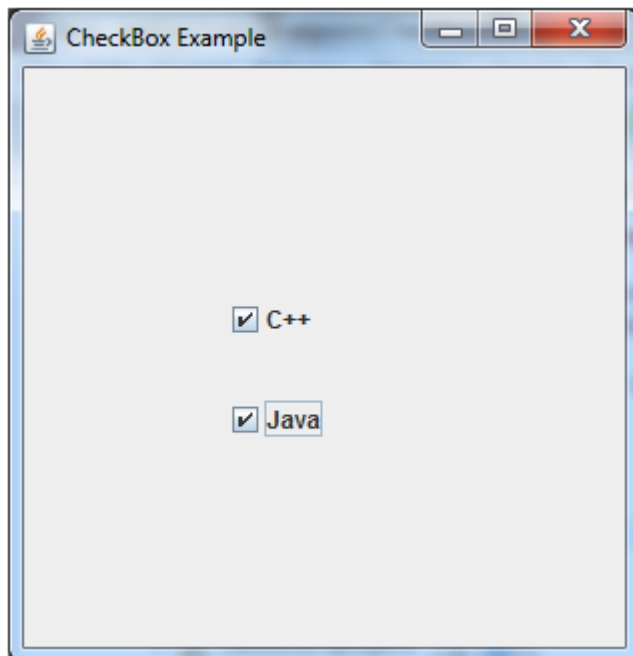


Java JCheckBox

The JCheckBox 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 ".It inherits JToggleButton class.

```
1.  import javax.swing.*;
2.  public class CheckBoxExample
3.  {
4.      CheckBoxExample(){
5.          JFrame f= new JFrame("CheckBox Example");
6.          JCheckBox checkBox1 = new JCheckBox("C++");
7.          checkBox1.setBounds(100,100, 50,50);
8.          JCheckBox checkBox2 = new JCheckBox("Java", true);
9.          checkBox2.setBounds(100,150, 50,50);
10.         f.add(checkBox1);
11.         f.add(checkBox2);
12.         f.setSize(400,400);
13.         f.setLayout(null);
14.         f.setVisible(true);
15.     }
16.     public static void main(String args[])
17.     {
18.         new CheckBoxExample();
19.     }}
```

Output:



Difference between JButton,JRadioButton,JToggleButton

Both JRadioButton and JCheckBox components can extend JToggleButton class, the main difference is that JRadioButton is a group of buttons in which only one button can be selected at a time whereas JCheckBox is a group of checkboxes in which multiple items can be selected at a time.

JButton is a nothing more than a push-button which you might already have gone through many times. User can click on it to get anything done.

A JToggleButton is another component much similar to a JButton. But the difference is that a JToggleButton goes into pressed state when mouse is pressed on it. To get that back into normal state, we need to press again. This component is mostly used when there is an on/off situation. You'll get a better idea of what it is when you see it in practical.

JApplet class in Applet

The JApplet class extends the Applet class.

```
import java.applet.*;

import javax.swing.*;

public class EventJApplet extends JApplet

{

JButton b;

JTextField tf;

public void init()

{

tf=new JTextField();

tf.setBounds(30,40,150,20);

b=new JButton("Click");
```



```
b.setBounds(80,150,70,40);
```

```
add(b);add(tf);
```

```
setLayout(null);
```

```
} }
```

```
/*<html>
```

```
<body>
```

```
<applet code="EventJApplet.class" width="300" height="300">
```

```
</applet>
```

```
</body>
```

```
</html> */
```