

Java Swing

Java Swing tutorial is a part of Java Foundation Classes (JFC) that is *used to create window-based applications*. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

Unlike AWT, Java Swing provides platform-independent and lightweight components.

The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

Difference between AWT and Swing

There are many differences between java awt and swing that are given below.

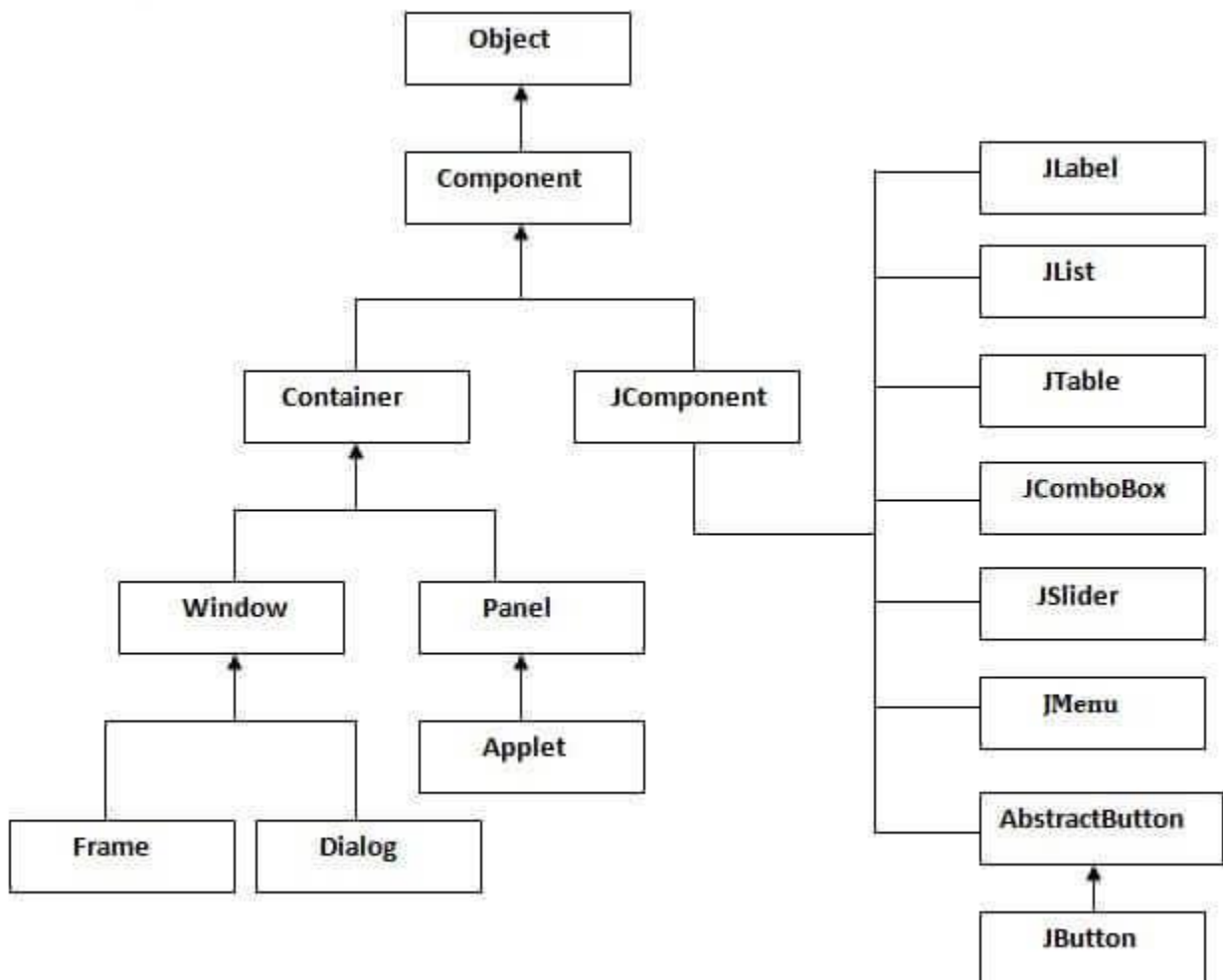
No.	Java AWT	Java Swing
1)	AWT components are platform-dependent .	Java swing components are platform-independent .
2)	AWT components are heavyweight .	Swing components are lightweight .
3)	AWT doesn't support pluggable look and feel .	Swing supports pluggable look and feel .
4)	AWT provides less components than Swing.	Swing provides more powerful components such as tables, lists, scrollpanes, colorchooser, tabbedpane etc.
5)	AWT doesn't follows MVC (Model View Controller) where model represents data, view represents presentation and controller acts as an interface between model and view.	Swing follows MVC .

What is JFC

The Java Foundation Classes (JFC) are a set of GUI components which simplify the development of desktop applications.

Hierarchy of Java Swing classes

The hierarchy of java swing API is given below.



Commonly used Methods of Component class

The methods of Component class are widely used in java swing that are given below.

Method	Description
public void add(Component c)	add a component on another component.
public void setSize(int width,int height)	sets size of the component.
public void setLayout(LayoutManager m)	sets the layout manager for the component.
public void setVisible(boolean b)	sets the visibility of the component. It is by default false.

Java Swing Examples

There are two ways to create a frame:

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

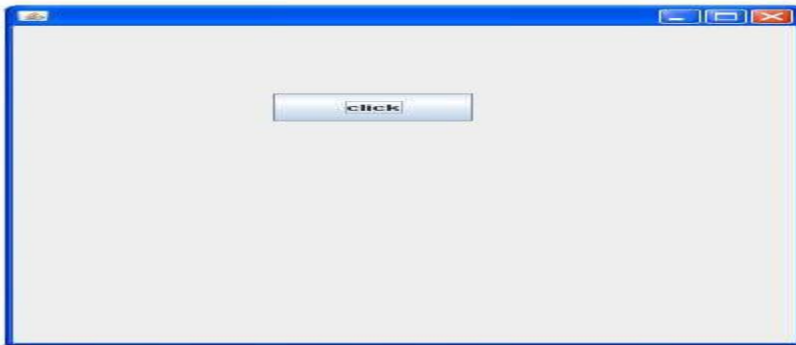
We can write the code of swing inside the main(), constructor or any other method.

Simple Java Swing Example

Let's see a simple swing example where we are creating one button and adding it on the JFrame object inside the main() method.

File: FirstSwingExample.java

1. **import** javax.swing.*;
2. **public class** FirstSwingExample {
3. **public static void** main(String[] args) {
4. JFrame f=**new** JFrame();//creating instance of JFrame
- 5.
6. JButton b=**new** JButton("click");//creating instance of JButton
7. b.setBounds(130,100,100, 40);//x axis, y axis, width, height
- 8.
9. f.add(b);//adding button in JFrame
- 10.
11. f.setSize(400,500);//400 width and 500 height
12. f.setLayout(**null**);//using no layout managers
13. f.setVisible(**true**);//making the frame visible
14. }
15. }



Example of Swing by Association inside constructor

We can also write all the codes of creating JFrame, JButton and method call inside the java constructor.

File: Simple.java

```
1. import javax.swing.*;
2. public class Simple {
3.     JFrame f;
4.     Simple(){
5.         f=new JFrame();//creating instance of JFrame
6.
7.         JButton b=new JButton("click");//creating instance of JButton
8.         b.setBounds(130,100,100, 40);
9.
10.        f.add(b);//adding button in JFrame
11.
12.        f.setSize(400,500);//400 width and 500 height
13.        f.setLayout(null);//using no layout managers
14.        f.setVisible(true);//making the frame visible
15.    }
16.
17.    public static void main(String[] args) {
18.        new Simple();
19.    }
20. }
```

The `setBounds(int xaxis, int yaxis, int width, int height)` is used in the above example that sets the position of the button.

Simple example of Swing by inheritance

We can also inherit the JFrame class, so there is no need to create the instance of JFrame class explicitly.

File: Simple2.java

```
1. import javax.swing.*;
2. public class Simple2 extends JFrame{//inheriting JFrame
3. JFrame f;
4. Simple2(){
5. JButton b=new JButton("click");//create button
6. b.setBounds(130,100,100, 40);
7.
8. add(b);//adding button on frame
9. setSize(400,500);
10. setLayout(null);
11. setVisible(true);
12. }
13. public static void main(String[] args) {
14. new Simple2();
15. }}
```

Java JButton

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

JButton class declaration

Let's see the declaration for javax.swing.JButton class.

1. **public class** JButton **extends** AbstractButton **implements** Accessible

Commonly used Constructors:

Constructor	Description
JButton()	It creates a button with no text and icon.
JButton(String s)	It creates a button with the specified text.
JButton(Icon i)	It creates a button with the specified icon object.

Commonly used Methods of AbstractButton class:

Methods	Description
void setText(String s)	It is used to set specified text on button
String getText()	It is used to return the text of the button.
void setEnabled(boolean b)	It is used to enable or disable the button.
void setIcon(Icon b)	It is used to set the specified Icon on the button.
Icon getIcon()	It is used to get the Icon of the button.
void setMnemonic(int a)	It is used to set the mnemonic on the button.
void addActionListener(ActionListener a)	It is used to add the action listener to this object.

Java JButton Example

```
1. import javax.swing.*;
2. public class ButtonExample {
3.     public static void main(String[] args) {
4.         JFrame f=new JFrame("Button Example");
5.         JButton b=new JButton("Click Here");
6.         b.setBounds(50,100,95,30);
7.         f.add(b);
8.         f.setSize(400,400);
9.         f.setLayout(null);
10.        f.setVisible(true);
11.    }
12. }
```

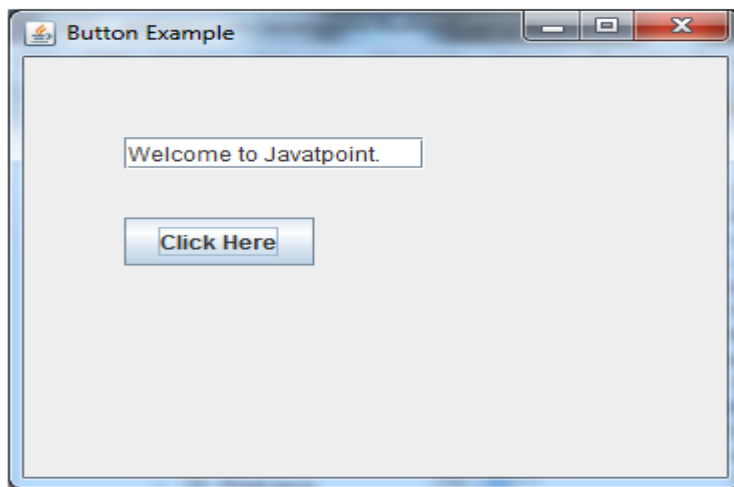
Output:



Java JButton Example with ActionListener

```
1. import java.awt.event.*;
2. import javax.swing.*;
3. public class ButtonExample {
4.     public static void main(String[] args) {
5.         JFrame f=new JFrame("Button Example");
6.         final JTextField tf=new JTextField();
7.         tf.setBounds(50,50, 150,20);
8.         JButton b=new JButton("Click Here");
9.         b.setBounds(50,100,95,30);
10.        b.addActionListener(new ActionListener(){
11.            public void actionPerformed(ActionEvent e){
12.                tf.setText("Welcome to Javatpoint.");
13.            }
14.        });
15.        f.add(b);f.add(tf);
16.        f.setSize(400,400);
17.        f.setLayout(null);
18.        f.setVisible(true);
19.    }
20. }
```

Output:



Example of displaying image on the button:

```
1. import javax.swing.*;
2. public class ButtonExample{
3.     ButtonExample(){
4.         JFrame f=new JFrame("Button Example");
5.         JButton b=new JButton(new ImageIcon("D:\\icon.png"));
6.         b.setBounds(100,100,100, 40);
7.         f.add(b);
8.         f.setSize(300,400);
9.         f.setLayout(null);
10.        f.setVisible(true);
11.        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12.    }
13.    public static void main(String[] args) {
14.        new ButtonExample();
15.    }
16. }
```

Output:



Java JTextField

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

JTextField class declaration

Let's see the declaration for javax.swing.JTextField class.

1. **public class** JTextField **extends** JTextComponent **implements** SwingConstants

Commonly used Constructors:

Constructor	Description
JTextField()	Creates a new TextField
JTextField(String text)	Creates a new TextField initialized with the specified text.
JTextField(String text, int columns)	Creates a new TextField initialized with the specified text and columns.
JTextField(int columns)	Creates a new empty TextField with the specified number of columns.

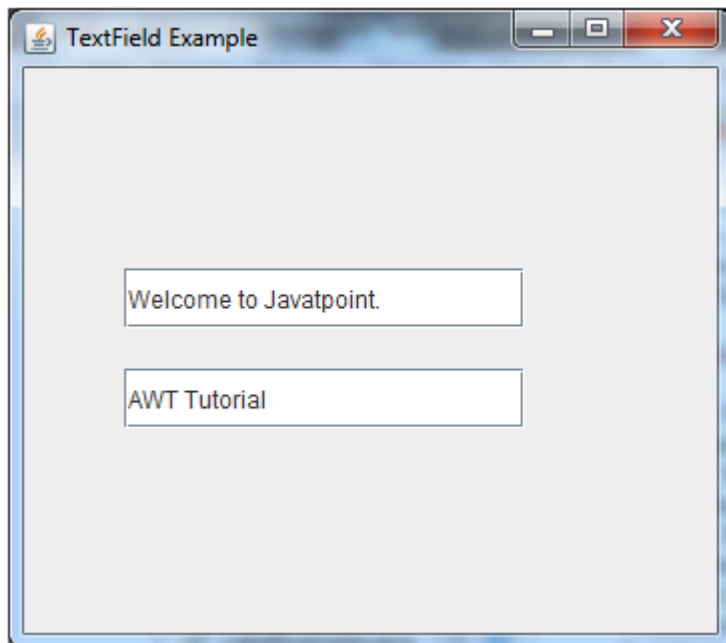
Commonly used Methods:

Methods	Description
void addActionListener(ActionListener l)	It is used to add the specified action listener to receive action events from this textfield.
Action getAction()	It returns the currently set Action for this ActionEvent source, or null if no Action is set.
void setFont(Font f)	It is used to set the current font.
void removeActionListener(ActionListener l)	It is used to remove the specified action listener so that it no longer receives action events from this textfield.

Java JTextField Example

```
1. import javax.swing.*;
2. class TextFieldExample
3. {
4.     public static void main(String args[])
5.     {
6.         JFrame f= new JFrame("TextField Example");
7.         JTextField t1,t2;
8.         t1=new JTextField("Welcome to Javatpoint.");
9.         t1.setBounds(50,100, 200,30);
10.        t2=new JTextField("AWT Tutorial");
11.        t2.setBounds(50,150, 200,30);
12.        f.add(t1); f.add(t2);
13.        f.setSize(400,400);
14.        f.setLayout(null);
15.        f.setVisible(true);
16.    }
17. }
```

Output:

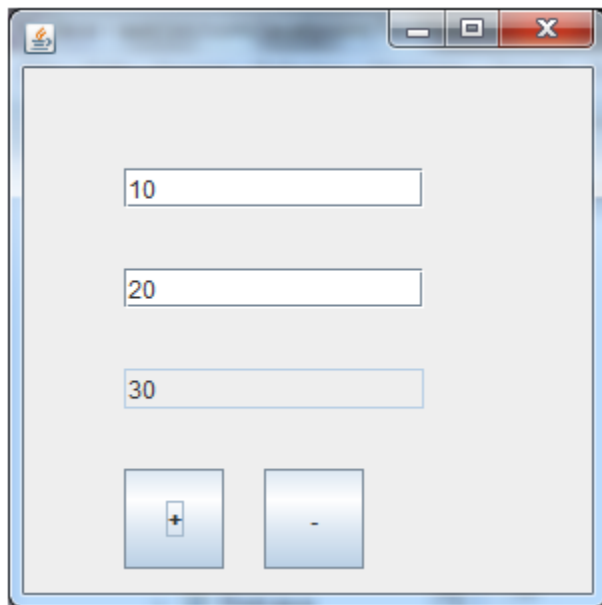


Java JTextField Example with ActionListener

```
1. import javax.swing.*;
2. import java.awt.event.*;
3. public class TextFieldExample implements ActionListener{
4.     JTextField tf1,tf2,tf3;
5.     JButton b1,b2;
6.     TextFieldExample(){
7.         JFrame f= new JFrame();
8.         tf1=new JTextField();
9.         tf1.setBounds(50,50,150,20);
10.        tf2=new JTextField();
11.        tf2.setBounds(50,100,150,20);
12.        tf3=new JTextField();
13.        tf3.setBounds(50,150,150,20);
14.        tf3.setEditable(false);
15.        b1=new JButton("+");
16.        b1.setBounds(50,200,50,50);
17.        b2=new JButton("-");
18.        b2.setBounds(120,200,50,50);
19.        b1.addActionListener(this);
20.        b2.addActionListener(this);
21.        f.add(tf1);f.add(tf2);f.add(tf3);f.add(b1);f.add(b2);
22.        f.setSize(300,300);
23.        f.setLayout(null);
24.        f.setVisible(true);
25.    }
26.    public void actionPerformed(ActionEvent e) {
27.        String s1=tf1.getText();
28.        String s2=tf2.getText();
29.        int a=Integer.parseInt(s1);
30.        int b=Integer.parseInt(s2);
31.        int c=0;
32.        if(e.getSource()==b1){
33.            c=a+b;
```

```
34.     }else if(e.getSource()==b2){
35.         c=a-b;
36.     }
37.     String result=String.valueOf(c);
38.     tf3.setText(result);
39. }
40. public static void main(String[] args) {
41.     new TextFieldExample();
42. }
```

Output:



Java JPasswordField

The object of a JPasswordField class is a text component specialized for password entry. It allows the editing of a single line of text. It inherits JTextField class.

JPasswordField class declaration

Let's see the declaration for javax.swing.JPasswordField class.

1. **public class** JPasswordField **extends** JTextField

Commonly used Constructors:

Constructor	Description
JPasswordField()	Constructs a new JPasswordField, with a default document, null starting text string, and 0 column width.
JPasswordField(int columns)	Constructs a new empty JPasswordField with the specified number of columns.
JPasswordField(String text)	Constructs a new JPasswordField initialized with the specified text.
JPasswordField(String text, int columns)	Construct a new JPasswordField initialized with the specified text and columns.

Java JPasswordField Example

```
1. import javax.swing.*;
2. public class PasswordFieldExample {
3.     public static void main(String[] args) {
4.         JFrame f=new JFrame("Password Field Example");
5.         JPasswordField value = new JPasswordField();
6.         JLabel l1=new JLabel("Password:");
7.         l1.setBounds(20,100, 80,30);
8.         value.setBounds(100,100,100,30);
9.         f.add(value); f.add(l1);
10.        f.setSize(300,300);
11.        f.setLayout(null);
12.        f.setVisible(true);
13.    }
14. }
```

Output:



Java JPasswordField Example with ActionListener

```
1. import javax.swing.*;
2. import java.awt.event.*;
3. public class PasswordFieldExample {
4.     public static void main(String[] args) {
5.         JFrame f=new JFrame("Password Field Example");
6.         final JLabel label = new JLabel();
7.         label.setBounds(20,150, 200,50);
8.         final JPasswordField value = new JPasswordField();
9.         value.setBounds(100,75,100,30);
10.        JLabel l1=new JLabel("Username:");
11.        l1.setBounds(20,20, 80,30);
12.        JLabel l2=new JLabel("Password:");
13.        l2.setBounds(20,75, 80,30);
14.        JButton b = new JButton("Login");
15.        b.setBounds(100,120, 80,30);
16.        final JTextField text = new JTextField();
17.        text.setBounds(100,20, 100,30);
18.        f.add(value); f.add(l1); f.add(label); f.add(l2); f.add(b); f.add(text);
19.        f.setSize(300,300);
20.        f.setLayout(null);
21.        f.setVisible(true);
22.        b.addActionListener(new ActionListener() {
23.            public void actionPerformed(ActionEvent e) {
24.                String data = "Username " + text.getText();
25.                data += ", Password: ";
26.                + new String(value.getPassword());
27.                label.setText(data);
28.            }
29.        });
30.    }
31. }
```

Output:

Password Field Example

Username:

Password:

Username Nakul , Password: 1234

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.

JCheckBox class declaration

Let's see the declaration for javax.swing.JCheckBox class.

1. **public class** JCheckBox **extends** JToggleButton **implements** Accessible

Commonly used Constructors:

Constructor	Description
JCheckBox()	Creates an initially unselected check box button with no text, no icon.
JChechBox(String s)	Creates an initially unselected check box with text.
JCheckBox(String text, boolean selected)	Creates a check box with text and specifies whether or not it is initially selected.
JCheckBox(Action a)	Creates a check box where properties are taken from the Action supplied.

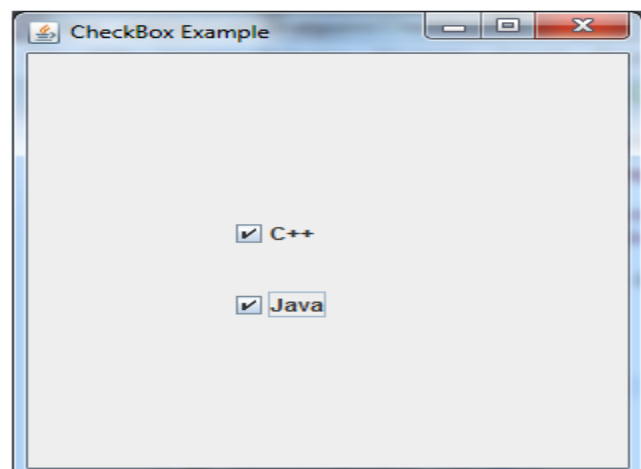
Commonly used Methods:

Methods	Description
AccessibleContext getAccessibleContext()	It is used to get the AccessibleContext associated with this JCheckBox.
protected String paramString()	It returns a <u>string</u> representation of this JCheckBox.

Java JCheckBox Example

```
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:

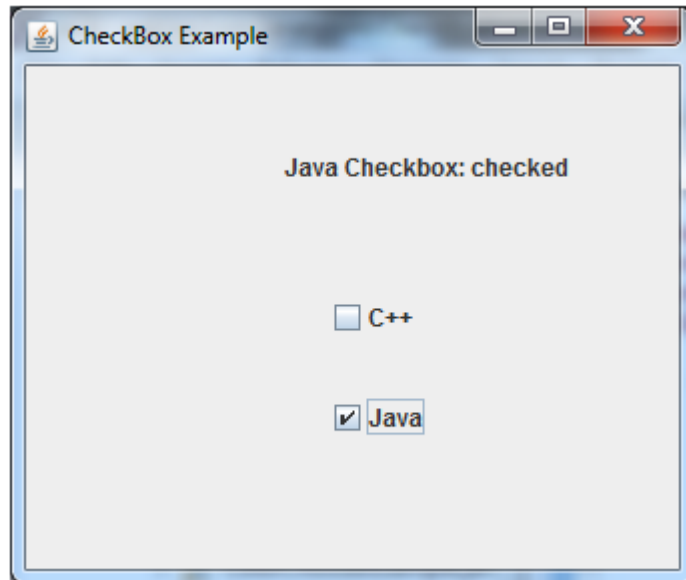


Java JCheckBox Example with ItemListener

```
1. import javax.swing.*;
2. import java.awt.event.*;
3. public class CheckBoxExample
4. {
5.     CheckBoxExample(){
6.         JFrame f= new JFrame("CheckBox Example");
7.         final JLabel label = new JLabel();
8.         label.setHorizontalAlignment(JLabel.CENTER);
9.         label.setSize(400,100);
10.        JCheckBox checkbox1 = new JCheckBox("C++");
11.        checkbox1.setBounds(150,100, 50,50);
12.        JCheckBox checkbox2 = new JCheckBox("Java");
13.        checkbox2.setBounds(150,150, 50,50);
14.        f.add(checkbox1); f.add(checkbox2); f.add(label);
15.        checkbox1.addItemListener(new ItemListener() {
16.            public void itemStateChanged(ItemEvent e) {
17.                label.setText("C++ Checkbox: "
18.                    + (e.getStateChange() == 1?"checked":"unchecked"));
19.            }
20.        });
21.        checkbox2.addItemListener(new ItemListener() {
22.            public void itemStateChanged(ItemEvent e) {
23.                label.setText("Java Checkbox: "
24.                    + (e.getStateChange() == 1?"checked":"unchecked"));
25.            }
26.        });
27.        f.setSize(400,400);
28.        f.setLayout(null);
29.        f.setVisible(true);
30.    }
31.    public static void main(String args[])
32.    {
```

```
33. new CheckBoxExample();  
34. }  
35. }
```

Output:



Java JCheckBox Example: Food Order

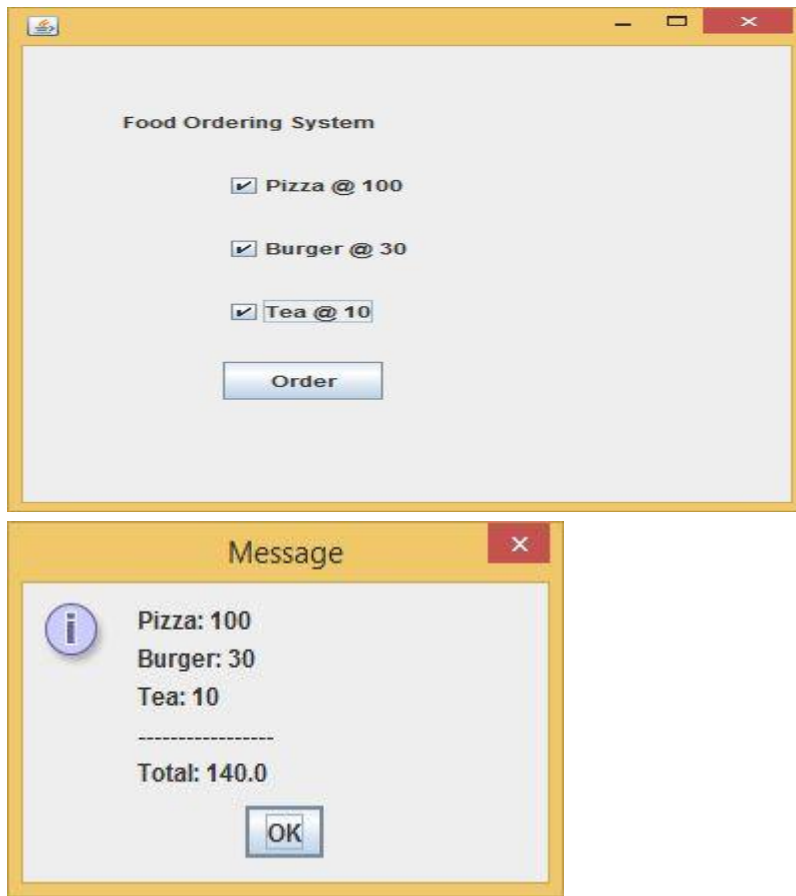
```
1. import javax.swing.*;
2. import java.awt.event.*;
3. public class CheckBoxExample extends JFrame implements ActionListener{
4.     JLabel l;
5.     JCheckBox cb1,cb2,cb3;
6.     JButton b;
7.     CheckBoxExample(){
8.         l=new JLabel("Food Ordering System");
9.         l.setBounds(50,50,300,20);
10.        cb1=new JCheckBox("Pizza @ 100");
11.        cb1.setBounds(100,100,150,20);
12.        cb2=new JCheckBox("Burger @ 30");
13.        cb2.setBounds(100,150,150,20);
14.        cb3=new JCheckBox("Tea @ 10");
15.        cb3.setBounds(100,200,150,20);
16.        b=new JButton("Order");
17.        b.setBounds(100,250,80,30);
18.        b.addActionListener(this);
19.        add(l);add(cb1);add(cb2);add(cb3);add(b);
20.        setSize(400,400);
21.        setLayout(null);
22.        setVisible(true);
23.        setDefaultCloseOperation(EXIT_ON_CLOSE);
24.    }
25.    public void actionPerformed(ActionEvent e){
26.        float amount=0;
27.        String msg="";
28.        if(cb1.isSelected()){
29.            amount+=100;
30.            msg="Pizza: 100\n";
31.        }
32.        if(cb2.isSelected()){
33.            amount+=30;
```

```

34.     msg+="Burger: 30\n";
35. }
36. if(cb3.isSelected()){
37.     amount+=10;
38.     msg+="Tea: 10\n";
39. }
40. msg+="-----\n";
41. JOptionPane.showMessageDialog(this,msg+"Total: "+amount);
42. }
43. public static void main(String[] args) {
44.     new CheckBoxExample();
45. }
46. }

```

Output:



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.

It should be added in ButtonGroup to select one radio button only.

JRadioButton class declaration

Let's see the declaration for javax.swing.JRadioButton class.

1. **public class** JRadioButton **extends** JToggleButton **implements** Accessible

Commonly used Constructors:

Constructor	Description
JRadioButton()	Creates an unselected radio button with no text.
JRadioButton(String s)	Creates an unselected radio button with specified text.
JRadioButton(String s, boolean selected)	Creates a radio button with the specified text and selected status.

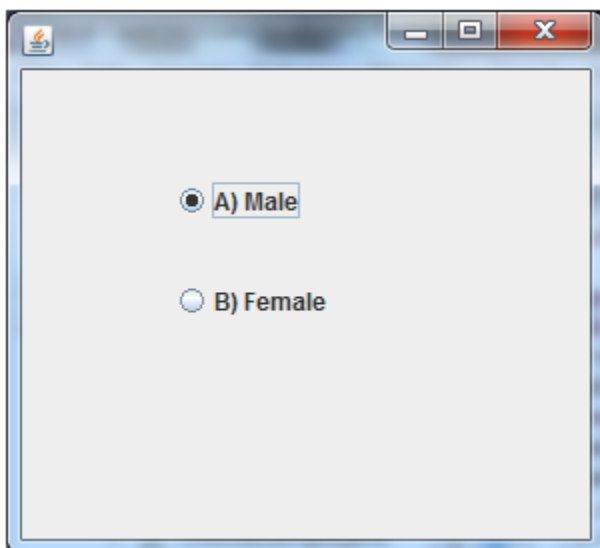
Commonly used Methods:

Methods	Description
<code>void setText(String s)</code>	It is used to set specified text on button.
<code>String getText()</code>	It is used to return the text of the button.
<code>void setEnabled(boolean b)</code>	It is used to enable or disable the button.
<code>void setIcon(Icon b)</code>	It is used to set the specified Icon on the button.
<code>Icon getIcon()</code>	It is used to get the Icon of the button.
<code>void setMnemonic(int a)</code>	It is used to set the mnemonic on the button.
<code>void addActionListener(ActionListener a)</code>	It is used to add the action listener to this object.

Java JRadioButton Example

```
1. import javax.swing.*;
2. public class RadioButtonExample {
3.     JFrame f;
4.     RadioButtonExample(){
5.         f=new JFrame();
6.         JRadioButton r1=new JRadioButton("A) Male");
7.         JRadioButton r2=new JRadioButton("B) Female");
8.         r1.setBounds(75,50,100,30);
9.         r2.setBounds(75,100,100,30);
10.        ButtonGroup bg=new ButtonGroup();
11.        bg.add(r1);bg.add(r2);
12.        f.add(r1);f.add(r2);
13.        f.setSize(300,300);
14.        f.setLayout(null);
15.        f.setVisible(true);
16.    }
17.    public static void main(String[] args) {
18.        new RadioButtonExample();
19.    }
20. }
```

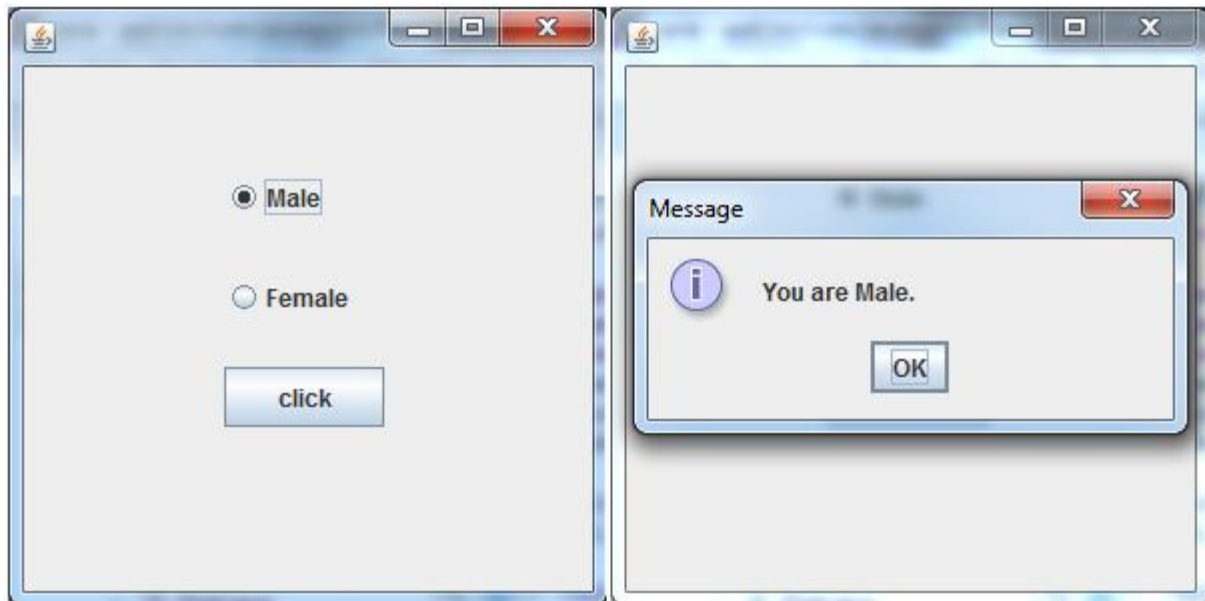
Output:



Java JRadioButton Example with ActionListener

```
1. import javax.swing.*;
2. import java.awt.event.*;
3. class RadioButtonExample extends JFrame implements ActionListener{
4.     JRadioButton rb1,rb2;
5.     JButton b;
6.     RadioButtonExample(){
7.         rb1=new JRadioButton("Male");
8.         rb1.setBounds(100,50,100,30);
9.         rb2=new JRadioButton("Female");
10.        rb2.setBounds(100,100,100,30);
11.        ButtonGroup bg=new ButtonGroup();
12.        bg.add(rb1);bg.add(rb2);
13.        b=new JButton("click");
14.        b.setBounds(100,150,80,30);
15.        b.addActionListener(this);
16.        add(rb1);add(rb2);add(b);
17.        setSize(300,300);
18.        setLayout(null);
19.        setVisible(true);
20.    }
21.    public void actionPerformed(ActionEvent e){
22.        if(rb1.isSelected()){
23.            JOptionPane.showMessageDialog(this,"You are Male.");
24.        }
25.        if(rb2.isSelected()){
26.            JOptionPane.showMessageDialog(this,"You are Female.");
27.        }
28.    }
29.    public static void main(String args[]){
30.        new RadioButtonExample();
31.    }
```

Output:



Java JComboBox

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 [JComponent](#) class.

JComboBox class declaration

Let's see the declaration for javax.swing.JComboBox class.

1. **public class** JComboBox **extends** JComponent **implements** ItemSelectable, ListDataListener, ActionListener, Accessible

Commonly used Constructors:

Constructor	Description
JComboBox()	Creates a JComboBox with a default data model.
JComboBox(Object[] items)	Creates a JComboBox that contains the elements in the specified array .
JComboBox(Vector<?> items)	Creates a JComboBox that contains the elements in the specified Vector .

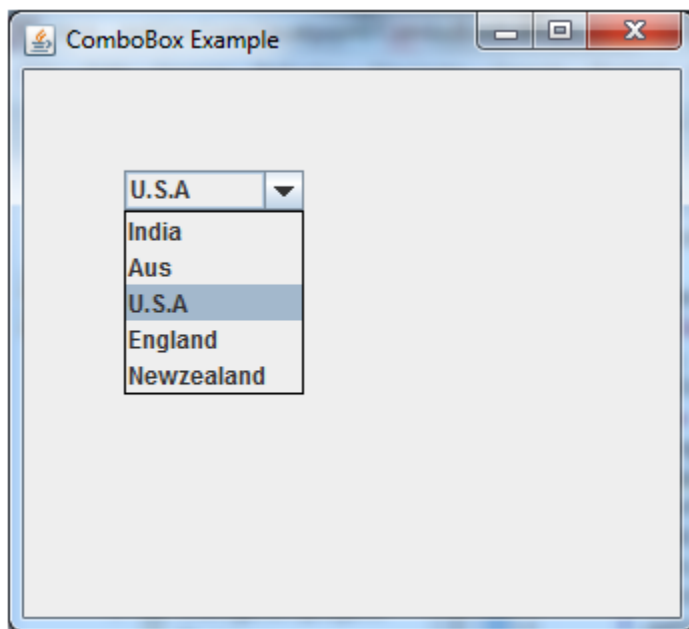
Commonly used Methods:

Methods	Description
void addItem(Object anObject)	It is used to add an item to the item list.
void removeItem(Object anObject)	It is used to delete an item to the item list.
void removeAllItems()	It is used to remove all the items from the list.
void setEditable(boolean b)	It is used to determine whether the JComboBox is editable.
void addActionListener(ActionListener a)	It is used to add the ActionListener .
void addItemListener(ItemListener i)	It is used to add the ItemListener .

Java JComboBox Example

```
1. import javax.swing.*;
2. public class ComboBoxExample {
3.     JFrame f;
4.     ComboBoxExample(){
5.         f=new JFrame("ComboBox Example");
6.         String country[]={"India","Aus","U.S.A","England","Newzealand"};
7.         JComboBox cb=new JComboBox(country);
8.         cb.setBounds(50, 50,90,20);
9.         f.add(cb);
10.        f.setLayout(null);
11.        f.setSize(400,500);
12.        f.setVisible(true);
13.    }
14.    public static void main(String[] args) {
15.        new ComboBoxExample();
16.    }
17. }
```

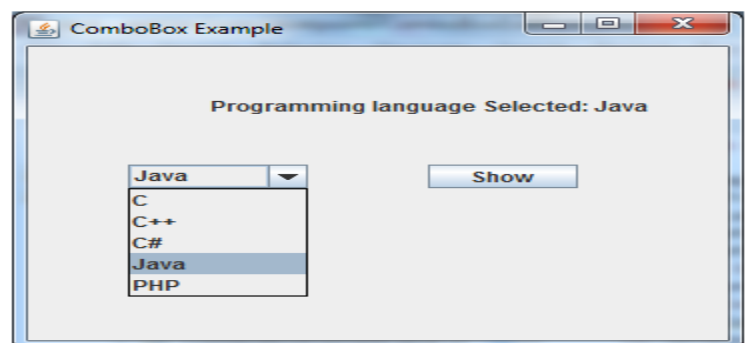
Output:



Java JComboBox Example with ActionListener

```
1. import javax.swing.*;
2. import java.awt.event.*;
3. public class ComboBoxExample {
4.     JFrame f;
5.     ComboBoxExample(){
6.         f=new JFrame("ComboBox Example");
7.         final JLabel label = new JLabel();
8.         label.setHorizontalAlignment(JLabel.CENTER);
9.         label.setSize(400,100);
10.        JButton b=new JButton("Show");
11.        b.setBounds(200,100,75,20);
12.        String languages[]={"C","C++","C#","Java","PHP"};
13.        final JComboBox cb=new JComboBox(languages);
14.        cb.setBounds(50, 100,90,20);
15.        f.add(cb); f.add(label); f.add(b);
16.        f.setLayout(null);
17.        f.setSize(350,350);
18.        f.setVisible(true);
19.        b.addActionListener(new ActionListener() {
20.            public void actionPerformed(ActionEvent e) {
21.                String data = "Programming language Selected: "
22.                + cb.getItemAt(cb.getSelectedIndex());
23.                label.setText(data);
24.            }
25.        });
26.    }
27.    public static void main(String[] args) {
28.        new ComboBoxExample();
29.    }
30. }
```

Output:



Java JTable

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

JTable class declaration

Let's see the declaration for javax.swing.JTable class.

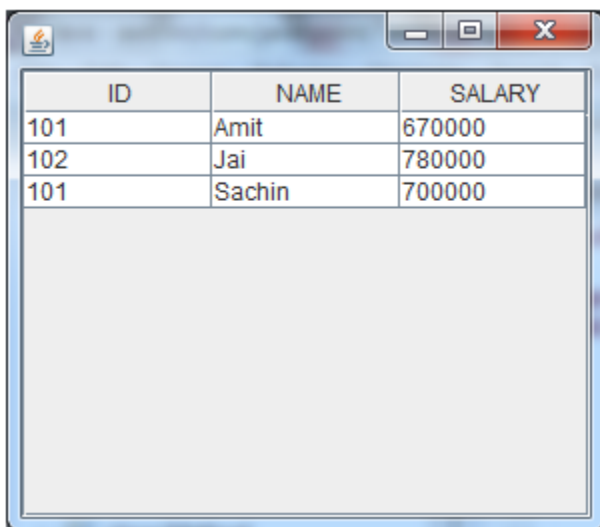
Commonly used Constructors:

Constructor	Description
JTable()	Creates a table with empty cells.
JTable(Object[][] rows, Object[] columns)	Creates a table with the specified data.

Java JTable Example

```
1. import javax.swing.*;
2. public class TableExample {
3.     JFrame f;
4.     TableExample(){
5.         f=new JFrame();
6.         String data[][]={ {"101","Amit","670000"},
7.                             {"102","Jai","780000"},
8.                             {"101","Sachin","700000"}};
9.         String column[]={ "ID","NAME","SALARY"};
10.        JTable jt=new JTable(data,column);
11.        jt.setBounds(30,40,200,300);
12.        JScrollPane sp=new JScrollPane(jt);
13.        f.add(sp);
14.        f.setSize(300,400);
15.        f.setVisible(true);
16.    }
17.    public static void main(String[] args) {
18.        new TableExample();
19.    }
20. }
```

Output:

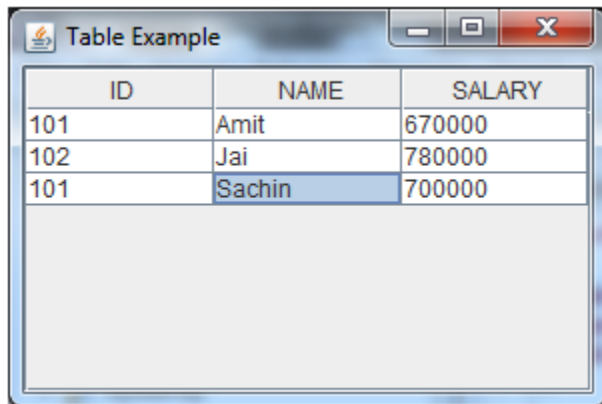


ID	NAME	SALARY
101	Amit	670000
102	Jai	780000
101	Sachin	700000

Java JTable Example with ListSelectionListener

```
1. import javax.swing.*;
2. import javax.swing.event.*;
3. public class TableExample {
4.     public static void main(String[] a) {
5.         JFrame f = new JFrame("Table Example");
6.         String data[][] = { {"101", "Amit", "670000"},
7.                               {"102", "Jai", "780000"},
8.                               {"101", "Sachin", "700000"} };
9.         String column[] = {"ID", "NAME", "SALARY"};
10.        final JTable jt = new JTable(data, column);
11.        jt.setCellSelectionEnabled(true);
12.        ListSelectionModel select = jt.getSelectionModel();
13.        select.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
14.        select.addListSelectionListener(new ListSelectionListener() {
15.            public void valueChanged(ListSelectionEvent e) {
16.                String Data = null;
17.                int[] row = jt.getSelectedRows();
18.                int[] columns = jt.getSelectedColumns();
19.                for (int i = 0; i < row.length; i++) {
20.                    for (int j = 0; j < columns.length; j++) {
21.                        Data = (String) jt.getValueAt(row[i], columns[j]);
22.                    }
23.                }
24.                System.out.println("Table element selected is: " + Data);
25.            }
26.        });
27.        JScrollPane sp = new JScrollPane(jt);
28.        f.add(sp);
29.        f.setSize(300, 200);
30.        f.setVisible(true);
31.    }
```

Output:



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

If you select an element in column **NAME**, name of the element will be displayed on the console:

1. Table element selected is: Sachin

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.

JDialog class declaration

Let's see the declaration for javax.swing.JDialog class.

1. **public class** JDialog **extends** Dialog **implements** WindowConstants, Accessible, RootPaneContainer

Commonly used Constructors:

Constructor	Description
JDialog()	It is used to create a modeless dialog without a title and without a specified Frame owner.
JDialog(Frame owner)	It is used to create a modeless dialog with specified Frame as its owner and an empty title.
JDialog(Frame owner, String title, boolean modal)	It is used to create a dialog with the specified title, owner Frame and modality.

Java JDialog Example

```
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:

