JCheckBox, JRadioButton, JList, JScrollBar, JTextArea and JTable

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.

- **public class** JCheckBox **extends** JToggleButton **implement s** Accessible

- Create JCheckBox with no label and unselected
- JCheckBox cb1 = **new** JCheckBox();

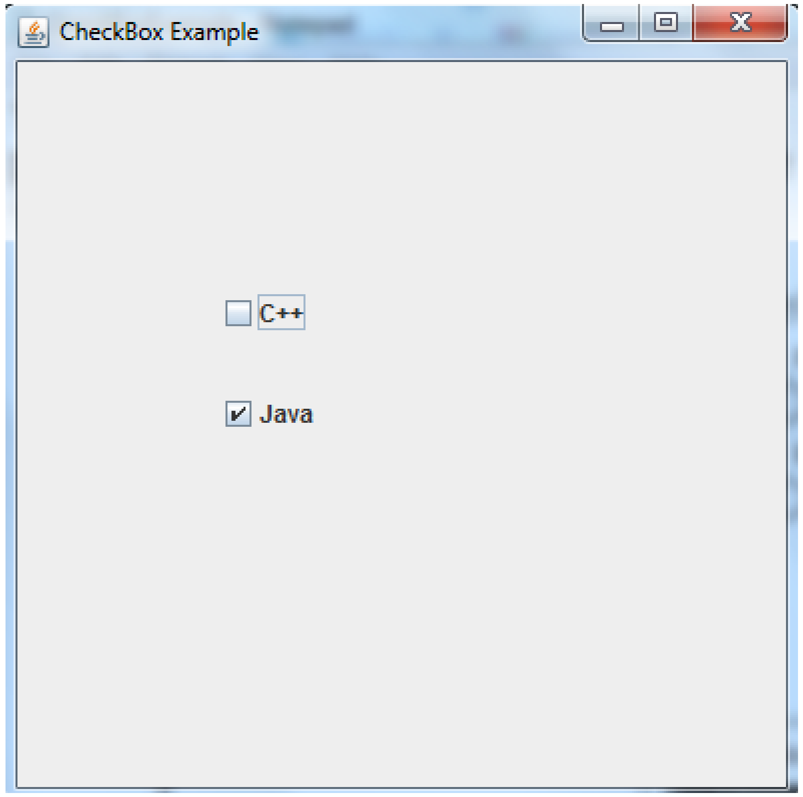
- Create JCheckBox with text as "My Choice“ and unselected
- JCheckBox cb2 = **new** JCheckBox(**"My Choice"**);

- Create JCheckBox with text as "My Choice" and selected by default
- JCheckBox cb3 = **new** JCheckBox(**"My Choice"**, true);

Example

```
import javax.swing.*;
public class CheckBoxExample
{
     CheckBoxExample(){
        JFrame f= new JFrame("CheckBox Example");
        JCheckBox checkBox1 = new JCheckBox("C++");  // unselected
        checkBox1.setBounds(100,100, 50,50);
        JCheckBox checkBox2 = new JCheckBox("Java", true);  // selected
        checkBox2.setBounds(100,150, 100,50);
        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 JRadioButton

- The JRadioButton class is used to create a radio button.
- JRadioButton is a Swing component that represents an item with a state selected or unselected.
- 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

- Declaration for javax.swing.JRadioButton class:

public class JRadioButton **extends** JToggleButton **implement s** Accessible

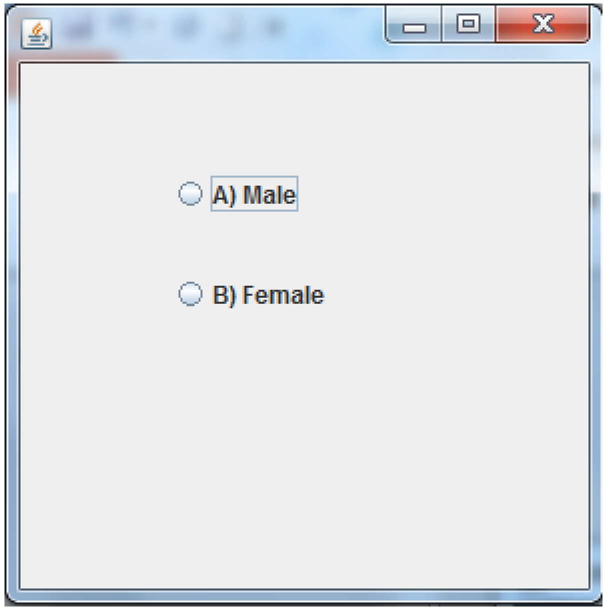
Java JRadioButton Example

```
import javax.swing.*;
public class RadioButtonExample {
JFrame f;
RadioButtonExample(){
f=new JFrame();
JRadioButton r1=new JRadioButton("A) Male");    // unselected

//JRadioButton r1=new JRadioButton("A) Male", true);   Male will be selected as true

JRadioButton r2=new JRadioButton("B) Female");    // unselected
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 JList

- The object of JList class represents a list of text items.
- The list of text items can be set up so that the user can choose either one item or multiple items.
- It inherits JComponent class.

JList class declaration

The declaration for javax.swing.JList class:

public class JList **extends** JComponent **implements** Scrollabl e, Accessible

Note:

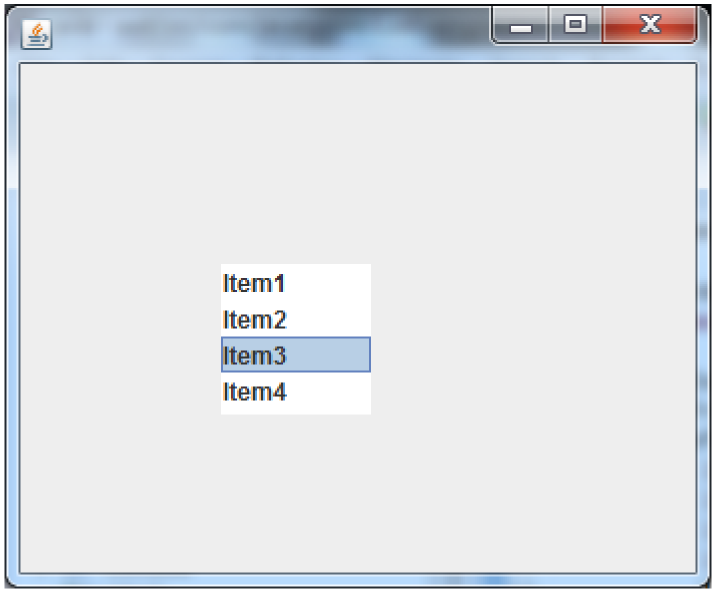
- DefaultListModel class provides a simple implementation of a *list model*, which can be used to manage items displayed by a JList control.

- When you create the default data model, it’s empty, but you can call the add or addElement() method to add elements to the list

Example

```
import javax.swing.*;
public class ListExample
{
     ListExample(){
        JFrame f= new JFrame();
        DefaultListModel<String> l1 = new DefaultListModel<>();
          l1.addElement("Item1");
          l1.addElement("Item2");
          l1.addElement("Item3");
          l1.addElement("Item4");
          JList<String> list = new JList<>(l1);
          list.setBounds(100,100, 75,75);
          f.add(list);
          f.setSize(400,400);
          f.setLayout(null);
          f.setVisible(true);
     }
public static void main(String args[])
    {
   new ListExample();
    }}
```

Output



Java JTextArea

- The object of a JTextArea class is a multi line region that displays text.
- It allows the editing of multiple line text.
- It inherits JTextComponent class

JTextArea class declaration

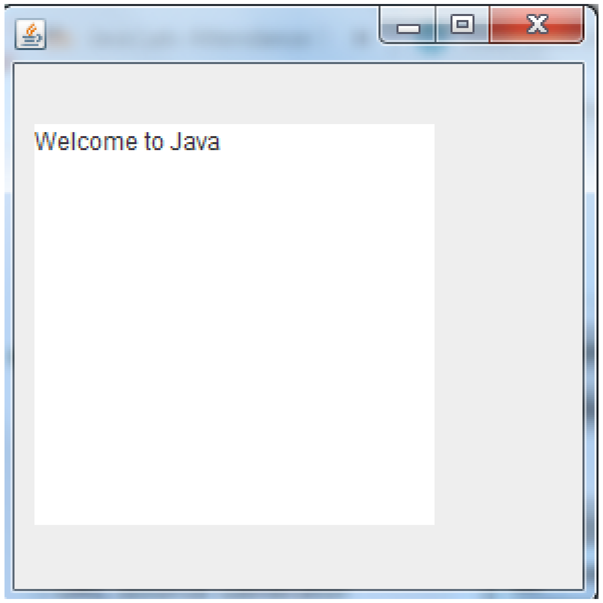
The declaration for javax.swing.JTextArea class:

public class JTextArea **extends** JTextComponent

Example

```
import javax.swing.*;
public class TextAreaExample
{
     TextAreaExample(){
        JFrame f= new JFrame();
        JTextArea area=new JTextArea("Welcome to java");
// area = new JTextArea(10, 10);  it create a text area, specifying the rows and columns
        area.setBounds(10,30, 200,200);
        f.add(area);
        f.setSize(300,300);
        f.setLayout(null);
        f.setVisible(true);
     }
public static void main(String args[])
    {
   new TextAreaExample();
    }}
```

Output



```
Welcome to Java
```

Java JScrollBar

- The object of JScrollBar class is used to add horizontal and vertical scrollbar.
- It is an implementation of a scrollbar.
- It inherits JComponent class.

JScrollBar class declaration

The declaration for javax.swing.JScrollBar class.

public class JScrollBar **extends** JComponent **implements** Adj
ustable, Accessible

Constructor	**Description**
JScrollBar()	Creates a vertical scrollbar with the initial values.
JScrollBar(int orientation)	Creates a scrollbar with the specified orientation and the initial values.
JScrollBar(int orientation, int value, int extent, int min, int max)	Creates a scrollbar with the specified orientation, value, extent, minimum, and maximum.

1. To create a JScrollBar with all default properties. Its orientation will be vertical, current value 0, extent 10, minimum 0, and maximum 100.

```
JScrollBar sb1  = new JScrollBar();
```

2. To create a horizontal JScrollBar with default values

```
JScrollBar sb2  = new JScrollBar(JScrollBar.HORIZONTAL);
```

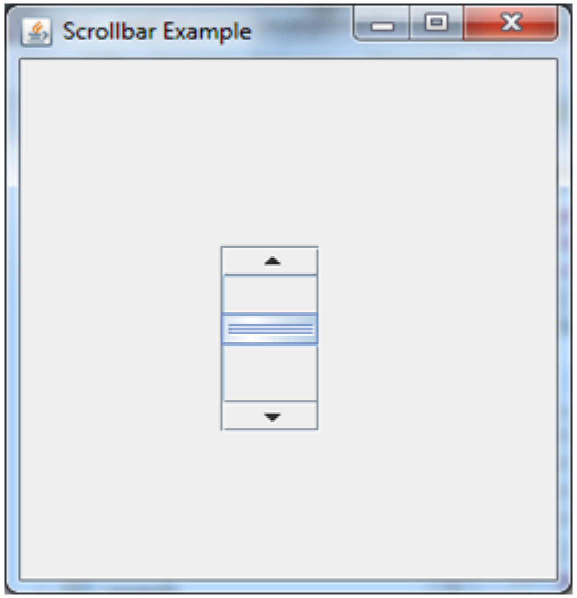
3. To create a horizontal JScrollBar with a current value of 50, extent 15, minimum 1 and maximum 150.

```
JScrollBar sb3 = new JScrollBar(JScrollBar.HORIZONTAL, 50, 15, 1, 150);
```

Example

```
import javax.swing.*;
class ScrollBarExample
{
ScrollBarExample(){
    JFrame f= new JFrame("Scrollbar Example");
 JScrollBar s=new JScrollBar();
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



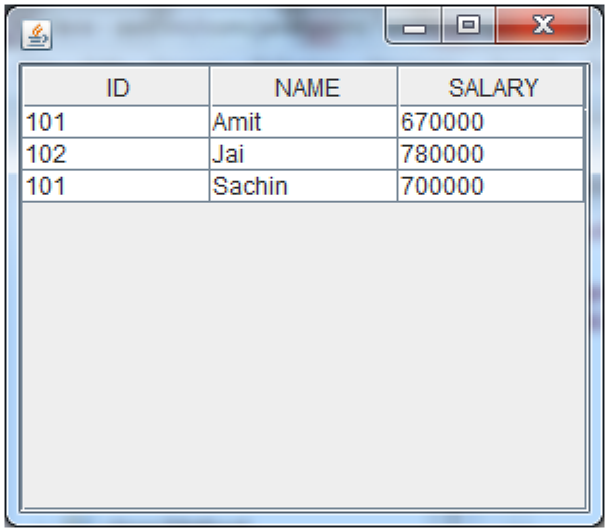
Java JTable

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

Java JTable Example

```
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