# 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](https://www.javatpoint.com/java-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** |
| JJCheckBox() | 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. |

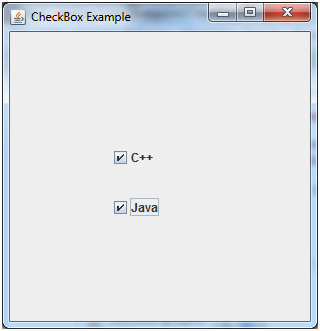
### Commonly used Methods:

|  |  |
| --- | --- |
| **Methods** | **Description** |
| AccessibleContext getAccessibleContext() | It is used to get the AccessibleContext associated with this JCheckBox. |
| protected String paramString() | It returns a [string](https://www.javatpoint.com/java-string) 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. }}

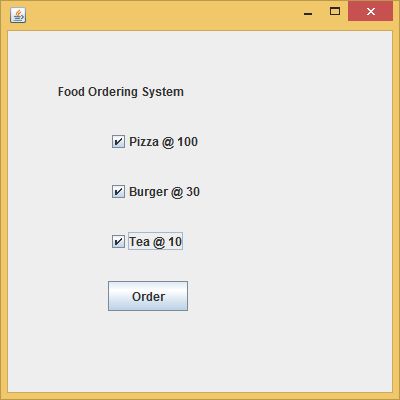
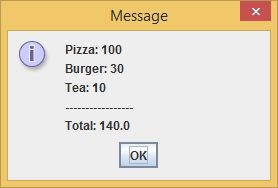
Output:



## Java JCheckBox Example: 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:

# 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. |

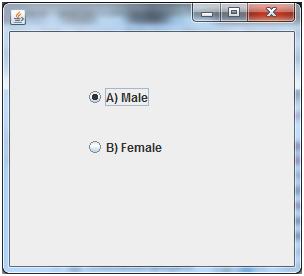
### Commonly used Methods:

|  |  |
| --- | --- |
| **Methods** | **Description** |
| 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. |

## 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:



# ava 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](https://www.javatpoint.com/java-jmenuitem-and-jmenu). It inherits [JComponent](https://www.javatpoint.com/java-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](https://www.javatpoint.com/array-in-java). |
| JComboBox(Vector<?> items) | Creates a JComboBox that contains the elements in the specified [Vector](https://www.javatpoint.com/scala-vector). |

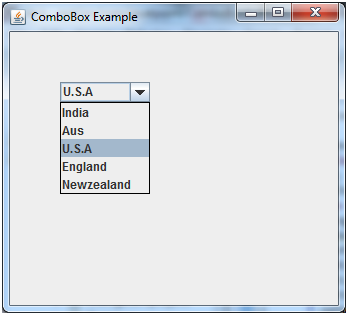
### Commonly used Methods:

|  |  |
| --- | --- |
| **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](https://www.javatpoint.com/java-actionlistener). |
| void addItemListener(ItemListener i) | It is used to add the [ItemListener](https://www.javatpoint.com/java-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:



# 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.

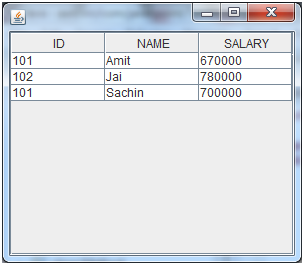
### Commonly used Constructors:

|  |  |
| --- | --- |
| **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:



# Java JPanel

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

It doesn't have title bar.

## JPanel class declaration

1. **public** **class** JPanel **extends** JComponent **implements** Accessible

### Commonly used Constructors:

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| JPanel() | It is used to create a new JPanel with a double buffer and a flow layout. |
| JPanel(boolean isDoubleBuffered) | It is used to create a new JPanel with FlowLayout and the specified buffering strategy. |
| JPanel(LayoutManager layout) | It is used to create a new JPanel with the specified layout manager. |

## Java JPanel Example

1. **import** java.awt.\*;
2. **import** javax.swing.\*;
3. **public** **class** PanelExample {
4. PanelExample()
5. {
6. JFrame f= **new** JFrame("Panel Example");
7. JPanel panel=**new** JPanel();
8. panel.setBounds(40,80,200,200);
9. panel.setBackground(Color.gray);
10. JButton b1=**new** JButton("Button 1");
11. b1.setBounds(50,100,80,30);
12. b1.setBackground(Color.yellow);
13. JButton b2=**new** JButton("Button 2");
14. b2.setBounds(100,100,80,30);
15. b2.setBackground(Color.green);
16. panel.add(b1); panel.add(b2);
17. f.add(panel);
18. f.setSize(400,400);
19. f.setLayout(**null**);
20. f.setVisible(**true**);
21. }
22. **public** **static** **void** main(String args[])
23. {
24. **new** PanelExample();
25. }
26. }

Output:

