Java AWT Tutorial

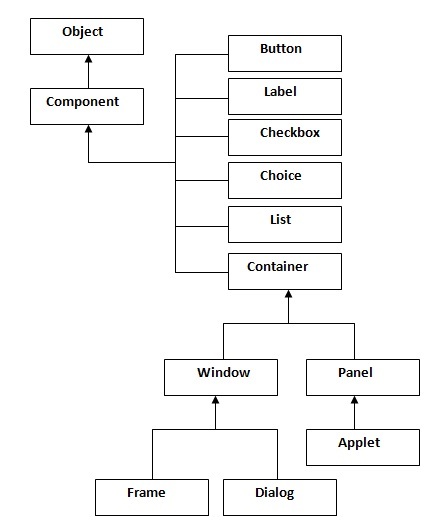
**Java AWT** (Abstract Window Toolkit) is *an API to develop GUI or window-based applications* in java.

Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavyweight i.e. its components are using the resources of OS.

The java.awt package provides classes for AWT api such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.

Java AWT Hierarchy

The hierarchy of Java AWT classes are given below.



Container

The Container is a component in AWT that can contain another components like buttons, textfields, labels etc. The classes that extends Container class are known as container such as Frame, Dialog and Panel.

Window

The window is the container that have no borders and menu bars. You must use frame, dialog or another window for creating a window.

Panel

The Panel is the container that doesn't contain title bar and menu bars. It can have other components like button, textfield etc.

Frame

The Frame is the container that contain title bar and can have menu bars. It can have other components like button, textfield etc.

Useful Methods of Component class

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void add(Component c) | inserts a component on this component. |
| public void setSize(int width,int height) | sets the size (width and height) of the component. |
| public void setLayout(LayoutManager m) | defines the layout manager for the component. |
| public void setVisible(boolean status) | changes the visibility of the component, by default false. |

Java AWT Example

To create simple awt example, you need a frame. There are two ways to create a frame in AWT.

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

AWT Example by Inheritance

Let's see a simple example of AWT where we are inheriting Frame class. Here, we are showing Button component on the Frame.

**import** java.awt.\*;

**class** First **extends** Frame{

First(){

Button b=**new** Button("click me");

b.setBounds(30,100,80,30);// setting button position

add(b);//adding button into frame

setSize(300,300);//frame size 300 width and 300 height

setLayout(**null**);//no layout manager

setVisible(**true**);//now frame will be visible, by default not visible

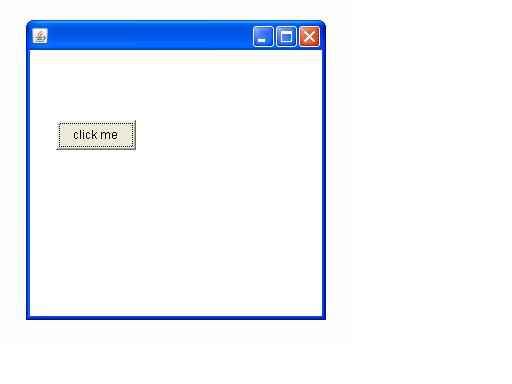
}

**public** **static** **void** main(String args[]){

First f=**new** First();

}}

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



AWT Example by Association

Let's see a simple example of AWT where we are creating instance of Frame class. Here, we are showing Button component on the Frame.

**import** java.awt.\*;

**class** First2{

First2(){

Frame f=**new** Frame();

Button b=**new** Button("click me");

b.setBounds(30,50,80,30);

f.add(b);

f.setSize(300,300);

f.setLayout(**null**);

f.setVisible(**true**);

}

**public** **static** **void** main(String args[]){

First2 f=**new** First2();

}}

# awt example

# Event and Listener (Java Event Handling)

|  |
| --- |
| Changing the state of an object is known as an event. For example, click on button, dragging mouse etc. The java.awt.event package provides many event classes and Listener interfaces for event handling. |

Java Event classes and Listener interfaces

|  |  |
| --- | --- |
| **Event Classes** | **Listener Interfaces** |
| ActionEvent | ActionListener |
| MouseEvent | MouseListener and MouseMotionListener |
| MouseWheelEvent | MouseWheelListener |
| KeyEvent | KeyListener |
| ItemEvent | ItemListener |
| TextEvent | TextListener |
| AdjustmentEvent | AdjustmentListener |
| WindowEvent | WindowListener |
| ComponentEvent | ComponentListener |
| ContainerEvent | ContainerListener |
| FocusEvent | FocusListener |

Java AWT Button

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

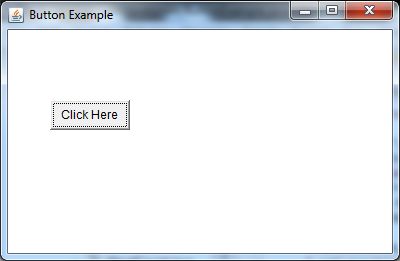
AWT Button Class declaration

1. **public** **class** Button **extends** Component **implements** Accessible

Java AWT Button Example

1. **import** java.awt.\*;
2. **public** **class** ButtonExample {
3. **public** **static** **void** main(String[] args) {
4. Frame f=**new** Frame("Button Example");
5. Button b=**new** Button("Click Here");
6. b.setBounds(50,100,80,30);
7. f.add(b);
8. f.setSize(400,400);
9. f.setLayout(**null**);
10. f.setVisible(**true**);
11. }
12. }

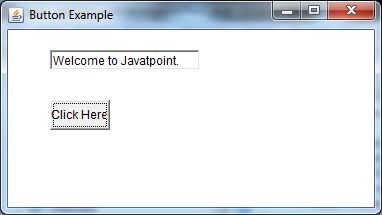
Output:



Java AWT Button Example with ActionListener

1. **import** java.awt.\*;
2. **import** java.awt.event.\*;
3. **public** **class** ButtonExample {
4. **public** **static** **void** main(String[] args) {
5. Frame f=**new** Frame("Button Example");
6. **final** TextField tf=**new** TextField();
7. tf.setBounds(50,50, 150,20);
8. Button b=**new** Button("Click Here");
9. b.setBounds(50,100,60,30);
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:



Java AWT Label

The object of Label 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.

AWT Label Class Declaration

1. **public** **class** Label **extends** Component **implements** Accessible

Java Label Example

1. **import** java.awt.\*;
2. **class** LabelExample{
3. **public** **static** **void** main(String args[]){
4. Frame f= **new** Frame("Label Example");
5. Label l1,l2;
6. l1=**new** Label("First Label.");
7. l1.setBounds(50,100, 100,30);
8. l2=**new** Label("Second Label.");
9. l2.setBounds(50,150, 100,30);
10. f.add(l1); f.add(l2);
11. f.setSize(400,400);
12. f.setLayout(**null**);
13. f.setVisible(**true**);
14. }
15. }

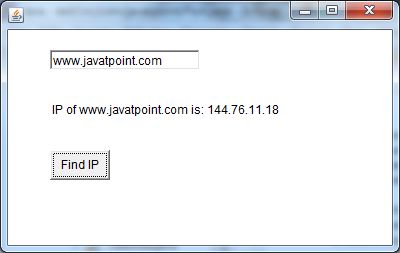
Output:



Java AWT Label Example with ActionListener

1. **import** java.awt.\*;
2. **import** java.awt.event.\*;
3. **public** **class** LabelExample **extends** Frame **implements** ActionListener{
4. TextField tf; Label l; Button b;
5. LabelExample(){
6. tf=**new** TextField();
7. tf.setBounds(50,50, 150,20);
8. l=**new** Label();
9. l.setBounds(50,100, 250,20);
10. b=**new** Button("Find IP");
11. b.setBounds(50,150,60,30);
12. b.addActionListener(**this**);
13. add(b);add(tf);add(l);
14. setSize(400,400);
15. setLayout(**null**);
16. setVisible(**true**);
17. ;    }
18. **public** **void** actionPerformed(ActionEvent e) {
19. **try**{
20. String host=tf.getText();
21. String ip=java.net.InetAddress.getByName(host).getHostAddress();
22. l.setText("IP of "+host+" is: "+ip);
23. }**catch**(Exception ex){System.out.println(ex);}
24. }
25. **public** **static** **void** main(String[] args) {
26. **new** LabelExample();
27. }
28. }

Output:



Java AWT TextField

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

AWT TextField Class Declaration

1. **public** **class** TextField **extends** TextComponent

Java AWT TextField Example

**import** java.awt.\*;

**class** TextFieldExample{

**public** **static** **void** main(String args[]){

    Frame f= **new** Frame("TextField Example");

    TextField t1,t2;

    t1=**new** TextField("Welcome to Javatpoint.");

    t1.setBounds(50,100, 200,30);

    t2=**new** TextField("AWT Tutorial");

    t2.setBounds(50,150, 200,30);

    f.add(t1); f.add(t2);

    f.setSize(400,400);

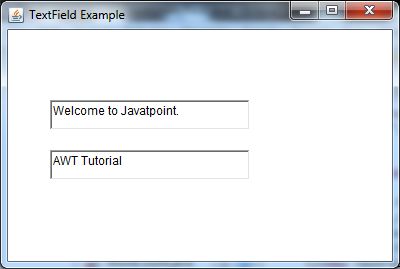
    f.setLayout(**null**);

    f.setVisible(**true**);

}

1. }

Output:



Java AWT TextField Example with ActionListener

**import** java.awt.\*;

**import** java.awt.event.\*;

**public** **class** TextFieldExample **extends** Frame **implements** ActionListener{

    TextField tf1,tf2,tf3;

    Button b1,b2;

    TextFieldExample(){

        tf1=**new** TextField();

        tf1.setBounds(50,50,150,20);

        tf2=**new** TextField();

        tf2.setBounds(50,100,150,20);

        tf3=**new** TextField();

        tf3.setBounds(50,150,150,20);

        tf3.setEditable(**false**);

        b1=**new** Button("+");

        b1.setBounds(50,200,50,50);

        b2=**new** Button("-");

        b2.setBounds(120,200,50,50);

        b1.addActionListener(**this**);

        b2.addActionListener(**this**);

        add(tf1);add(tf2);add(tf3);add(b1);add(b2);

        setSize(300,300);

        setLayout(**null**);

        setVisible(**true**);

    }

**public** **void** actionPerformed(ActionEvent e) {

        String s1=tf1.getText();

        String s2=tf2.getText();

**int** a=Integer.parseInt(s1);

**int** b=Integer.parseInt(s2);

**int** c=0;

**if**(e.getSource()==b1){

            c=a+b;

        }**else** **if**(e.getSource()==b2){

            c=a-b;

        }

        String result=String.valueOf(c);

        tf3.setText(result);

    }

**public** **static** **void** main(String[] args) {

**new** TextFieldExample();

}

}

Java AWT PopupMenu

PopupMenu can be dynamically popped up at specific position within a component. It inherits the Menu class.

AWT PopupMenu class declaration

**public** **class** PopupMenu **extends** Menu **implements** MenuContainer,

Accessible

Java AWT PopupMenu Example

**import** java.awt.\*;

**import** java.awt.event.\*;

**class** PopupMenuExample

{

     PopupMenuExample(){

**final** Frame f= **new** Frame("PopupMenu Example");

**final** PopupMenu popupmenu = **new** PopupMenu("Edit");

         MenuItem cut = **new** MenuItem("Cut");

         cut.setActionCommand("Cut");

         MenuItem copy = **new** MenuItem("Copy");

         copy.setActionCommand("Copy");

         MenuItem paste = **new** MenuItem("Paste");

         paste.setActionCommand("Paste");

         popupmenu.add(cut);

         popupmenu.add(copy);

         popupmenu.add(paste);

         f.addMouseListener(**new** MouseAdapter() {

**public** **void** mouseClicked(MouseEvent e) {

                popupmenu.show(f , e.getX(), e.getY());

            }

         });

         f.add(popupmenu);

         f.setSize(400,400);

         f.setLayout(**null**);

         f.setVisible(**true**);

     }

**public** **static** **void** main(String args[])

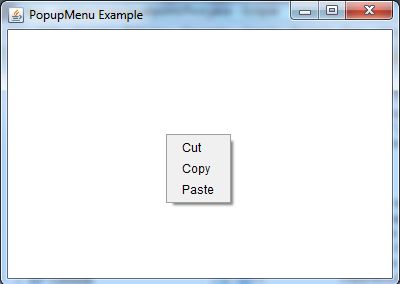
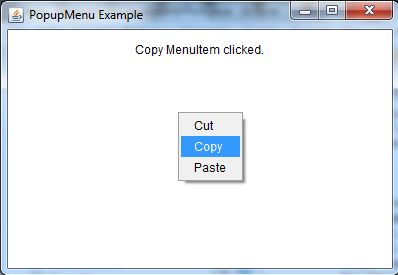
{

**new** PopupMenuExample();

}

}

Output:

Java ActionListener Interface

The Java ActionListener is notified whenever you click on the button or menu item. It is notified against ActionEvent. The ActionListener interface is found in java.awt.event package. It has only one method: actionPerformed().

actionPerformed() method

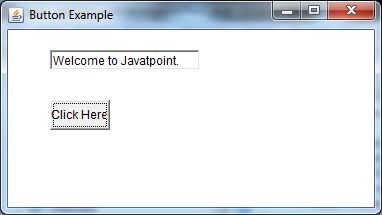
The actionPerformed() method is invoked automatically whenever you click on the registered component.

1. **public** **abstract** **void** actionPerformed(ActionEvent e);

Java ActionListener Example: On Button click

1. **import** java.awt.\*;
2. **import** java.awt.event.\*;
3. **public** **class** ActionListenerExample {
4. **public** **static** **void** main(String[] args) {
5. Frame f=**new** Frame("ActionListener Example");
6. **final** TextField tf=**new** TextField();
7. tf.setBounds(50,50, 150,20);
8. Button b=**new** Button("Click Here");
9. b.setBounds(50,100,60,30);
11. b.addActionListener(**new** ActionListener(){
12. **public** **void** actionPerformed(ActionEvent e){
13. tf.setText("Welcome to Javatpoint.");
14. }
15. });
16. f.add(b);f.add(tf);
17. f.setSize(400,400);
18. f.setLayout(**null**);
19. f.setVisible(**true**);
20. }
21. }

Output:



ava MouseListener Interface

The Java MouseListener is notified whenever you change the state of mouse. It is notified against MouseEvent. The MouseListener interface is found in java.awt.event package. It has five methods.

Methods of MouseListener interface

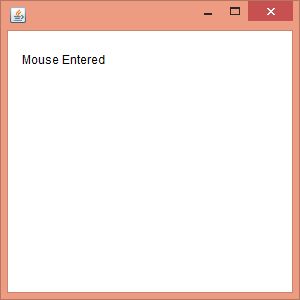
The signature of 5 methods found in MouseListener interface are given below:

1. **public** **abstract** **void** mouseClicked(MouseEvent e);
2. **public** **abstract** **void** mouseEntered(MouseEvent e);
3. **public** **abstract** **void** mouseExited(MouseEvent e);
4. **public** **abstract** **void** mousePressed(MouseEvent e);
5. **public** **abstract** **void** mouseReleased(MouseEvent e);

Java MouseListener Example

1. **import** java.awt.\*;
2. **import** java.awt.event.\*;
3. **public** **class** MouseListenerExample **extends** Frame **implements** MouseListener{
4. Label l;
5. MouseListenerExample(){
6. addMouseListener(**this**);
8. l=**new** Label();
9. l.setBounds(20,50,100,20);
10. add(l);
11. setSize(300,300);
12. setLayout(**null**);
13. setVisible(**true**);
14. }
15. **public** **void** mouseClicked(MouseEvent e) {
16. l.setText("Mouse Clicked");
17. }
18. **public** **void** mouseEntered(MouseEvent e) {
19. l.setText("Mouse Entered");
20. }
21. **public** **void** mouseExited(MouseEvent e) {
22. l.setText("Mouse Exited");
23. }
24. **public** **void** mousePressed(MouseEvent e) {
25. l.setText("Mouse Pressed");
26. }
27. **public** **void** mouseReleased(MouseEvent e) {
28. l.setText("Mouse Released");
29. }
30. **public** **static** **void** main(String[] args) {
31. **new** MouseListenerExample();
32. }
33. }

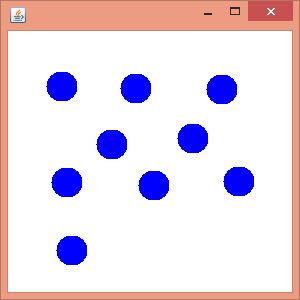
Output:



Java MouseListener Example 2

1. **import** java.awt.\*;
2. **import** java.awt.event.\*;
3. **public** **class** MouseListenerExample2 **extends** Frame **implements** MouseListener{
4. MouseListenerExample2(){
5. addMouseListener(**this**);
7. setSize(300,300);
8. setLayout(**null**);
9. setVisible(**true**);
10. }
11. **public** **void** mouseClicked(MouseEvent e) {
12. Graphics g=getGraphics();
13. g.setColor(Color.BLUE);
14. g.fillOval(e.getX(),e.getY(),30,30);
15. }
16. **public** **void** mouseEntered(MouseEvent e) {}
17. **public** **void** mouseExited(MouseEvent e) {}
18. **public** **void** mousePressed(MouseEvent e) {}
19. **public** **void** mouseReleased(MouseEvent e) {}
21. **public** **static** **void** main(String[] args) {
22. **new** MouseListenerExample2();
23. }
24. }

Output:



ava MouseMotionListener Interface

The Java MouseMotionListener is notified whenever you move or drag mouse. It is notified against MouseEvent. The MouseMotionListener interface is found in java.awt.event package. It has two methods.

Methods of MouseMotionListener interface

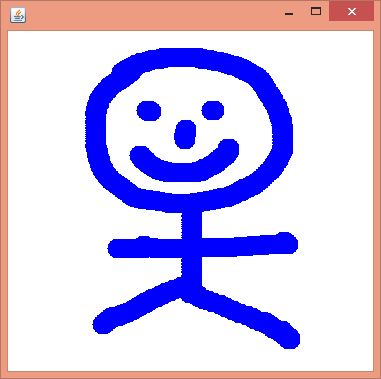
The signature of 2 methods found in MouseMotionListener interface are given below:

1. **public** **abstract** **void** mouseDragged(MouseEvent e);
2. **public** **abstract** **void** mouseMoved(MouseEvent e);

Java MouseMotionListener Example

1. **import** java.awt.\*;
2. **import** java.awt.event.\*;
3. **public** **class** MouseMotionListenerExample **extends** Frame **implements** MouseMotionListener{
4. MouseMotionListenerExample(){
5. addMouseMotionListener(**this**);
7. setSize(300,300);
8. setLayout(**null**);
9. setVisible(**true**);
10. }
11. **public** **void** mouseDragged(MouseEvent e) {
12. Graphics g=getGraphics();
13. g.setColor(Color.BLUE);
14. g.fillOval(e.getX(),e.getY(),20,20);
15. }
16. **public** **void** mouseMoved(MouseEvent e) {}
18. **public** **static** **void** main(String[] args) {
19. **new** MouseMotionListenerExample();
20. }
21. }

Output:



Java MouseMotionListener Example 2

**import** java.awt.\*;

**import** java.awt.event.MouseEvent;

**import** java.awt.event.MouseMotionListener;

**public** **class** Paint **extends** Frame **implements** MouseMotionListener{

    Label l;

    Color c=Color.BLUE;

    Paint(){

    l=**new** Label();

    l.setBounds(20,40,100,20);

    add(l);

  addMouseMotionListener(**this**);

   setSize(400,400);

    setLayout(**null**);

    setVisible(**true**);

}

**public** **void** mouseDragged(MouseEvent e) {

    l.setText("X="+e.getX()+", Y="+e.getY());

    Graphics g=getGraphics();

    g.setColor(Color.RED);

    g.fillOval(e.getX(),e.getY(),20,20);

}

**public** **void** mouseMoved(MouseEvent e) {

    l.setText("X="+e.getX()+", Y="+e.getY());

}

**public** **static** **void** main(String[] args) {

**new** Paint();

}

}

Output:



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.

|  |  |  |
| --- | --- | --- |
| **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

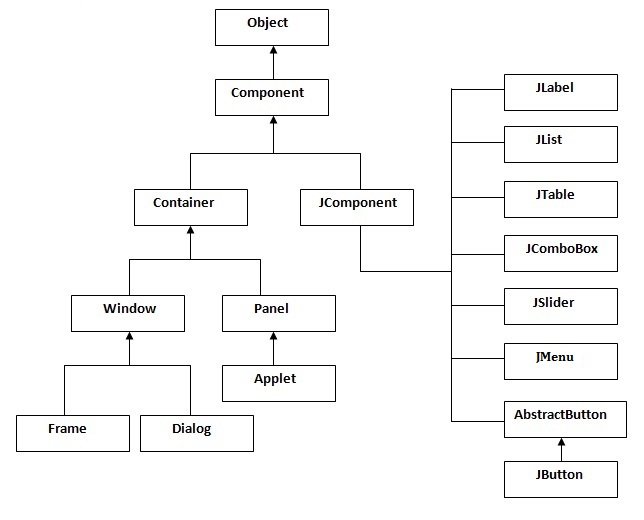
The Java Foundation Classes (JFC) are a set of GUI components which simplify the development of desktop applications.

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.

|  |  |
| --- | --- |
| **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*

**import** javax.swing.\*;

**public** **class** FirstSwingExample  Extends JFrame{

**public** **static** **void** main(String[] args) {

//JFrame f=**new** JFrame();//creating instance of JFrame

JButton b=**new** JButton("click");//creating instance of JButton

b.setBounds(130,100,100, 40);//x axis, y axis, width, height

add(b);//adding button in JFrame

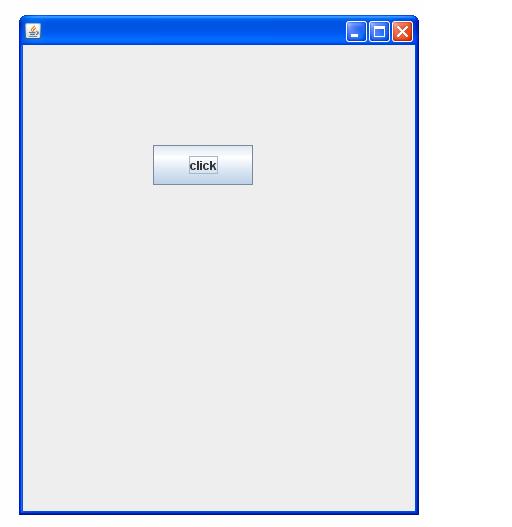
setSize(400,500);//400 width and 500 height

setLayout(**null**);//using no layout managers

setVisible(**true**);//making the frame visible

}

}



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://www.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

Word Character Counter in Java with Source Code

**Word Character Counter in Java with Source Code:** We can develop Word Character Counter in java with the help of string, AWT/Swing with event handling. Let's see the code of creating Word Character Counter in java.

String text="hello javatpoint this is wcc tool";

String words[]=text.split("\\s");

**int** length=words.length;//returns total number of words

**int** clength=text.length();//returns total number of characters with space

Let's see the swing code to count word and character.

**import** java.awt.event.\*;

**import** javax.swing.\*;

**public** **class** WCC **extends** JFrame **implements** ActionListener

{

JTextArea ta;

JButton b1,b2;

WCC(){

**super**("Word Character Counter - JavaTpoint");

    ta=**new** JTextArea();

    ta.setBounds(50,50,300,200);

    b1=**new** JButton("Word");

    b1.setBounds(50,300,100,30);

    b2=**new** JButton("Character");

    b2.setBounds(180,300,100,30);

    b1.addActionListener(**this**);

    b2.addActionListener(**this**);

    add(b1);add(b2);add(ta);

    setSize(400,400);

    setLayout(**null**);

    setVisible(**true**);

}

**public** **void** actionPerformed(ActionEvent e){

    String text=ta.getText();

**if**(e.getSource()==b1){

        String words[]=text.split("\\s");

        JOptionPane.showMessageDialog(**this**,"Total words: "+words.length);

    }

**if**(e.getSource()==b2){

        JOptionPane.showMessageDialog(**this**,"Total Characters with space: "+text.length());

    }

}

**public** **static** **void** main(String[] args) {

**new** WCC();

}

}

Java Swing Apps

[Notepad](https://www.javatpoint.com/notepad)

[Calculator](https://www.javatpoint.com/calculator-in-java)

[IP Finder](https://www.javatpoint.com/ip-finder-in-java)

[**Word Counter** [open link](https://www.javatpoint.com/word-count-in-java)](https://www.javatpoint.com/word-count-in-java)

[URL Source Generator](https://www.javatpoint.com/url-source-code-generator-in-java)

[Folder Explorer](https://www.javatpoint.com/folder-explorer-in-java)

[Puzzle Game](https://www.javatpoint.com/Puzzle-Game)

[Pic Puzzle Game](https://www.javatpoint.com/Pic-Puzzle-Game)

[Tic Tac Toe Game](https://www.javatpoint.com/tic-tac-toe-game)

[Online Exam](https://www.javatpoint.com/online-exam-project-in-java-swing-without-database)

Word Count Example with Pad and Text Color

1. **import** java.awt.\*;
2. **import** javax.swing.\*;
3. **import** java.awt.event.\*;
4. **public** **class** CharCount **extends** JFrame **implements** ActionListener{
5. JLabel lb1,lb2;
6. JTextArea ta;
7. JButton b;
8. JButton pad,text;
9. CharCount(){
10. **super**("Char Word Count Tool - JTP");
11. lb1=**new** JLabel("Characters: ");
12. lb1.setBounds(50,50,100,20);
13. lb2=**new** JLabel("Words: ");
14. lb2.setBounds(50,80,100,20);
16. ta=**new** JTextArea();
17. ta.setBounds(50,110,300,200);
19. b=**new** JButton("click");
20. b.setBounds(50,320, 80,30);//x,y,w,h
21. b.addActionListener(**this**);
23. pad=**new** JButton("Pad Color");
24. pad.setBounds(140,320, 110,30);//x,y,w,h
25. pad.addActionListener(**this**);
27. text=**new** JButton("Text Color");
28. text.setBounds(260,320, 110,30);//x,y,w,h
29. text.addActionListener(**this**);
31. add(lb1);add(lb2);add(ta);add(b);add(pad);add(text);
33. setSize(400,400);
34. setLayout(**null**);//using no layout manager
35. setVisible(**true**);
36. setDefaultCloseOperation(EXIT\_ON\_CLOSE);
37. }
38. **public** **void** actionPerformed(ActionEvent e){
39. **if**(e.getSource()==b){
40. String text=ta.getText();
41. lb1.setText("Characters: "+text.length());
42. String words[]=text.split("\\s");
43. lb2.setText("Words: "+words.length);
44. }**else** **if**(e.getSource()==pad){
45. Color c=JColorChooser.showDialog(**this**,"Choose Color",Color.BLACK);
46. ta.setBackground(c);
47. }**else** **if**(e.getSource()==text){
48. Color c=JColorChooser.showDialog(**this**,"Choose Color",Color.BLACK);
49. ta.setForeground(c);
50. }
51. }
52. **public** **static** **void** main(String[] args) {
53. **new** CharCount();
54. }}

