Java Swing Tutorial

**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.

Skip Ad

|  |  |  |
| --- | --- | --- |
| **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.

Do You Know

* How to create runnable jar file in java?
* How to display image on a button in swing?
* How to change the component color by choosing a color from ColorChooser ?
* How to display the digital watch in swing tutorial ?
* How to create a notepad in swing?
* How to create puzzle game and pic puzzle game in swing ?
* How to create tic tac toe game in swing ?

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.

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)

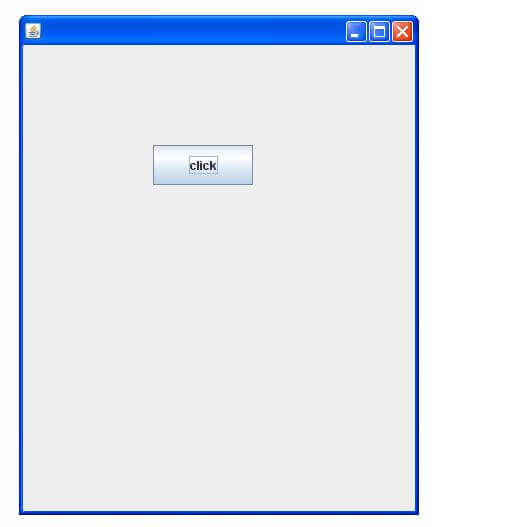
We can write the code of swing inside the main(), constructor or any other method.

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
6. JButton b=**new** JButton("click");//creating instance of JButton
7. b.setBounds(130,100,100, 40);//x axis, y axis, width, height
9. f.add(b);//adding button in JFrame
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.

*ile: Simple.java*

1. **import** javax.swing.\*;
2. **public** **class** Simple {
3. JFrame f;
4. Simple(){
5. f=**new** JFrame();//creating instance of JFrame
7. JButton b=**new** JButton("click");//creating instance of JButton
8. b.setBounds(130,100,100, 40);
10. f.add(b);//adding button in JFrame
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. }
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);
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. }}

[download this example](https://static.javatpoint.com/src/swing/first2.zip)

*What we will learn in Swing Tutorial*

* JButton class
* JRadioButton class
* JTextArea class
* JComboBox class
* JTable class
* JColorChooser class
* JProgressBar class
* JSlider class
* Digital Watch
* Graphics in swing
* Displaying image
* Edit menu code for Notepad
* OpenDialog Box
* Notepad
* Puzzle Game
* Pic Puzzle Game
* Tic Tac Toe Game
* BorderLayout
* GridLayout
* FlowLayout
* CardLayout

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. |

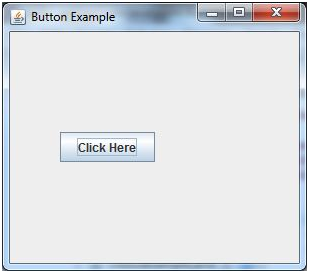
Commonly used Methods of AbstractButton class:

|  |  |
| --- | --- |
| **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](https://www.javatpoint.com/java-actionlistener) 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. }

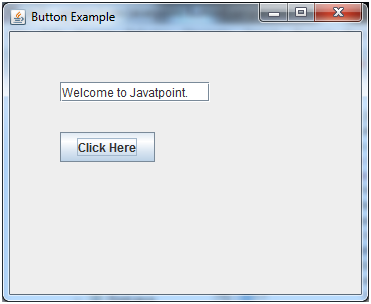
Output:



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. }

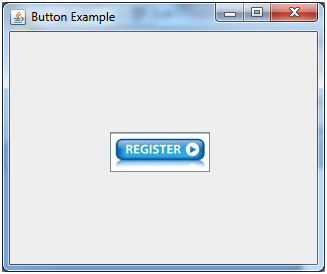
Output:



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. |

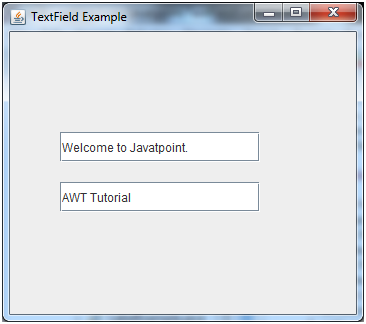
Commonly used Methods:

|  |  |
| --- | --- |
| **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. }

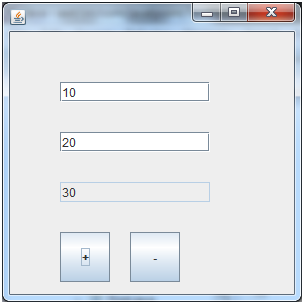
Output:



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 JLabel

The object of JLabel 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. It inherits JComponent class.

JLabel class declaration

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

1. **public** **class** JLabel **extends** JComponent **implements** SwingConstants, Accessible

Commonly used Constructors:

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| JLabel() | Creates a JLabel instance with no image and with an empty string for the title. |
| JLabel(String s) | Creates a JLabel instance with the specified text. |
| JLabel(Icon i) | Creates a JLabel instance with the specified image. |
| JLabel(String s, Icon i, int horizontalAlignment) | Creates a JLabel instance with the specified text, image, and horizontal alignment. |

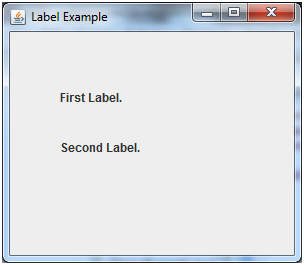
Commonly used Methods:

|  |  |
| --- | --- |
| **Methods** | **Description** |
| String getText() | t returns the text string that a label displays. |
| void setText(String text) | It defines the single line of text this component will display. |
| void setHorizontalAlignment(int alignment) | It sets the alignment of the label's contents along the X axis. |
| Icon getIcon() | It returns the graphic image that the label displays. |
| int getHorizontalAlignment() | It returns the alignment of the label's contents along the X axis. |

Java JLabel Example

1. **import** javax.swing.\*;
2. **class** LabelExample
3. {
4. **public** **static** **void** main(String args[])
5. {
6. JFrame f= **new** JFrame("Label Example");
7. JLabel l1,l2;
8. l1=**new** JLabel("First Label.");
9. l1.setBounds(50,50, 100,30);
10. l2=**new** JLabel("Second Label.");
11. l2.setBounds(50,100, 100,30);
12. f.add(l1); f.add(l2);
13. f.setSize(300,300);
14. f.setLayout(**null**);
15. f.setVisible(**true**);
16. }
17. }

Output:



Java JLabel Example with ActionListener

1. **import** javax.swing.\*;
2. **import** java.awt.\*;
3. **import** java.awt.event.\*;
4. **public** **class** LabelExample **extends** Frame **implements** ActionListener{
5. JTextField tf; JLabel l; JButton b;
6. LabelExample(){
7. tf=**new** JTextField();
8. tf.setBounds(50,50, 150,20);
9. l=**new** JLabel();
10. l.setBounds(50,100, 250,20);
11. b=**new** JButton("Find IP");
12. b.setBounds(50,150,95,30);
13. b.addActionListener(**this**);
14. add(b);add(tf);add(l);
15. setSize(400,400);
16. setLayout(**null**);
17. setVisible(**true**);
18. }
19. **public** **void** actionPerformed(ActionEvent e) {
20. **try**{
21. String host=tf.getText();
22. String ip=java.net.InetAddress.getByName(host).getHostAddress();
23. l.setText("IP of "+host+" is: "+ip);
24. }**catch**(Exception ex){System.out.println(ex);}
25. }
26. **public** **static** **void** main(String[] args) {
27. **new** LabelExample();
28. } }

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. |

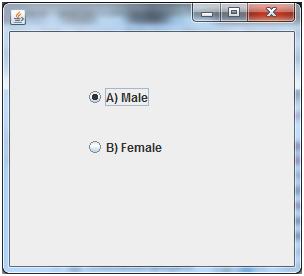
Commonly used Methods:

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

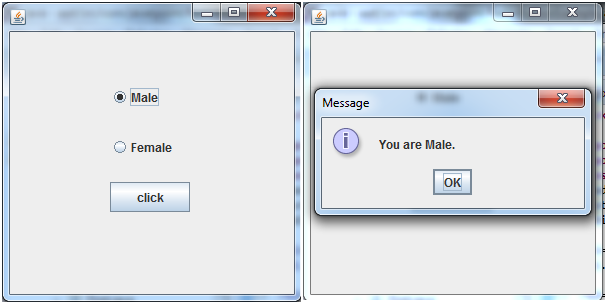
Output:



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](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). |

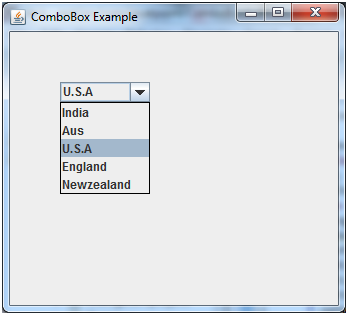
Commonly used Methods:

|  |  |
| --- | --- |
| **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. }

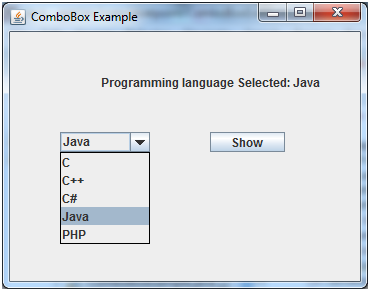
Output:



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 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

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

1. **public** **class** JList **extends** JComponent **implements** Scrollable, Accessible

Commonly used Constructors:

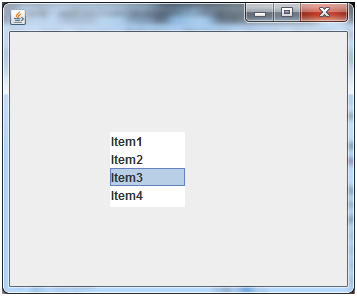
|  |  |
| --- | --- |
| **Constructor** | **Description** |
| JList() | Creates a JList with an empty, read-only, model. |
| JList(ary[] listData) | Creates a JList that displays the elements in the specified array. |
| JList(ListModel<ary> dataModel) | Creates a JList that displays elements from the specified, non-null, model. |

Commonly used Methods:

|  |  |
| --- | --- |
| **Methods** | **Description** |
| Void addListSelectionListener(ListSelectionListener listener) | It is used to add a listener to the list, to be notified each time a change to the selection occurs. |
| int getSelectedIndex() | It is used to return the smallest selected cell index. |
| ListModel getModel() | It is used to return the data model that holds a list of items displayed by the JList component. |
| void setListData(Object[] listData) | It is used to create a read-only ListModel from an array of objects. |

1. Java JList Example
2. **import** javax.swing.\*;
3. **public** **class** ListExample
4. {
5. ListExample(){
6. JFrame f= **new** JFrame();
7. DefaultListModel<String> l1 = **new** DefaultListModel<>();
8. l1.addElement("Item1");
9. l1.addElement("Item2");
10. l1.addElement("Item3");
11. l1.addElement("Item4");
12. JList<String> list = **new** JList<>(l1);
13. list.setBounds(100,100, 75,75);
14. f.add(list);
15. f.setSize(400,400);
16. f.setLayout(**null**);
17. f.setVisible(**true**);
18. }
19. **public** **static** **void** main(String args[])
20. {
21. **new** ListExample();
22. }}

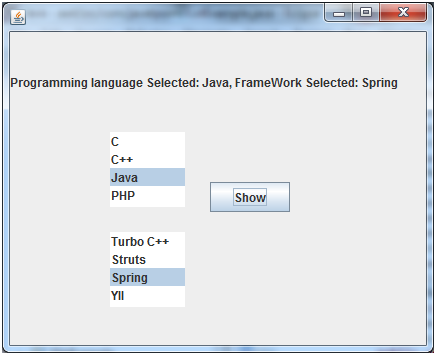
Output:



Java JList Example with ActionListener

1. **import** javax.swing.\*;
2. **import** java.awt.event.\*;
3. **public** **class** ListExample
4. {
5. ListExample(){
6. JFrame f= **new** JFrame();
7. **final** JLabel label = **new** JLabel();
8. label.setSize(500,100);
9. JButton b=**new** JButton("Show");
10. b.setBounds(200,150,80,30);
11. **final** DefaultListModel<String> l1 = **new** DefaultListModel<>();
12. l1.addElement("C");
13. l1.addElement("C++");
14. l1.addElement("Java");
15. l1.addElement("PHP");
16. **final** JList<String> list1 = **new** JList<>(l1);
17. list1.setBounds(100,100, 75,75);
18. DefaultListModel<String> l2 = **new** DefaultListModel<>();
19. l2.addElement("Turbo C++");
20. l2.addElement("Struts");
21. l2.addElement("Spring");
22. l2.addElement("YII");
23. **final** JList<String> list2 = **new** JList<>(l2);
24. list2.setBounds(100,200, 75,75);
25. f.add(list1); f.add(list2); f.add(b); f.add(label);
26. f.setSize(450,450);
27. f.setLayout(**null**);
28. f.setVisible(**true**);
29. b.addActionListener(**new** ActionListener() {
30. **public** **void** actionPerformed(ActionEvent e) {
31. String data = "";
32. **if** (list1.getSelectedIndex() != -1) {
33. data = "Programming language Selected: " + list1.getSelectedValue();
34. label.setText(data);
35. }
36. **if**(list2.getSelectedIndex() != -1){
37. data += ", FrameWork Selected: ";
38. **for**(Object frame :list2.getSelectedValues()){
39. data += frame + " ";
40. }
41. }
42. label.setText(data);
43. }
44. });
45. }
46. **public** **static** **void** main(String args[])
47. {
48. **new** ListExample();
49. }}

Output:



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.

JOptionPane class declaration

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

Common Constructors of JOptionPane class

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| JOptionPane() | It is used to create a JOptionPane with a test message. |
| JOptionPane(Object message) | It is used to create an instance of JOptionPane to display a message. |
| JOptionPane(Object message, int messageType | It is used to create an instance of JOptionPane to display a message with specified message type and default options. |

Common Methods of JOptionPane class

|  |  |
| --- | --- |
| **Methods** | **Description** |
| JDialog createDialog(String title) | It is used to create and return a new parentless JDialog with the specified title. |
| static void showMessageDialog(Component parentComponent, Object message) | It is used to create an information-message dialog titled "Message". |
| static void showMessageDialog(Component parentComponent, Object message, String title, int messageType) | It is used to create a message dialog with given title and messageType. |
| static int showConfirmDialog(Component parentComponent, Object message) | It is used to create a dialog with the options Yes, No and Cancel; with the title, Select an Option. |
| static String showInputDialog(Component parentComponent, Object message) | It is used to show a question-message dialog requesting input from the user parented to parentComponent. |
| void setInputValue(Object newValue) | It is used to set the input value that was selected or input by the user. |

Java JOptionPane Example: showMessageDialog()

1. **import** javax.swing.\*;
2. **public** **class** OptionPaneExample {
3. JFrame f;
4. OptionPaneExample(){
5. f=**new** JFrame();
6. JOptionPane.showMessageDialog(f,"Hello, Welcome to Javatpoint.");
7. }
8. **public** **static** **void** main(String[] args) {
9. **new** OptionPaneExample();
10. }
11. }

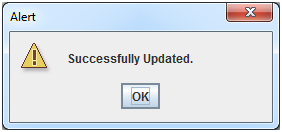
Output:



Java JOptionPane Example: showMessageDialog()

1. **import** javax.swing.\*;
2. **public** **class** OptionPaneExample {
3. JFrame f;
4. OptionPaneExample(){
5. f=**new** JFrame();
6. JOptionPane.showMessageDialog(f,"Successfully Updated.","Alert",JOptionPane.WARNING\_MESSAGE);
7. }
8. **public** **static** **void** main(String[] args) {
9. **new** OptionPaneExample();
10. }
11. }

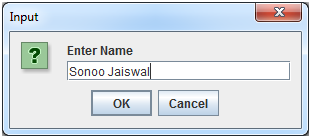
Output:



Java JOptionPane Example: showInputDialog()

1. **import** javax.swing.\*;
2. **public** **class** OptionPaneExample {
3. JFrame f;
4. OptionPaneExample(){
5. f=**new** JFrame();
6. String name=JOptionPane.showInputDialog(f,"Enter Name");
7. }
8. **public** **static** **void** main(String[] args) {
9. **new** OptionPaneExample();
10. }
11. }

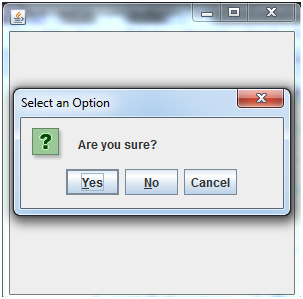
Output:



Java JOptionPane Example: showConfirmDialog()

1. **import** javax.swing.\*;
2. **import** java.awt.event.\*;
3. **public** **class** OptionPaneExample **extends** WindowAdapter{
4. JFrame f;
5. OptionPaneExample(){
6. f=**new** JFrame();
7. f.addWindowListener(**this**);
8. f.setSize(300, 300);
9. f.setLayout(**null**);
10. f.setDefaultCloseOperation(JFrame.DO\_NOTHING\_ON\_CLOSE);
11. f.setVisible(**true**);
12. }
13. **public** **void** windowClosing(WindowEvent e) {
14. **int** a=JOptionPane.showConfirmDialog(f,"Are you sure?");
15. **if**(a==JOptionPane.YES\_OPTION){
16. f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);
17. }
18. }
19. **public** **static** **void** main(String[] args) {
20. **new**  OptionPaneExample();
21. }
22. }

Output:



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

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

1. **public** **class** JScrollBar **extends** JComponent **implements** Adjustable, Accessible

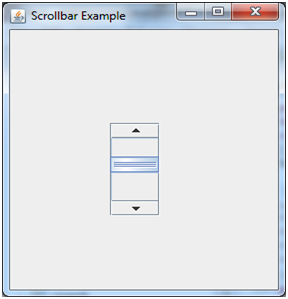
Commonly used Constructors:

|  |  |
| --- | --- |
| **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. |

Java JScrollBar Example

1. **import** javax.swing.\*;
2. **class** ScrollBarExample
3. {
4. ScrollBarExample(){
5. JFrame f= **new** JFrame("Scrollbar Example");
6. JScrollBar s=**new** JScrollBar();
7. s.setBounds(100,100, 50,100);
8. f.add(s);
9. f.setSize(400,400);
10. f.setLayout(**null**);
11. f.setVisible(**true**);
12. }
13. **public** **static** **void** main(String args[])
14. {
15. **new** ScrollBarExample();
16. }}

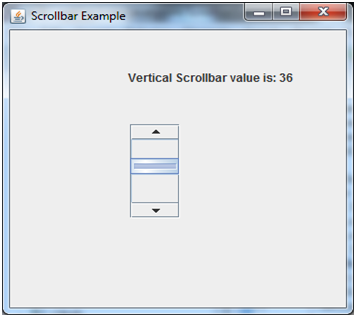
Output:



Java JScrollBar Example with AdjustmentListener

1. **import** javax.swing.\*;
2. **import** java.awt.event.\*;
3. **class** ScrollBarExample
4. {
5. ScrollBarExample(){
6. JFrame f= **new** JFrame("Scrollbar Example");
7. **final** JLabel label = **new** JLabel();
8. label.setHorizontalAlignment(JLabel.CENTER);
9. label.setSize(400,100);
10. **final** JScrollBar s=**new** JScrollBar();
11. s.setBounds(100,100, 50,100);
12. f.add(s); f.add(label);
13. f.setSize(400,400);
14. f.setLayout(**null**);
15. f.setVisible(**true**);
16. s.addAdjustmentListener(**new** AdjustmentListener() {
17. **public** **void** adjustmentValueChanged(AdjustmentEvent e) {
18. label.setText("Vertical Scrollbar value is:"+ s.getValue());
19. }
20. });
21. }
22. **public** **static** **void** main(String args[])
23. {
24. **new** ScrollBarExample();
25. }}

Output:



Java JFrame

The javax.swing.JFrame class is a type of container which inherits the java.awt.Frame class. JFrame works like the main window where components like labels, buttons, textfields are added to create a GUI.

Unlike Frame, JFrame has the option to hide or close the window with the help of setDefaultCloseOperation(int) method.

Nested Class

|  |  |  |
| --- | --- | --- |
| **Modifier and Type** | **Class** | **Description** |
| protected class | JFrame.AccessibleJFrame | This class implements accessibility support for the JFrame class. |

Fields

|  |  |  |
| --- | --- | --- |
| **Modifier and Type** | **Field** | **Description** |
| protected AccessibleContext | accessibleContext | The accessible context property. |
| static int | EXIT\_ON\_CLOSE | The exit application default window close operation. |
| protected JRootPane | rootPane | The JRootPane instance that manages the contentPane and optional menuBar for this frame, as well as the glassPane. |
| protected boolean | rootPaneCheckingEnabled | If true then calls to add and setLayout will be forwarded to the contentPane. |

Constructors

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| JFrame() | It constructs a new frame that is initially invisible. |
| JFrame(GraphicsConfiguration gc) | It creates a Frame in the specified GraphicsConfiguration of a screen device and a blank title. |
| JFrame(String title) | It creates a new, initially invisible Frame with the specified title. |
| JFrame(String title, GraphicsConfiguration gc) | It creates a JFrame with the specified title and the specified GraphicsConfiguration of a screen device. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Modifier and Type** | **Method** | | **Description** |
|  |  |  | |
| protected void | addImpl(Component comp, Object constraints, int index) | Adds the specified child Component. | |
| protected JRootPane | createRootPane() | Called by the constructor methods to create the default rootPane. | |
| protected void | frameInit() | Called by the constructors to init the JFrame properly. | |
| void | setContentPane(Containe contentPane) | It sets the contentPane property | |
| static void | setDefaultLookAndFeelDecorated(boolean defaultLookAndFeelDecorated) | Provides a hint as to whether or not newly created JFrames should have their Window decorations (such as borders, widgets to close the window, title...) provided by the current look and feel. | |
| void | setIconImage(Image image) | It sets the image to be displayed as the icon for this window. | |
| void | setJMenuBar(JMenuBar menubar) | It sets the menubar for this frame. | |
| void | setLayeredPane(JLayeredPane layeredPane) | It sets the layeredPane property. | |
| JRootPane | getRootPane() | It returns the rootPane object for this frame. | |
| TransferHandler | getTransferHandler() | It gets the transferHandler property. | |

Useful Methods

JFrame Example

1. **import** java.awt.FlowLayout;
2. **import** javax.swing.JButton;
3. **import** javax.swing.JFrame;
4. **import** javax.swing.JLabel;
5. **import** javax.swing.JPanel;
6. **public** **class** JFrameExample {
7. **public** **static** **void** main(String s[]) {
8. JFrame frame = **new** JFrame("JFrame Example");
9. JPanel panel = **new** JPanel();
10. panel.setLayout(**new** FlowLayout());
11. JLabel label = **new** JLabel("JFrame By Example");
12. JButton button = **new** JButton();
13. button.setText("Button");
14. panel.add(label);
15. panel.add(button);
16. frame.add(panel);
17. frame.setSize(200, 300);
18. frame.setLocationRelativeTo(**null**);
19. frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);
20. frame.setVisible(**true**);
21. }
22. }

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

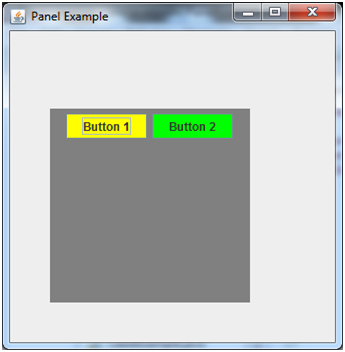
Commonly used Constructors:

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



# Java JScrollPane

A JscrollPane is used to make scrollable view of a component. When screen size is limited, we use a scroll pane to display a large component or a component whose size can change dynamically.

Constructors

|  |  |
| --- | --- |
| **Constructor** | **Purpose** |
| JScrollPane() | It creates a scroll pane. The Component parameter, when present, sets the scroll pane's client. The two int parameters, when present, set the vertical and horizontal scroll bar policies (respectively). |
| JScrollPane(Component) |
| JScrollPane(int, int) |
| JScrollPane(Component, int, int) |

Useful Methods

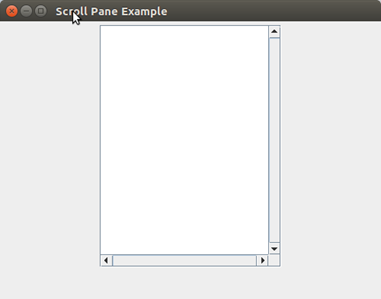
|  |  |  |
| --- | --- | --- |
| **Modifier** | **Method** | **Description** |
| void | setColumnHeaderView(Component) | It sets the column header for the scroll pane. |
| void | setRowHeaderView(Component) | It sets the row header for the scroll pane. |
| void | setCorner(String, Component) | It sets or gets the specified corner. The int parameter specifies which corner and must be one of the following constants defined in ScrollPaneConstants: UPPER\_LEFT\_CORNER, UPPER\_RIGHT\_CORNER, LOWER\_LEFT\_CORNER, LOWER\_RIGHT\_CORNER, LOWER\_LEADING\_CORNER, LOWER\_TRAILING\_CORNER, UPPER\_LEADING\_CORNER, UPPER\_TRAILING\_CORNER. |
| Component | getCorner(String) |
| void | setViewportView(Component) | Set the scroll pane's client. |

JScrollPane Example

1. **import** java.awt.FlowLayout;
2. **import** javax.swing.JFrame;
3. **import** javax.swing.JScrollPane;
4. **import** javax.swing.JtextArea;
6. **public** **class** JScrollPaneExample {
7. **private** **static** **final** **long** serialVersionUID = 1L;
9. **private** **static** **void** createAndShowGUI() {
11. // Create and set up the window.
12. **final** JFrame frame = **new** JFrame("Scroll Pane Example");
14. // Display the window.
15. frame.setSize(500, 500);
16. frame.setVisible(**true**);
17. frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);
19. // set flow layout for the frame
20. frame.getContentPane().setLayout(**new** FlowLayout());
22. JTextArea textArea = **new** JTextArea(20, 20);
23. JScrollPane scrollableTextArea = **new** JScrollPane(textArea);
25. scrollableTextArea.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL\_SCROLLBAR\_ALWAYS);
26. scrollableTextArea.setVerticalScrollBarPolicy(JScrollPane.VERTICAL\_SCROLLBAR\_ALWAYS);
28. frame.getContentPane().add(scrollableTextArea);
29. }
30. **public** **static** **void** main(String[] args) {

33. javax.swing.SwingUtilities.invokeLater(**new** Runnable() {
35. **public** **void** run() {
36. createAndShowGUI();
37. }
38. });
39. }
40. }

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.

JMenuBar class declaration

1. **public** **class** JMenuBar **extends** JComponent **implements** MenuElement, Accessible

JMenu class declaration

1. **public** **class** JMenu **extends** JMenuItem **implements** MenuElement, Accessible

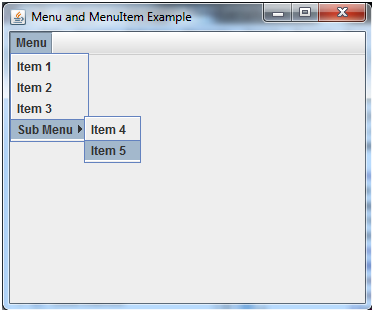
JMenuItem class declaration

1. **public** **class** JMenuItem **extends** AbstractButton **implements** Accessible, MenuElement

Java JMenuItem and JMenu Example

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. }}

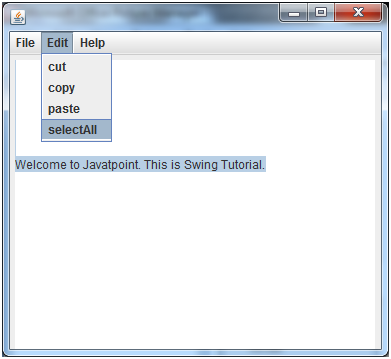
Output:



Example of creating Edit menu for Notepad:

1. **import** javax.swing.\*;
2. **import** java.awt.event.\*;
3. **public** **class** MenuExample **implements** ActionListener{
4. JFrame f;
5. JMenuBar mb;
6. JMenu file,edit,help;
7. JMenuItem cut,copy,paste,selectAll;
8. JTextArea ta;
9. MenuExample(){
10. f=**new** JFrame();
11. cut=**new** JMenuItem("cut");
12. copy=**new** JMenuItem("copy");
13. paste=**new** JMenuItem("paste");
14. selectAll=**new** JMenuItem("selectAll");
15. cut.addActionListener(**this**);
16. copy.addActionListener(**this**);
17. paste.addActionListener(**this**);
18. selectAll.addActionListener(**this**);
19. mb=**new** JMenuBar();
20. file=**new** JMenu("File");
21. edit=**new** JMenu("Edit");
22. help=**new** JMenu("Help");
23. edit.add(cut);edit.add(copy);edit.add(paste);edit.add(selectAll);
24. mb.add(file);mb.add(edit);mb.add(help);
25. ta=**new** JTextArea();
26. ta.setBounds(5,5,360,320);
27. f.add(mb);f.add(ta);
28. f.setJMenuBar(mb);
29. f.setLayout(**null**);
30. f.setSize(400,400);
31. f.setVisible(**true**);
32. }
33. **public** **void** actionPerformed(ActionEvent e) {
34. **if**(e.getSource()==cut)
35. ta.cut();
36. **if**(e.getSource()==paste)
37. ta.paste();
38. **if**(e.getSource()==copy)
39. ta.copy();
40. **if**(e.getSource()==selectAll)
41. ta.selectAll();
42. }
43. **public** **static** **void** main(String[] args) {
44. **new** MenuExample();
45. }
46. }

Output:



Java JToolBar

JToolBar container allows us to group other components, usually buttons with icons in a row or column. JToolBar provides a component which is useful for displaying commonly used actions or controls.

Nested Classes

|  |  |  |
| --- | --- | --- |
| **Modifier and Type** | **Class** | **Description** |
| protected class | JToolBar.AccessibleJToolBar | This class implements accessibility support for the JToolBar class. |
| static class | JToolBar.Separator | A toolbar-specific separator. |

Constructors

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| JToolBar() | It creates a new tool bar; orientation defaults to HORIZONTAL. |
| JToolBar(int orientation) | It creates a new tool bar with the specified orientation. |
| JToolBar(String name) | It creates a new tool bar with the specified name. |
| JToolBar(String name, int orientation) | It creates a new tool bar with a specified name and orientation. |

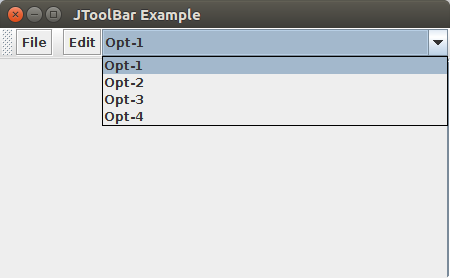
Useful Methods

|  |  |  |
| --- | --- | --- |
| **Modifier and Type** | **Method** | **Description** |
| JButton | add(Action a) | It adds a new JButton which dispatches the action. |
| protected void | addImpl(Component comp, Object constraints, int index) | If a JButton is being added, it is initially set to be disabled. |
| void | addSeparator() | It appends a separator of default size to the end of the tool bar. |
| protected PropertyChangeListener | createActionChangeListener(JButton b) | It returns a properly configured PropertyChangeListener which updates the control as changes to the Action occur, or null if the default property change listener for the control is desired. |
| protected JButton | createActionComponent(Action a) | Factory method which creates the JButton for Actions added to the JToolBar. |
| ToolBarUI | getUI() | It returns the tool bar's current UI. |
| void | setUI(ToolBarUI ui) | It sets the L&F object that renders this component. |
| void | setOrientation(int o) | It sets the orientation of the tool bar. |

Java JToolBar Example

1. **import** java.awt.BorderLayout;
2. **import** java.awt.Container;
3. **import** javax.swing.JButton;
4. **import** javax.swing.JComboBox;
5. **import** javax.swing.JFrame;
6. **import** javax.swing.JScrollPane;
7. **import** javax.swing.JTextArea;
8. **import** javax.swing.JToolBar;
10. **public** **class** JToolBarExample {
11. **public** **static** **void** main(**final** String args[]) {
12. JFrame myframe = **new** JFrame("JToolBar Example");
13. myframe.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);
14. JToolBar toolbar = **new** JToolBar();
15. toolbar.setRollover(**true**);
16. JButton button = **new** JButton("File");
17. toolbar.add(button);
18. toolbar.addSeparator();
19. toolbar.add(**new** JButton("Edit"));
20. toolbar.add(**new** JComboBox(**new** String[] { "Opt-1", "Opt-2", "Opt-3", "Opt-4" }));
21. Container contentPane = myframe.getContentPane();
22. contentPane.add(toolbar, BorderLayout.NORTH);
23. JTextArea textArea = **new** JTextArea();
24. JScrollPane mypane = **new** JScrollPane(textArea);
25. contentPane.add(mypane, BorderLayout.EAST);
26. myframe.setSize(450, 250);
27. myframe.setVisible(**true**);
28. }
29. }

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.

JTabbedPane class declaration

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

1. **public** **class** JTabbedPane **extends** JComponent **implements** Serializable, Accessible, SwingConstants

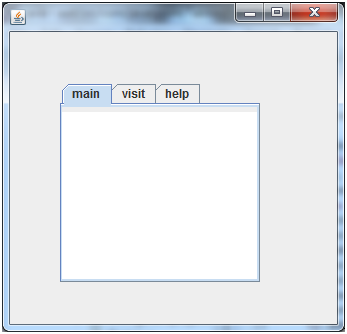
Commonly used Constructors:

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| JTabbedPane() | Creates an empty TabbedPane with a default tab placement of JTabbedPane.Top. |
| JTabbedPane(int tabPlacement) | Creates an empty TabbedPane with a specified tab placement. |
| JTabbedPane(int tabPlacement, int tabLayoutPolicy) | Creates an empty TabbedPane with a specified tab placement and tab layout policy. |

Java JTabbedPane Example

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 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.

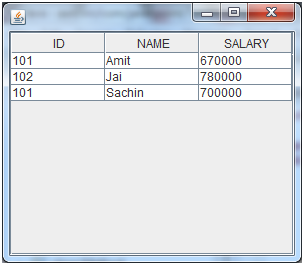
Commonly used Constructors:

|  |  |
| --- | --- |
| 1. **Constructor** | 1. **Description** |
| 1. JTable() | 1. Creates a table with empty cells. |
| 1. JTable(Object[][] rows, Object[] columns) | 1. 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. }

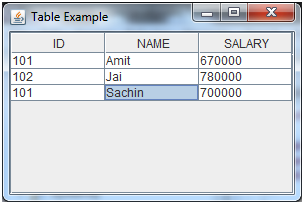
Output:



Java 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. System.out.println("Table element selected is: " + Data);
24. }
25. });
26. JScrollPane sp=**new** JScrollPane(jt);
27. f.add(sp);
28. f.setSize(300, 200);
29. f.setVisible(**true**);
30. }
31. }

Output:



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 JTree

The JTree class is used to display the tree structured data or hierarchical data. JTree is a complex component. It has a 'root node' at the top most which is a parent for all nodes in the tree. It inherits JComponent class.

JTree class declaration

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

1. **public** **class** JTree **extends** JComponent **implements** Scrollable, Accessible

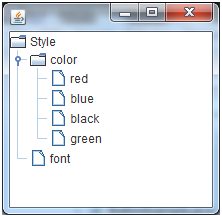
Commonly used Constructors:

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| JTree() | Creates a JTree with a sample model. |
| JTree(Object[] value) | Creates a JTree with every element of the specified array as the child of a new root node. |
| JTree(TreeNode root) | Creates a JTree with the specified TreeNode as its root, which displays the root node. |

Java JTree Example

1. **import** javax.swing.\*;
2. **import** javax.swing.tree.DefaultMutableTreeNode;
3. **public** **class** TreeExample {
4. JFrame f;
5. TreeExample(){
6. f=**new** JFrame();
7. DefaultMutableTreeNode style=**new** DefaultMutableTreeNode("Style");
8. DefaultMutableTreeNode color=**new** DefaultMutableTreeNode("color");
9. DefaultMutableTreeNode font=**new** DefaultMutableTreeNode("font");
10. style.add(color);
11. style.add(font);
12. DefaultMutableTreeNode red=**new** DefaultMutableTreeNode("red");
13. DefaultMutableTreeNode blue=**new** DefaultMutableTreeNode("blue");
14. DefaultMutableTreeNode black=**new** DefaultMutableTreeNode("black");
15. DefaultMutableTreeNode green=**new** DefaultMutableTreeNode("green");
16. color.add(red); color.add(blue); color.add(black); color.add(green);
17. JTree jt=**new** JTree(style);
18. f.add(jt);
19. f.setSize(200,200);
20. f.setVisible(**true**);
21. }
22. **public** **static** **void** main(String[] args) {
23. **new** TreeExample();
24. }}

Output:



|  |  |
| --- | --- |
| **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)

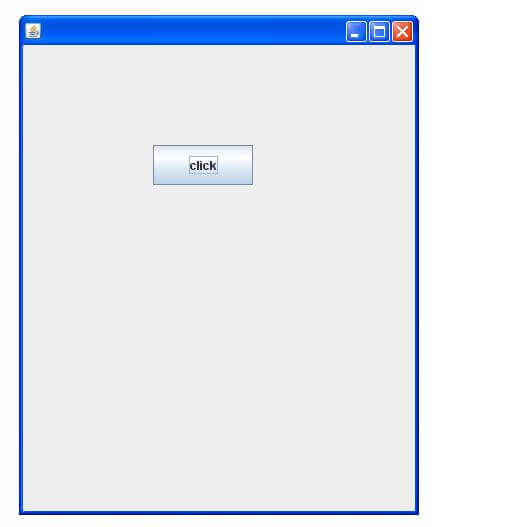
We can write the code of swing inside the main(), constructor or any other method.

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
6. JButton b=**new** JButton("click");//creating instance of JButton
7. b.setBounds(130,100,100, 40);//x axis, y axis, width, height
9. f.add(b);//adding button in JFrame
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
7. JButton b=**new** JButton("click");//creating instance of JButton
8. b.setBounds(130,100,100, 40);
10. f.add(b);//adding button in JFrame
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. }
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);
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. }}

[download this example](https://static.javatpoint.com/src/swing/first2.zip)

*What we will learn in Swing Tutorial*

* JButton class
* JRadioButton class
* JTextArea class
* JComboBox class
* JTable class
* JColorChooser class
* JProgressBar class
* JSlider class
* Digital Watch
* Graphics in swing
* Displaying image
* Edit menu code for Notepad
* OpenDialog Box
* Notepad
* Puzzle Game
* Pic Puzzle Game
* Tic Tac Toe Game
* BorderLayout
* GridLayout
* FlowLayout
* CardLayout