

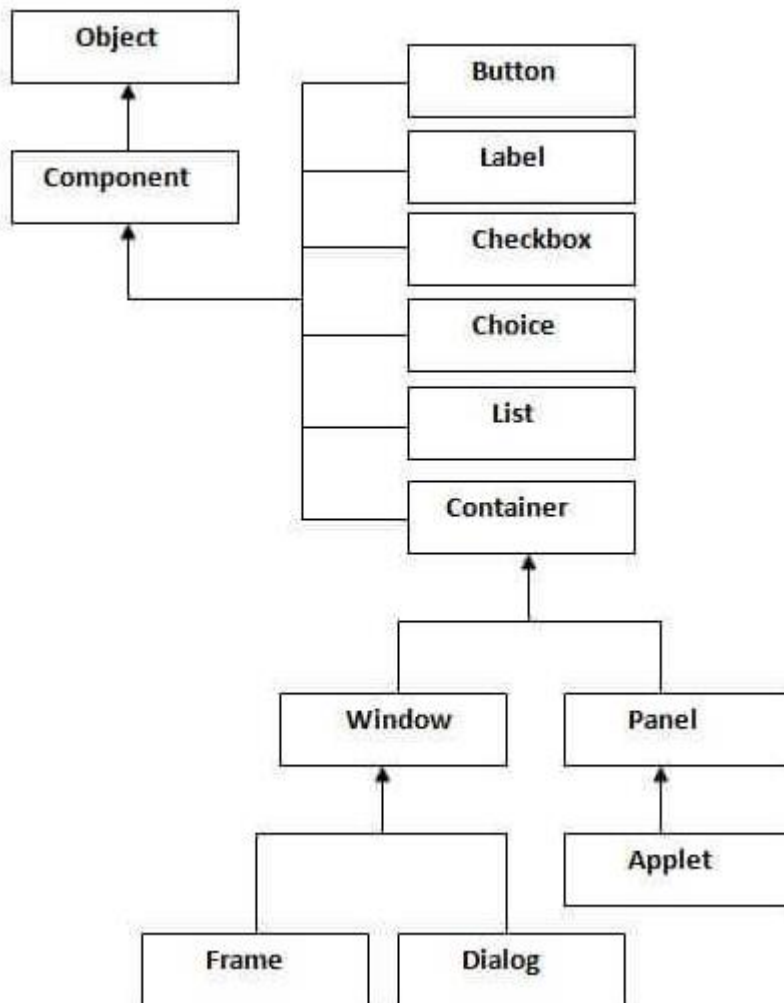
Java AWT

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

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

    }

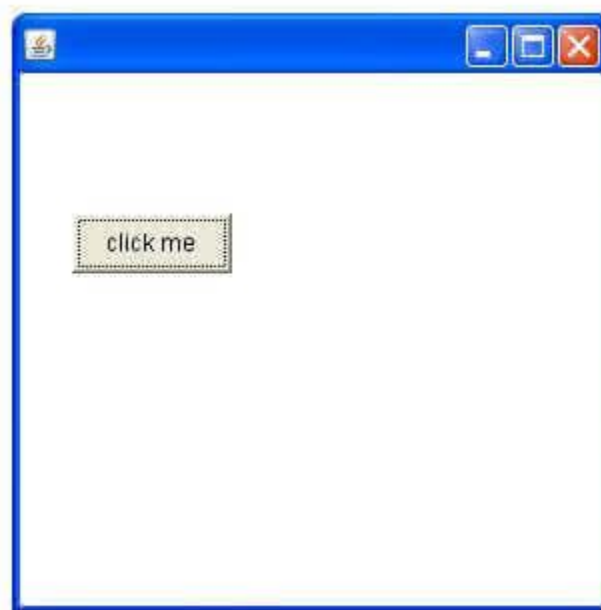
    public static void main(String args[]){

        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.

Output:



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);

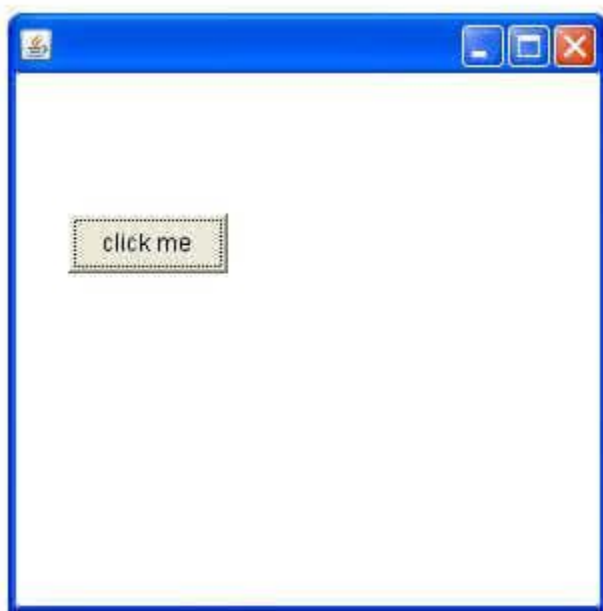
}

public static void main(String args[]){

    First2 f=new First2();

    }}
```

Output:



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

Steps to perform Event Handling

Following steps are required to perform event handling:

- Register the component with the Listener
- Registration Methods

For registering the component with the Listener, many classes provide the registration methods. For example:

Button: public void addActionListener(ActionListener a){ }

MenuItem: public void addActionListener(ActionListener a){ }

TextField: public void addActionListener(ActionListener a){ }

public void addTextListener(TextListener a){ }

TextArea: public void addTextListener(TextListener a){ }

Checkbox: public void addItemListener(ItemListener a){ }

Choice: public void addItemListener(ItemListener a){ }

List: public void addActionListener(ActionListener a){ }

public void addItemListener(ItemListener a){ }

Java event handling by implementing ActionListener

```
import java.awt.*;
```

```
import java.awt.event.*;
```

```
class AEvent extends Frame implements ActionListener{
```

```
    TextField tf;
```

```
    AEvent(){
```

```
        //create components
```

```
        tf=new TextField();
```

```
        tf.setBounds(60,50,170,20);
```

```
        Button b=new Button("click me");
```

```
        b.setBounds(100,120,80,30);
```

```
//register listener  
  
b.addActionListener(this);//passing current instance
```

```
//add components and set size, layout and visibility
```

```
add(b);add(tf);
```

```
setSize(300,300);
```

```
setLayout(null);
```

```
setVisible(true);
```

```
}
```

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

```
tf.setText("Welcome");
```

```
}
```

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

```
new AEvent();
```

```
}
```

```
}
```

public void setBounds(int xaxis, int yaxis, int width, int height); have been used in the above example that sets the position of the component it may be button, textfield etc.



Output:

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

```
public class Button extends Component implements Accessible
```

Java AWT Button Example

```
import java.awt.*;

public class ButtonExample {

    public static void main(String[] args) {

        Frame f=new Frame("Button Example");

        Button b=new Button("Click Here");

        b.setBounds(50,100,80,30);

        f.add(b);

        f.setSize(400,400);

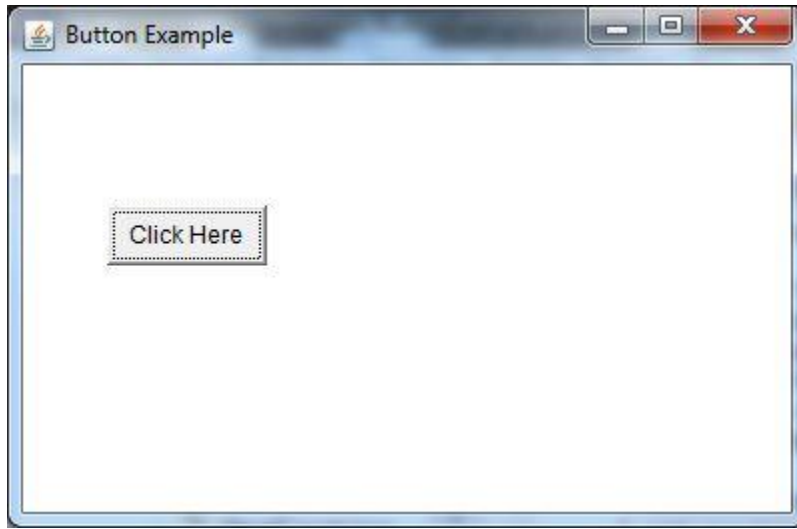
        f.setLayout(null);

        f.setVisible(true);

    }

}
```

Output:



Java AWT Button Example with ActionListener

```
import java.awt.*;

import java.awt.event.*;

public class ButtonExample {

    public static void main(String[] args) {

        Frame f=new Frame("Button Example");

        final TextField tf=new TextField();

        tf.setBounds(50,50, 150,20);

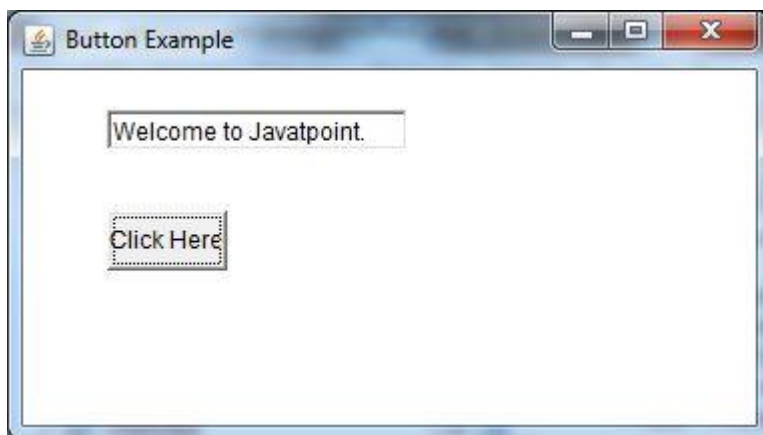
        Button b=new Button("Click Here");

        b.setBounds(50,100,60,30);

        b.addActionListener(new ActionListener(){
```

```
public void actionPerformed(ActionEvent e){  
    tf.setText("Welcome to Javatpoint.");  
}  
});  
f.add(b);f.add(tf);  
f.setSize(400,400);  
f.setLayout(null);  
f.setVisible(true);  
}  
}
```

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

```
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);

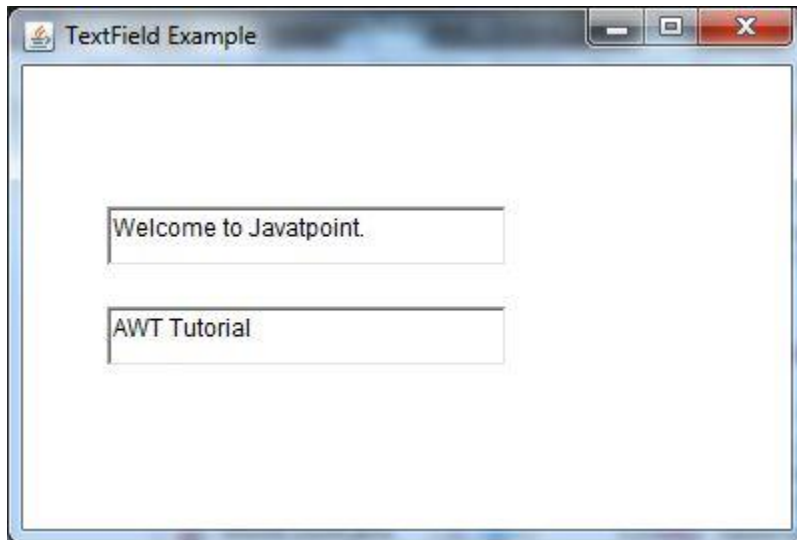
    f.setLayout(null);

    f.setVisible(true);

}
```

```
}
```

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);

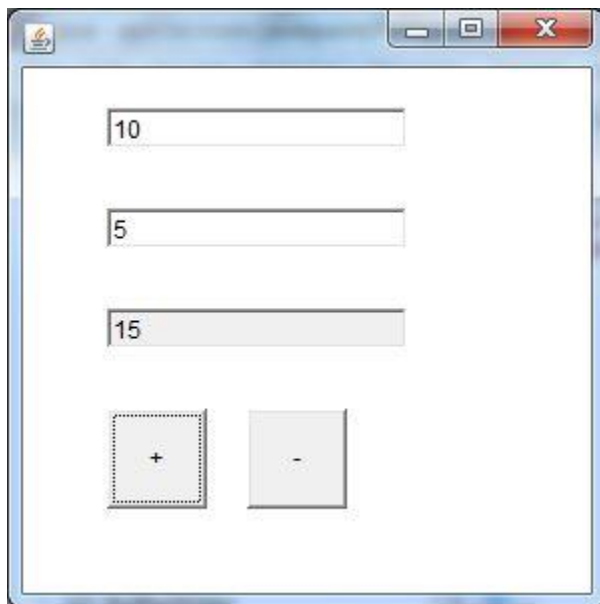
    int c=0;

    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();  
}  
}
```

Output:



Java AWT TextArea

The object of a TextArea class is a multi line region that displays text. It allows the editing of multiple line text. It inherits TextComponent class.

AWT TextArea Class Declaration

```
public class TextArea extends TextComponent
```

Java AWT TextArea Example

```
import java.awt.*;

public class TextAreaExample
{
    TextAreaExample(){
        Frame f= new Frame();

        TextArea area=new TextArea("Welcome to javatpoint");

        area.setBounds(10,30, 300,300);

        f.add(area);

        f.setSize(400,400);

        f.setLayout(null);

        f.setVisible(true);
    }

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

```
new TextAreaExample();  
  
}  
  
}
```

Output:



Java AWT TextArea Example with ActionListener

```
import java.awt.*;  
  
import java.awt.event.*;  
  
public class TextAreaExample extends Frame implements ActionListener{
```

Label l1,l2;

TextArea area;

Button b;

TextAreaExample(){

l1=new Label();

l1.setBounds(50,50,100,30);

l2=new Label();

l2.setBounds(160,50,100,30);

area=new TextArea();

area.setBounds(20,100,300,300);

b=new Button("Count Words");

b.setBounds(100,400,100,30);

b.addActionListener(this);

add(l1);add(l2);add(area);add(b);

setSize(400,450);

setLayout(null);

setVisible(true);

}

public void actionPerformed(ActionEvent e){

String text=area.getText();

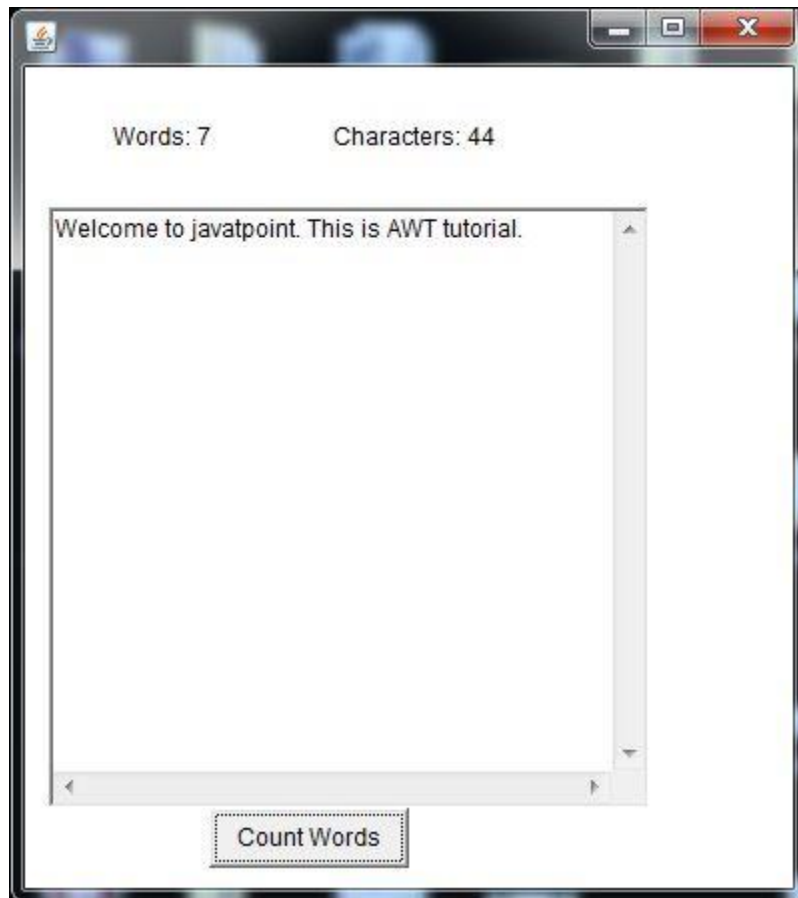
String words[]=text.split("\\s");

l1.setText("Words: "+words.length);

l2.setText("Characters: "+text.length());

```
}  
  
public static void main(String[] args) {  
    new TextAreaExample();  
}  
}
```

Output:



Java AWT Checkbox

The Checkbox 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".

AWT Checkbox Class Declaration

public class Checkbox extends Component implements ItemSelectable, Accessible

Java AWT Checkbox Example

```
import java.awt.*;

public class CheckboxExample
{
    CheckboxExample(){
        Frame f= new Frame("Checkbox Example");

        Checkbox checkbox1 = new Checkbox("C++");

        checkbox1.setBounds(100,100, 50,50);

        Checkbox checkbox2 = new Checkbox("Java", true);

        checkbox2.setBounds(100,150, 50,50);

        f.add(checkbox1);

        f.add(checkbox2);

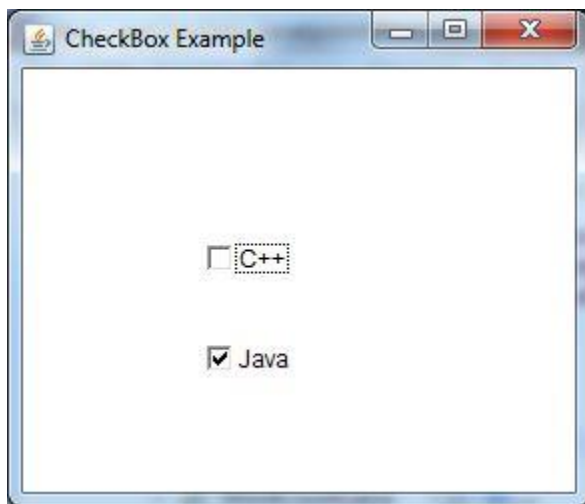
        f.setSize(400,400);

        f.setLayout(null);

        f.setVisible(true);
    }
}
```

```
    }  
  
    public static void main(String args[])  
    {  
        new CheckboxExample();  
    }  
}
```

Output



Java AWT Checkbox Example with ItemListener

```
import java.awt.*;  
  
import java.awt.event.*;  
  
public class CheckboxExample
```

```
{  
  
CheckboxExample(){  
  
    Frame f= new Frame("CheckBox Example");  
  
    final Label label = new Label();  
  
    label.setAlignment(Label.CENTER);  
  
    label.setSize(400,100);  
  
    Checkbox checkbox1 = new Checkbox("C++");  
  
    checkbox1.setBounds(100,100, 50,50);  
  
    Checkbox checkbox2 = new Checkbox("Java");  
  
    checkbox2.setBounds(100,150, 50,50);  
  
    f.add(checkbox1); f.add(checkbox2); f.add(label);  
  
    checkbox1.addItemListener(new ItemListener() {  
  
        public void itemStateChanged(ItemEvent e) {  
  
            label.setText("C++ Checkbox: "  
  
                + (e.getStateChange()==1?"checked":"unchecked"));  
  
        }  
  
    });  
  
    checkbox2.addItemListener(new ItemListener() {  
  
        public void itemStateChanged(ItemEvent e) {  
  
            label.setText("Java Checkbox: "  
  
                + (e.getStateChange()==1?"checked":"unchecked"));  
  
        }  
  
    });  
  
}
```

```
f.setSize(400,400);

f.setLayout(null);

f.setVisible(true);

}

public static void main(String args[])

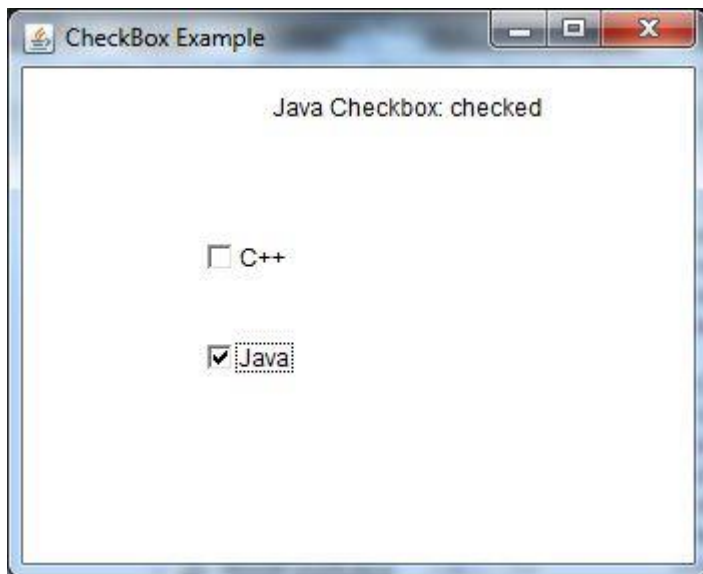
{

    new CheckboxExample();

}

}
```

Output:



Java AWT Choice

The object of Choice class is used to show popup menu of choices. Choice selected by user is shown on the top of a menu. It inherits Component class.

AWT Choice Class Declaration

public class Choice extends Component implements ItemSelectable, Accessible

Java AWT Choice Example

```
import java.awt.*;

public class ChoiceExample
{
    ChoiceExample(){
        Frame f= new Frame();

        Choice c=new Choice();

        c.setBounds(100,100, 75,75);

        c.add("Item 1");

        c.add("Item 2");

        c.add("Item 3");

        c.add("Item 4");

        c.add("Item 5");

        f.add(c);

        f.setSize(400,400);
```

```
f.setLayout(null);

f.setVisible(true);

}

public static void main(String args[])

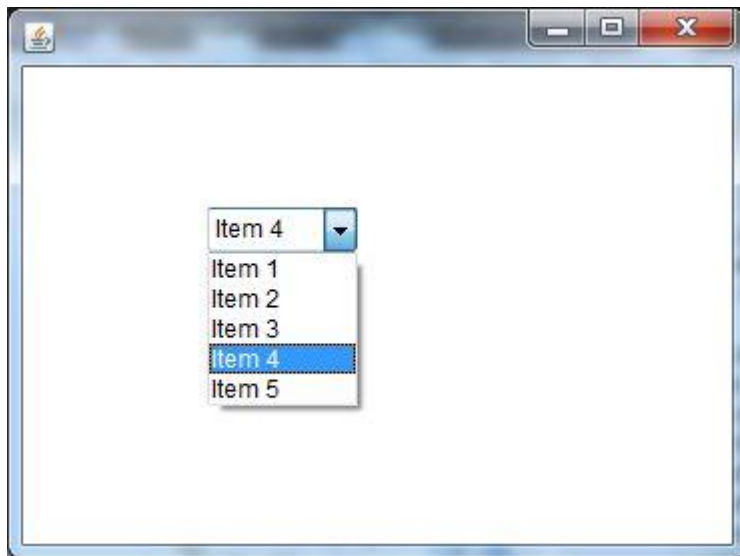
{

    new ChoiceExample();

}

}
```

Output:



Java AWT Choice Example with ActionListener

```
import java.awt.*;

import java.awt.event.*;
```

```
public class ChoiceExample
{
    ChoiceExample(){
        Frame f= new Frame();

        final Label label = new Label();

        label.setAlignment(Label.CENTER);

        label.setSize(400,100);

        Button b=new Button("Show");

        b.setBounds(200,100,50,20);

        final Choice c=new Choice();

        c.setBounds(100,100, 75,75);

        c.add("C");

        c.add("C++");

        c.add("Java");

        c.add("PHP");

        c.add("Android");

        f.add(c);f.add(label); f.add(b);

        f.setSize(400,400);

        f.setLayout(null);

        f.setVisible(true);

        b.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                String data = "Programming language Selected: "+ c.getItem(c.getSelectedIndex());
```

```
        label.setText(data);  
    }  
});  
}  
  
public static void main(String args[])  
{  
    new ChoiceExample();  
}  
}
```

Output:



Java AWT List

The object of List class represents a list of text items. By the help of list, user can choose either one item or multiple items. It inherits Component class.

AWT List class Declaration

```
public class List extends Component implements ItemSelectable, Accessible
```

Java AWT List Example

```
import java.awt.*;

public class ListExample
{
    ListExample(){
        Frame f= new Frame();

        List l1=new List(5);

        l1.setBounds(100,100, 75,75);

        l1.add("Item 1");

        l1.add("Item 2");

        l1.add("Item 3");

        l1.add("Item 4");

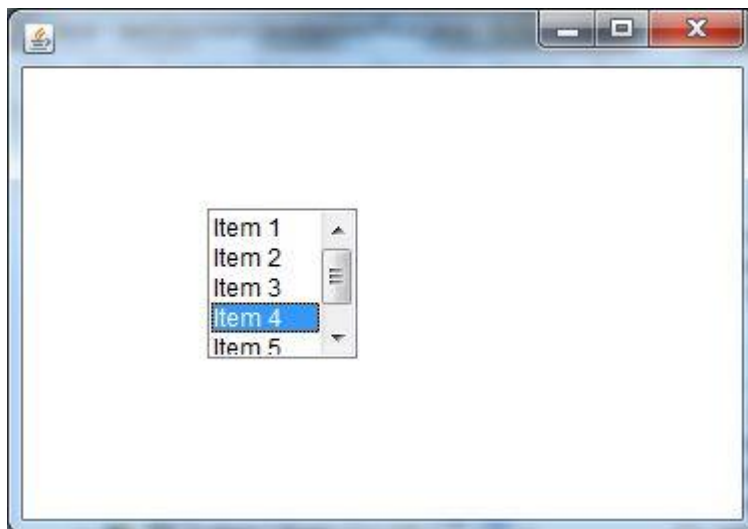
        l1.add("Item 5");

        f.add(l1);

        f.setSize(400,400);
```

```
f.setLayout(null);  
  
f.setVisible(true);  
  
}  
  
public static void main(String args[])  
{  
    new ListExample();  
}  
}
```

Output



Java AWT List Example with ActionListener

```
import java.awt.*;

import java.awt.event.*;

public class ListExample
{
    ListExample(){
        Frame f= new Frame();

        final Label label = new Label();

        label.setAlignment(Label.CENTER);

        label.setSize(500,100);

        Button b=new Button("Show");

        b.setBounds(200,150,80,30);

        final List l1=new List(4, false);

        l1.setBounds(100,100, 70,70);

        l1.add("C");

        l1.add("C++");

        l1.add("Java");

        l1.add("PHP");

        final List l2=new List(4, true);

        l2.setBounds(100,200, 70,70);

        l2.add("Turbo C++");

        l2.add("Spring");
```

```
l2.add("Hibernate");

l2.add("CodeIgniter");

f.add(l1); f.add(l2); f.add(label); f.add(b);

f.setSize(450,450);

f.setLayout(null);

f.setVisible(true);

b.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {

        String data = "Programming language Selected: "+l1.getSelectedItem();

        data += ", Framework Selected:";

        for(String frame:l2.getSelectedItems()){

            data += frame + " ";

        }

        label.setText(data);

    }

});

}

public static void main(String args[])

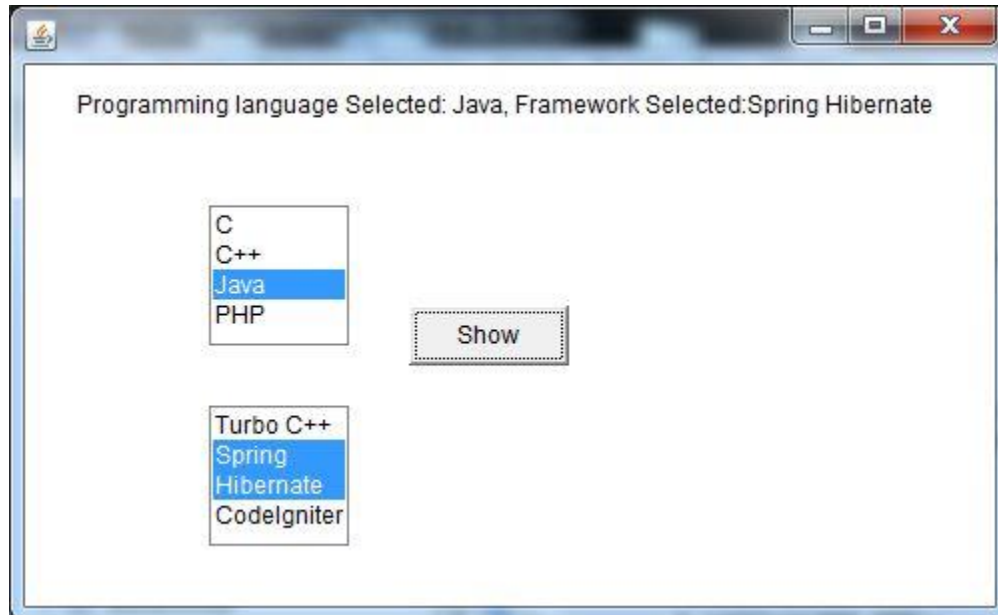
{

    new ListExample();

}

}
```


Output:



Java AWT Scrollbar

The object of Scrollbar class is used to add horizontal and vertical scrollbar. Scrollbar is a GUI component allows us to see invisible number of rows and columns.

AWT Scrollbar class declaration

public class Scrollbar extends Component implements Adjustable, Accessible

Java AWT Scrollbar Example

```
import java.awt.*;

class ScrollbarExample{

ScrollbarExample(){

    Frame f= new Frame("Scrollbar Example");

    Scrollbar s=new Scrollbar();

    s.setBounds(100,100, 50,100);

    f.add(s);

    f.setSize(400,400);

    f.setLayout(null);

    f.setVisible(true);

}

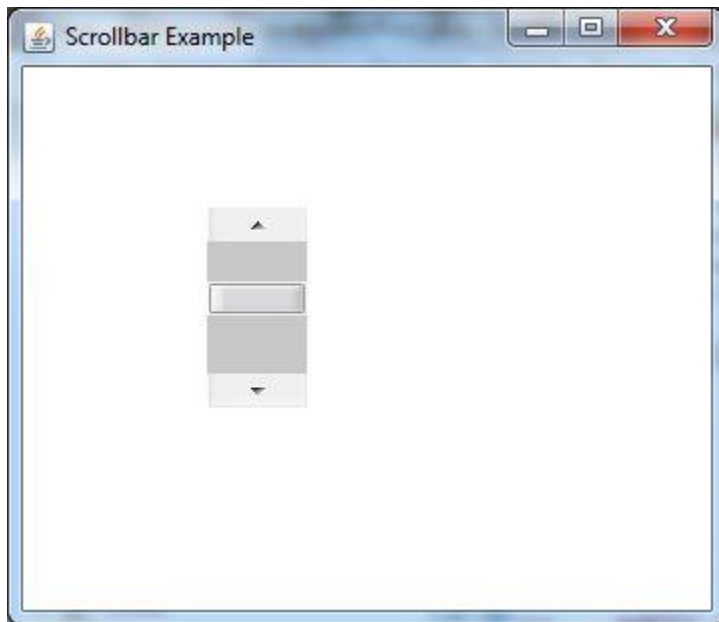
public static void main(String args[]){

    new ScrollbarExample();

}
```

```
}
```

Output



Java AWT Scrollbar Example with AdjustmentListener

```
import java.awt.*;

import java.awt.event.*;

class ScrollbarExample{

    ScrollbarExample(){

        Frame f= new Frame("Scrollbar Example");

        final Label label = new Label();

        label.setAlignment(Label.CENTER);
```

```
        label.setSize(400,100);

        final Scrollbar s=new Scrollbar();

        s.setBounds(100,100, 50,100);

        f.add(s);f.add(label);

        f.setSize(400,400);

        f.setLayout(null);

        f.setVisible(true);

        s.addAdjustmentListener(new AdjustmentListener() {

            public void adjustmentValueChanged(AdjustmentEvent e) {

                label.setText("Vertical Scrollbar value is:"+ s.getValue());

            }

        });

    }

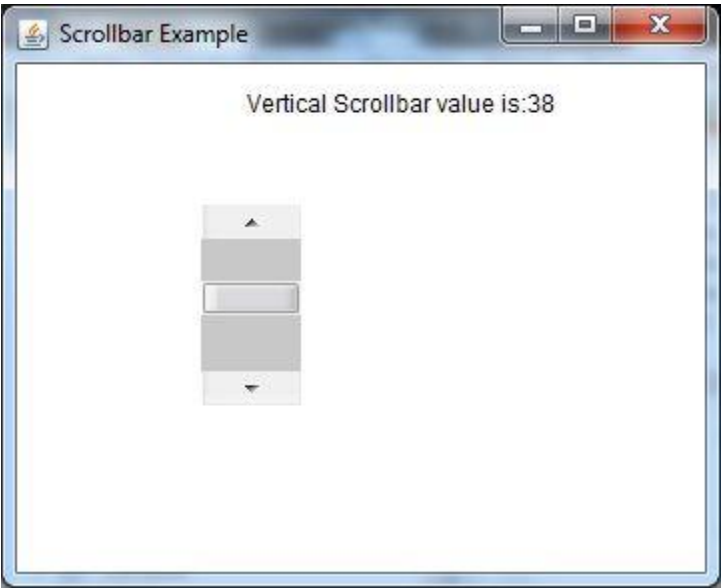
    public static void main(String args[]){

        new ScrollbarExample();

    }

}
```

Output:



Java AWT MenuItem and Menu

The object of MenuItem class adds a simple labeled menu item on menu. The items used in a menu must belong to the MenuItem or any of its subclass.

The object of Menu class is a pull down menu component which is displayed on the menu bar. It inherits the MenuItem class.

AWT MenuItem class declaration

```
public class MenuItem extends MenuComponent implements Accessible
```

AWT Menu class declaration

```
public class Menu extends MenuItem implements MenuContainer, Accessible
```

Java AWT MenuItem and Menu Example

```
import java.awt.*;

class MenuExample
{
    MenuExample(){
        Frame f= new Frame("Menu and MenuItem Example");

        MenuBar mb=new MenuBar();

        Menu menu=new Menu("Menu");

        Menu submenu=new Menu("Sub Menu");

        MenuItem i1=new MenuItem("Item 1");

        MenuItem i2=new MenuItem("Item 2");
```

```
MenuItem i3=new MenuItem("Item 3");

MenuItem i4=new MenuItem("Item 4");

MenuItem i5=new MenuItem("Item 5");

menu.add(i1);

menu.add(i2);

menu.add(i3);

submenu.add(i4);

submenu.add(i5);

menu.add(submenu);

mb.add(menu);

f.setMenuBar(mb);

f.setSize(400,400);

f.setLayout(null);

f.setVisible(true);

}

public static void main(String args[])

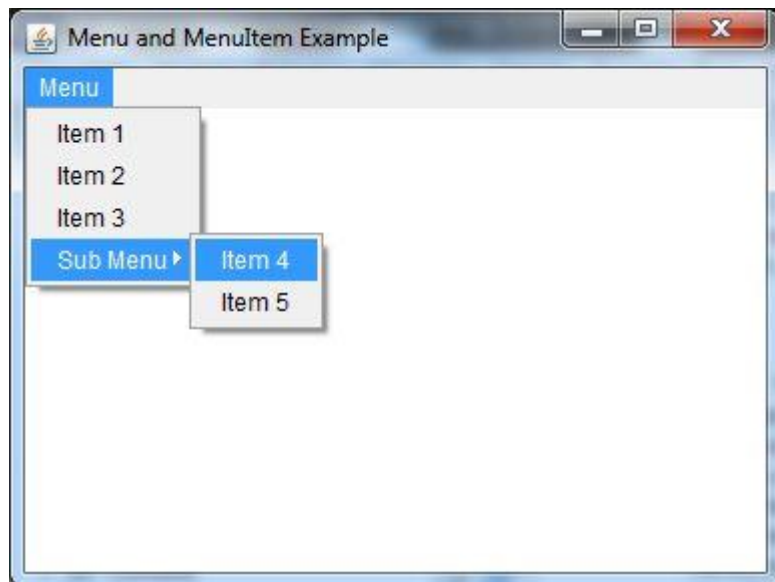
{

new MenuExample();

}

}
```

Output:



Java AWT Canvas

The Canvas control represents a blank rectangular area where the application can draw or trap input events from the user. It inherits the Component class.

AWT Canvas class Declaration

```
public class Canvas extends Component implements Accessible
```

Java AWT Canvas Example

```
import java.awt.*;

public class CanvasExample
{
    public CanvasExample()
    {
        Frame f= new Frame("Canvas Example");

        f.add(new MyCanvas());

        f.setLayout(null);

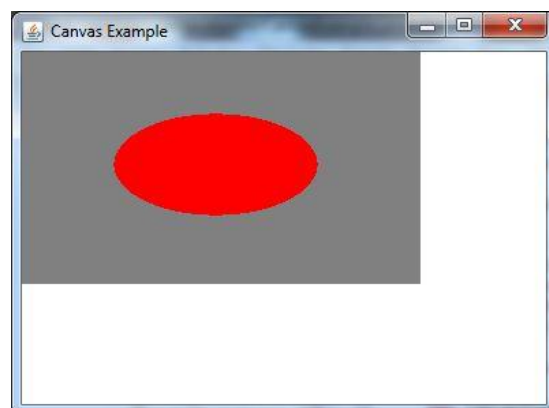
        f.setSize(400, 400);

        f.setVisible(true);
    }

    public static void main(String args[])
    {
        new CanvasExample();
    }
}
```

```
}  
  
}  
  
class MyCanvas extends Canvas  
{  
  
    public MyCanvas() {  
  
        setBackground (Color.GRAY);  
  
        setSize(300, 200);  
  
    }  
  
    public void paint(Graphics g)  
  
    {  
  
        g.setColor(Color.red);  
  
        g.fillOval(75, 75, 150, 75);  
  
    }  
  
}
```

Output:



How to close AWT Window in Java

We can close the AWT Window or Frame by calling `dispose()` or `System.exit()` inside `windowClosing()` method. The `windowClosing()` method is found in `WindowListener` interface and `WindowAdapter` class.

The `WindowAdapter` class implements `WindowListener` interfaces. It provides the default implementation of all the 7 methods of `WindowListener` interface. To override the `windowClosing()` method, you can either use `WindowAdapter` class or `WindowListener` interface.

If you implement the `WindowListener` interface, you will be forced to override all the 7 methods of `WindowListener` interface. So it is better to use `WindowAdapter` class.

overriding `windowClosing()` method:

Close AWT Window Example 3: implementing `WindowListener`

```
import java.awt.*;

import java.awt.event.WindowEvent;

import java.awt.event.WindowListener;

public class WindowExample extends Frame implements WindowListener{

    WindowExample(){

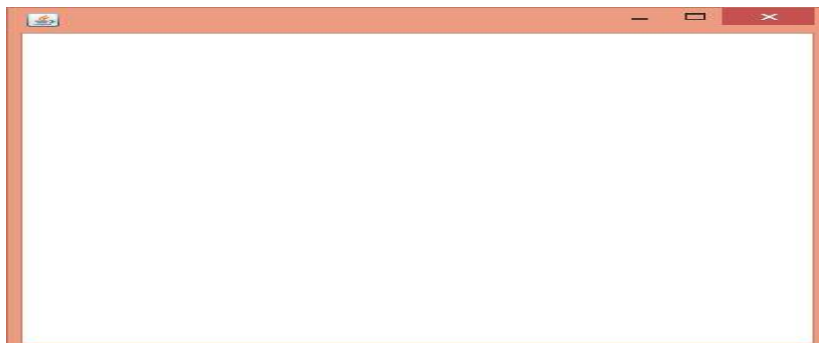
        addWindowListener(this);

        setSize(400,400);
```

```
        setLayout(null);  
  
        setVisible(true);  
    }
```

```
public static void main(String[] args) {  
    new WindowExample();  
}  
  
public void windowActivated(WindowEvent e) {}  
  
public void windowClosed(WindowEvent e) {}  
  
public void windowClosing(WindowEvent e) {  
    dispose();  
}  
  
public void windowDeactivated(WindowEvent e) {}  
  
public void windowDeiconified(WindowEvent e) {}  
  
public void windowIconified(WindowEvent e) {}  
  
public void windowOpened(WindowEvent arg0) {}  
}
```

Output:



Java Swing

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

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

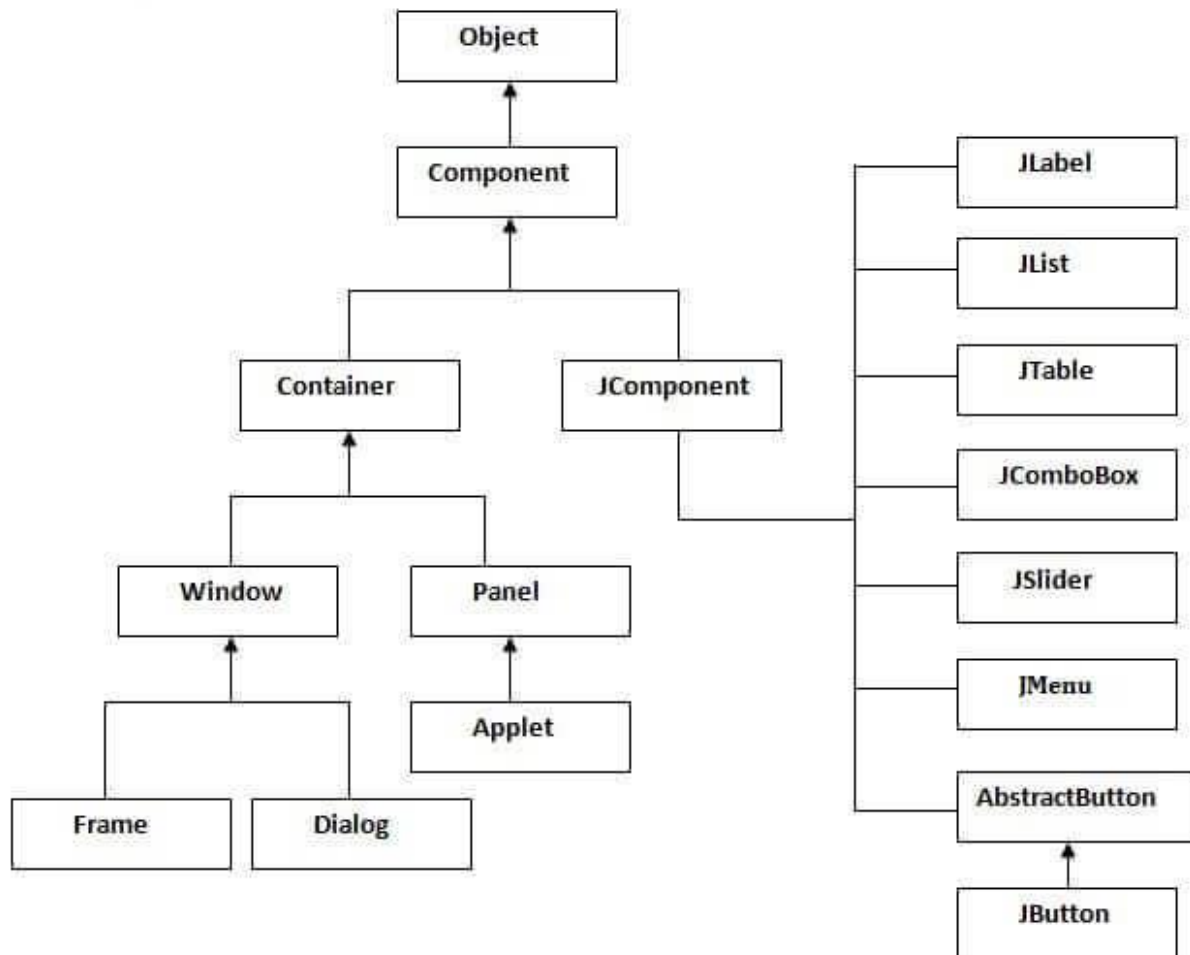
Difference between AWT and Swing

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 .

Hierarchy of Java Swing classes

The hierarchy of java swing API is given below.



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.

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.

```
import javax.swing.*;

public class FirstSwingExample {

    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

        f.add(b);//adding button in JFrame

        f.setSize(400,500);//400 width and 500 height
```

```
f.setLayout(null);//using no layout managers
```

```
f.setVisible(true);//making the frame visible
```

```
}
```

```
}
```

Output:



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.

```
public class JButton extends AbstractButton implements Accessible
```

Commonly used Constructors:

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

Commonly used Methods of AbstractButton class:

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

Java JButton Example

```
import javax.swing.*;

public class ButtonExample {

    public static void main(String[] args) {

        JFrame f=new JFrame("Button Example");

        JButton b=new JButton("Click Here");

        b.setBounds(50,100,95,30);

        f.add(b);

        f.setSize(400,400);

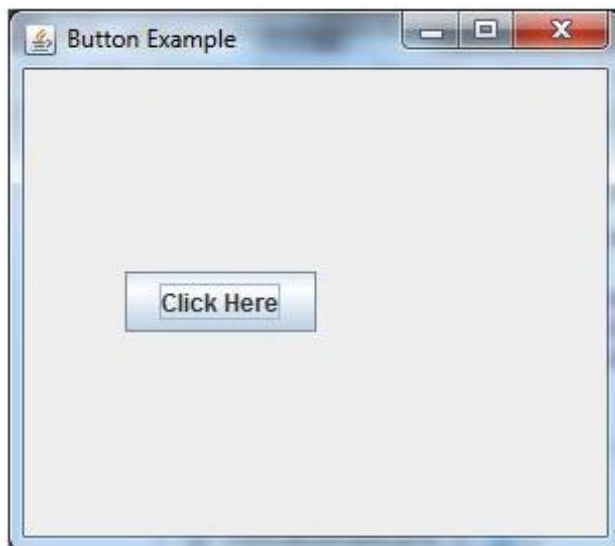
        f.setLayout(null);

        f.setVisible(true);

    }

}
```

Output:



Java JButton Example with ActionListener

```
import java.awt.event.*;

import javax.swing.*;

public class ButtonExample {

    public static void main(String[] args) {

        JFrame f=new JFrame("Button Example");

        final JTextField tf=new JTextField();

        tf.setBounds(50,50, 150,20);

        JButton b=new JButton("Click Here");

        b.setBounds(50,100,95,30);

        b.addActionListener(new ActionListener(){

            public void actionPerformed(ActionEvent e){

                tf.setText("Welcome to Javatpoint.");

            }

        });

        f.add(b);f.add(tf);

        f.setSize(400,400);

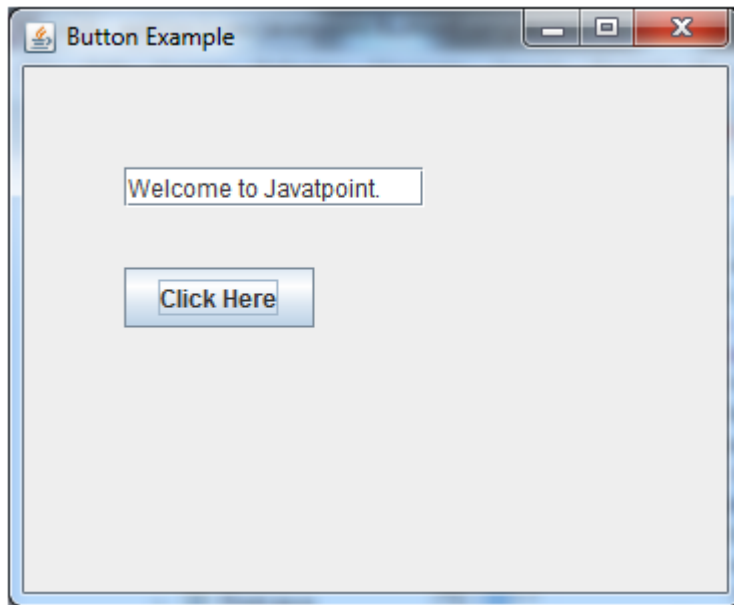
        f.setLayout(null);

        f.setVisible(true);

    }

}
```

Output:



Example of displaying image on the button:

```
import javax.swing.*;

public class ButtonExample{

    ButtonExample(){

        JFrame f=new JFrame("Button Example");

        JButton b=new JButton(new ImageIcon("D:\\icon.png"));

        b.setBounds(100,100,100, 40);

        f.add(b);

        f.setSize(300,400);

        f.setLayout(null);

        f.setVisible(true);
```

```
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
}
```

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

```
    new ButtonExample();
```

```
}
```

```
}
```



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.

```
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

```
import javax.swing.*;

class LabelExample

{

public static void main(String args[])

    {

        JFrame f= new JFrame("Label Example");

        JLabel l1,l2;

        l1=new JLabel("First Label.");

        l1.setBounds(50,50, 100,30);

        l2=new JLabel("Second Label.");

        l2.setBounds(50,100, 100,30);

        f.add(l1); f.add(l2);

        f.setSize(300,300);

        f.setLayout(null);

        f.setVisible(true);

    }

}
```

Output:



Java JLabel Example with ActionListener

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class LabelExample extends Frame implements ActionListener{

    JTextField tf; JLabel l; JButton b;

    LabelExample(){

        tf=new JTextField();

        tf.setBounds(50,50, 150,20);

        l=new JLabel();

        l.setBounds(50,100, 250,20);

        b=new JButton("Find IP");
```



```
b.setBounds(50,150,95,30);

b.addActionListener(this);

add(b);add(tf);add(l);

setSize(400,400);

setLayout(null);

setVisible(true);

}

public void actionPerformed(ActionEvent e) {

    try{

        String host=tf.getText();

        String ip=java.net.InetAddress.getByName(host).getHostAddress();

        l.setText("IP of "+host+" is: "+ip);

    }catch(Exception ex){System.out.println(ex);}

}

public static void main(String[] args) {

    new LabelExample();

} }
```

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.

```
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

```
import javax.swing.*;

class TextFieldExample

{

public static void main(String args[])

    {

        JFrame f= new JFrame("TextField Example");

        JTextField t1,t2;

        t1=new JTextField("Welcome to Javatpoint.");

        t1.setBounds(50,100, 200,30);

        t2=new JTextField("AWT Tutorial");

        t2.setBounds(50,150, 200,30);

        f.add(t1); f.add(t2);

        f.setSize(400,400);

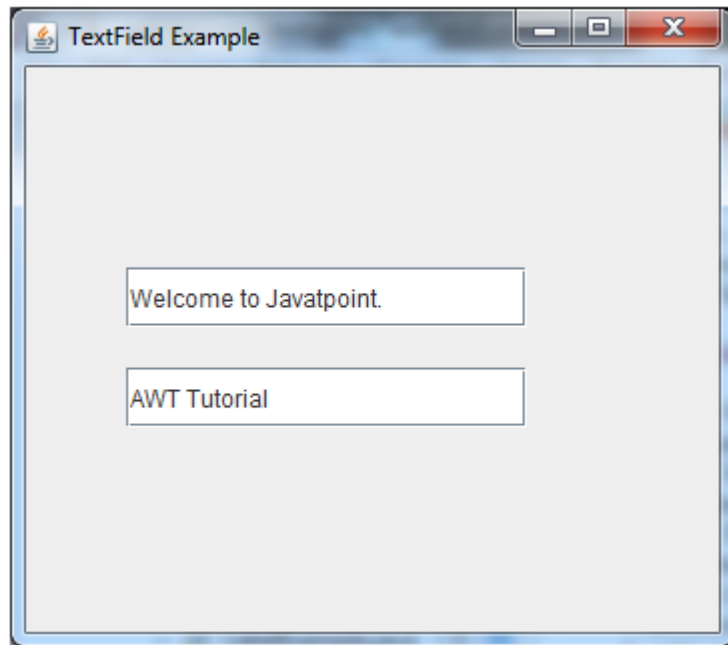
        f.setLayout(null);

        f.setVisible(true);

    }

}
```

Output:



Java JTextField Example with ActionListener

```
import javax.swing.*;
import java.awt.event.*;

public class TextFieldExample implements ActionListener{

    JTextField tf1,tf2,tf3;

    JButton b1,b2;

    TextFieldExample(){

        JFrame f= new JFrame();

        tf1=new JTextField();
```

```
tf1.setBounds(50,50,150,20);

tf2=new JTextField();

tf2.setBounds(50,100,150,20);

tf3=new JTextField();

tf3.setBounds(50,150,150,20);

tf3.setEditable(false);

b1=new JButton("+");

b1.setBounds(50,200,50,50);

b2=new JButton("-");

b2.setBounds(120,200,50,50);

b1.addActionListener(this);

b2.addActionListener(this);

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

f.setSize(300,300);

f.setLayout(null);

f.setVisible(true);

}

public void actionPerformed(ActionEvent e) {

    String s1=tf1.getText();

    String s2=tf2.getText();

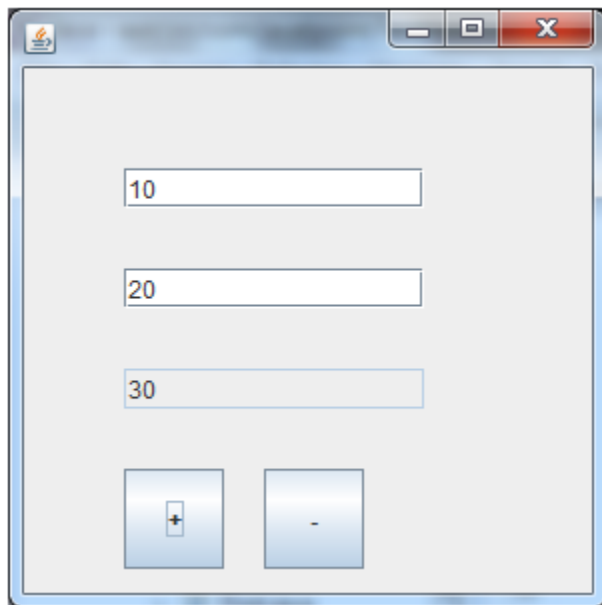
    int a=Integer.parseInt(s1);

    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();  
    } }
```

Output:



Java JTextArea

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

JTextArea class declaration

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

```
public class JTextArea extends JTextComponent
```

Commonly used Constructors:

Constructor	Description
JTextArea()	Creates a text area that displays no text initially.
JTextArea(String s)	Creates a text area that displays specified text initially.
JTextArea(int row, int column)	Creates a text area with the specified number of rows and columns that displays no text initially.
JTextArea(String s, int row, int column)	Creates a text area with the specified number of rows and columns that displays specified text.

Commonly used Methods:

Methods	Description
void setRows(int rows)	It is used to set specified number of rows.
void setColumns(int cols)	It is used to set specified number of columns.
void setFont(Font f)	It is used to set the specified font.
void insert(String s, int position)	It is used to insert the specified text on the specified position.
void append(String s)	It is used to append the given text to the end of the document.

Java JTextArea

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

JTextArea class declaration

```
public class JTextArea extends JTextComponent
```

Commonly used Constructors:

Constructor	Description
-------------	-------------

JTextArea()	Creates a text area that displays no text initially.
--------------------	--

JTextArea(String s)	Creates a text area that displays specified text initially.
----------------------------	---

JTextArea(int row, int column)	Creates a text area with the specified number of rows and columns that displays no text initially.
---------------------------------------	--

JTextArea(String s, int row, int column)	Creates a text area with the specified number of rows and columns that displays specified text.
---	---

Commonly used Methods:

Methods	Description
---------	-------------

void setRows(int rows)	It is used to set specified number of rows.
-------------------------------	---

void setColumns(int cols)	It is used to set specified number of columns.
----------------------------------	--

void setFont(Font f)	It is used to set the specified font.
-----------------------------	---------------------------------------

void insert(String s, int position)	It is used to insert the specified text on the specified position.
--	--

void append(String s)	It is used to append the given text to the end of the document.
------------------------------	---

Java JTextArea Example

```
import javax.swing.*;

public class TextAreaExample
{
    TextAreaExample(){
        JFrame f= new JFrame();

        JTextArea area=new JTextArea("Welcome to javatpoint");

        area.setBounds(10,30, 200,200);

        f.add(area);

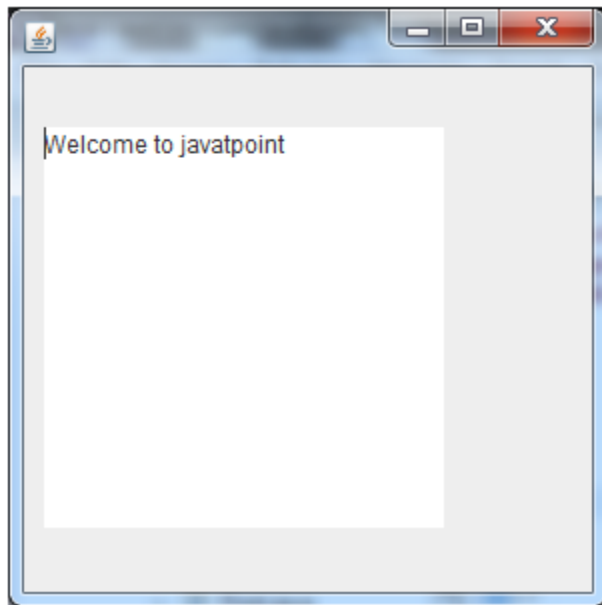
        f.setSize(300,300);

        f.setLayout(null);

        f.setVisible(true);
    }

    public static void main(String args[])
    {
        new TextAreaExample();
    }
}
```

Output:



Java JTextArea Example with ActionListener

```
import javax.swing.*;
import java.awt.event.*;

public class TextAreaExample implements ActionListener{

    JLabel l1,l2;

    JTextArea area;

    JButton b;

    TextAreaExample() {

        JFrame f= new JFrame();

        l1=new JLabel();

        l1.setBounds(50,25,100,30);
```

```
l2=new JLabel();

l2.setBounds(160,25,100,30);

area=new JTextArea();

area.setBounds(20,75,250,200);

b=new JButton("Count Words");

b.setBounds(100,300,120,30);

b.addActionListener(this);

f.add(l1);f.add(l2);f.add(area);f.add(b);

f.setSize(450,450);

f.setLayout(null);

f.setVisible(true);
}

public void actionPerformed(ActionEvent e){

    String text=area.getText();

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

    l1.setText("Words: "+words.length);

    l2.setText("Characters: "+text.length());

}

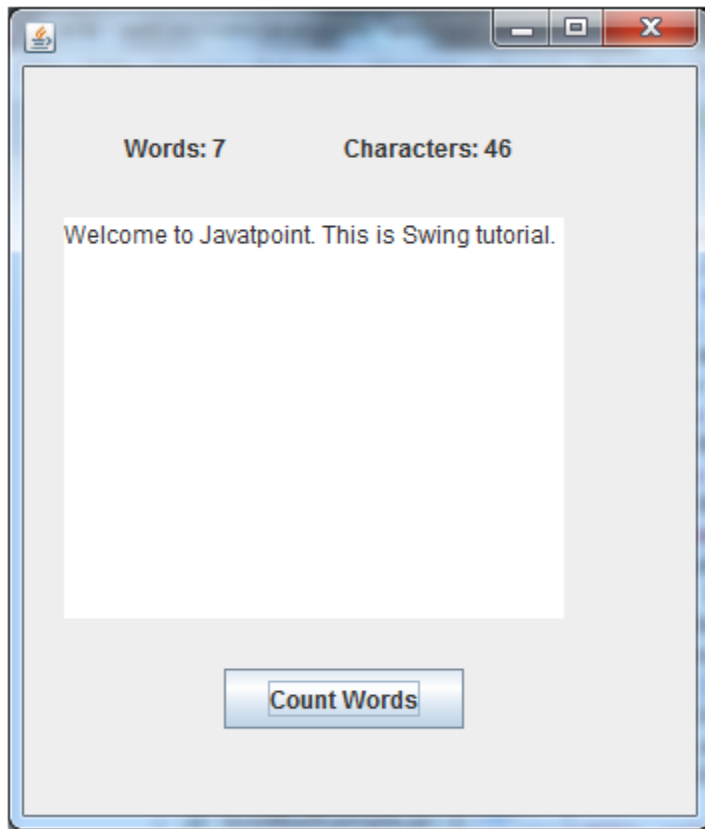
public static void main(String[] args) {

    new TextAreaExample();

}

}
```

Output:



Java JPasswordField

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

JPasswordField class declaration.

```
public class JPasswordField extends JTextField
```

Commonly used Constructors:

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

Java JPasswordField Example

```
import javax.swing.*;

public class PasswordFieldExample {

    public static void main(String[] args) {

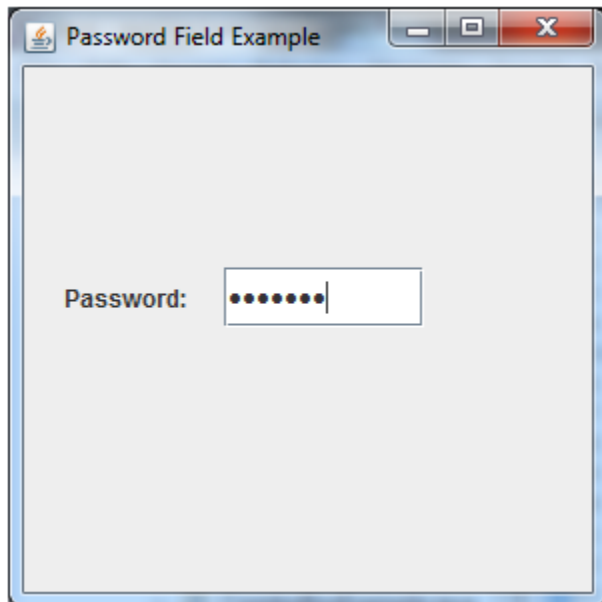
        JFrame f=new JFrame("Password Field Example");

        JPasswordField value = new JPasswordField();

        JLabel l1=new JLabel("Password:");
```

```
l1.setBounds(20,100, 80,30);  
  
value.setBounds(100,100,100,30);  
  
f.add(value); f.add(l1);  
  
f.setSize(300,300);  
  
f.setLayout(null);  
  
f.setVisible(true);  
  
}  
  
}
```

Output:



Java JPasswordField Example with ActionListener

```
import javax.swing.*;

import java.awt.event.*;

public class PasswordFieldExample {

    public static void main(String[] args) {

        JFrame f=new JFrame("Password Field Example");

        final JLabel label = new JLabel();

        label.setBounds(20,150, 200,50);

        final JPasswordField value = new JPasswordField();

        value.setBounds(100,75,100,30);

        JLabel l1=new JLabel("Username:");

        l1.setBounds(20,20, 80,30);

        JLabel l2=new JLabel("Password:");

        l2.setBounds(20,75, 80,30);

        JButton b = new JButton("Login");

        b.setBounds(100,120, 80,30);

        final JTextField text = new JTextField();

        text.setBounds(100,20, 100,30);

        f.add(value); f.add(l1); f.add(label); f.add(l2); f.add(b); f.add(text);

        f.setSize(300,300);

        f.setLayout(null);

        f.setVisible(true);
```



```
b.addActionListener(new ActionListener() {  
  
    public void actionPerformed(ActionEvent e) {  
  
        String data = "Username " + text.getText();  
  
        data += ", Password: "  
  
        + new String(value.getPassword());  
  
        label.setText(data);  
  
    }  
  
});  
  
}  
  
}
```



Java JTable

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

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

```
import javax.swing.*;

public class TableExample {

    JFrame f;

    TableExample(){

        f=new JFrame();

        String data[][]={ {"101","Amit","670000"},
                           {"102","Jai","780000"},
                           {"101","Sachin","700000"} };

        String column[]={ "ID","NAME","SALARY"};

        JTable jt=new JTable(data,column);

        jt.setBounds(30,40,200,300);

        JScrollPane sp=new JScrollPane(jt);
```

```
f.add(sp);

f.setSize(300,400);

f.setVisible(true);

}

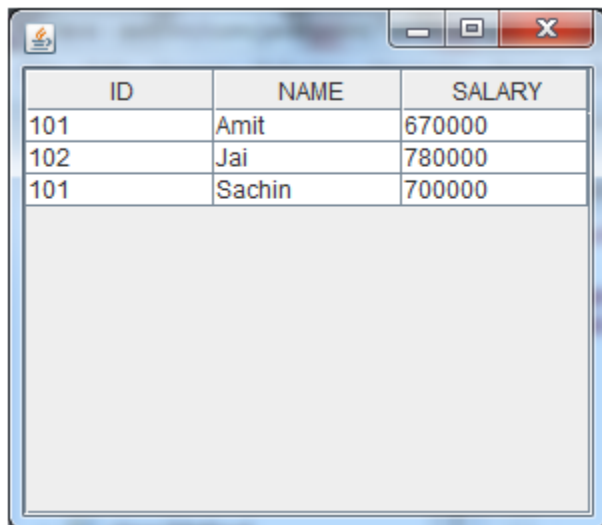
public static void main(String[] args) {

    new TableExample();

}

}
```

Output:



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

Java JTable Example with ListSelectionListener

```
import javax.swing.*;

import javax.swing.event.*;

public class TableExample {

    public static void main(String[] a) {

        JFrame f = new JFrame("Table Example");

        String data[][]={ { "101","Amit","670000"},

                           { "102","Jai","780000"},

                           { "101","Sachin","700000" } };

        String column[]={ "ID","NAME","SALARY" };

        final JTable jt=new JTable(data,column);

        jt.setCellSelectionEnabled(true);

        ListSelectionModel select= jt.getSelectionModel();

        select.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);

        select.addListSelectionListener(new ListSelectionListener() {

            public void valueChanged(ListSelectionEvent e) {

                String Data = null;

                int[] row = jt.getSelectedRows();

                int[] columns = jt.getSelectedColumns();

                for (int i = 0; i < row.length; i++) {

                    for (int j = 0; j < columns.length; j++) {

                        Data = (String) jt.getValueAt(row[i], columns[j]);
```

```
        } }

        System.out.println("Table element selected is: " + Data);

    }

});

JScrollPane sp=new JScrollPane(jt);

f.add(sp);

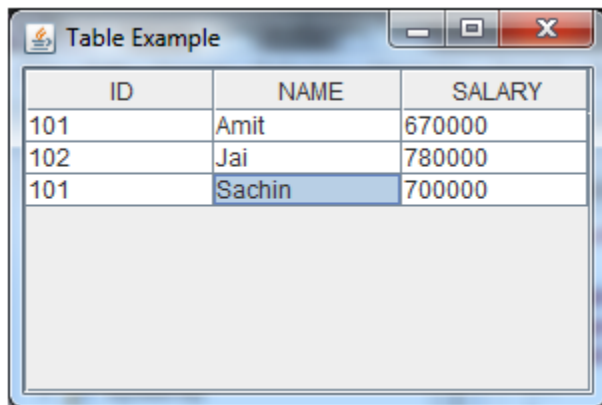
f.setSize(300, 200);

f.setVisible(true);

}

}
```

Output:



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

Example of digital clock in swing:

```
import javax.swing.*;

import java.awt.*;

import java.text.*;

import java.util.*;

public class DigitalWatch implements Runnable{

    JFrame f;

    Thread t=null;

    int hours=0, minutes=0, seconds=0;

    String timeString = "";

    JButton b;

    DigitalWatch(){

        f=new JFrame();

        t = new Thread(this);

        t.start();

        b=new JButton();

        b.setBounds(100,100,100,50);

        f.add(b);
```

```
f.setSize(300,400);

f.setLayout(null);

f.setVisible(true);
}


public void run() {

    try {

        while (true) {

            Calendar cal = Calendar.getInstance();

            hours = cal.get( Calendar.HOUR_OF_DAY );

            if ( hours > 12 ) hours -= 12;

            minutes = cal.get( Calendar.MINUTE );

            seconds = cal.get( Calendar.SECOND );

            SimpleDateFormat formatter = new SimpleDateFormat("hh:mm:ss");

            Date date = cal.getTime();

            timeString = formatter.format( date );

            printTime();

            t.sleep( 1000 ); // interval given in milliseconds

        }

    }

}
```

```
    }

    catch (Exception e) { }

}

public void printTime(){

    b.setText(timeString);

}

public static void main(String[] args) {

    new DigitalWatch();

}

}
```


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();
```

```
}
```

```
}
```