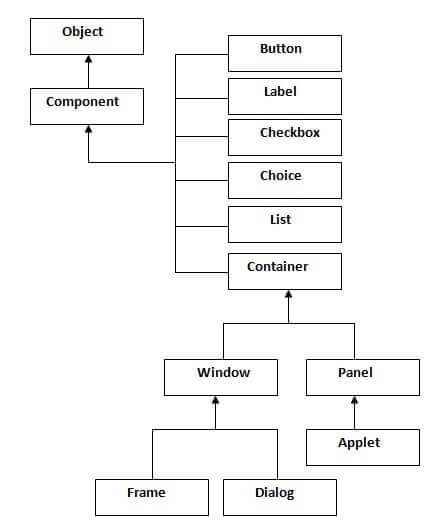
# Java AWT

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

### Java AWT Hierarchy

The hierarchy of Java AWT classes are given below.



|  |  |
| --- | --- |
|  |  |
|  |  |
| 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();

}}



## 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();

}}



# 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 |
| KeyEvent | KeyListener |
| ItemEvent | ItemListener |
| TextEvent | TextListener |
| WindowEvent | WindowListener |

## Steps to perform Event Handling

Following steps are required to perform event handling:

1. 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){}

**Button Example**

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");

}

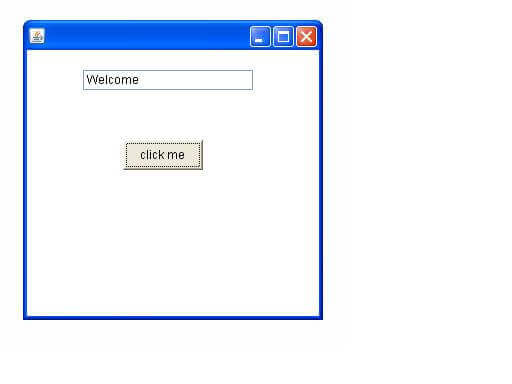
public static void main(String args[]){

new AEvent();

}

}

**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);

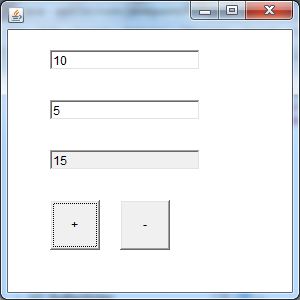
    }

public static void main(String[] args) {

    new TextFieldExample();

}

}



# Java ActionListener Interface

1. public abstract void actionPerformed(ActionEvent e);

import java.awt.\*;

import java.awt.event.\*;

//1st step

public class ActionListenerExample implements ActionListener{

public static void main(String[] args) {

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

    final TextField tf=new TextField();

    tf.setBounds(50,50, 150,20);

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

    b.setBounds(50,100,60,30);

    //2nd step

    b.addActionListener(this);

    f.add(b);f.add(tf);

    f.setSize(400,400);

    f.setLayout(null);

    f.setVisible(true);

}

//3rd step

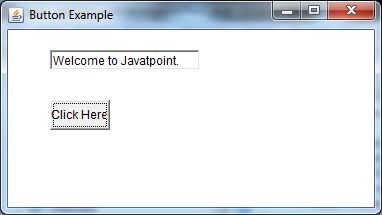
public void actionPerformed(ActionEvent e){

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

}

}

Output:



## 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

import java.awt.\*;

import java.awt.event.\*;

public class MouseListenerExample extends Frame implements MouseListener{

    Label l;

    MouseListenerExample(){

        addMouseListener(this);

        l=new Label();

        l.setBounds(20,50,100,20);

        add(l);

        setSize(300,300);

        setLayout(null);

        setVisible(true);

    }

    public void mouseClicked(MouseEvent e) {

        l.setText("Mouse Clicked");

    }

    public void mouseEntered(MouseEvent e) {

        l.setText("Mouse Entered");

    }

    public void mouseExited(MouseEvent e) {

        l.setText("Mouse Exited");

    }

    public void mousePressed(MouseEvent e) {

        l.setText("Mouse Pressed");

    }

    public void mouseReleased(MouseEvent e) {

        l.setText("Mouse Released");

    }

public static void main(String[] args) {

    new MouseListenerExample();

}

}

## Java KeyListener Example

public abstract void keyPressed(KeyEvent e);

public abstract void keyReleased(KeyEvent e);

public abstract void keyTyped(KeyEvent e);

import java.awt.\*;

import java.awt.event.\*;

public class KeyListenerExample extends Frame implements KeyListener{

    Label l;

    TextArea area;

    KeyListenerExample(){

        l=new Label();

        l.setBounds(20,50,100,20);

        area=new TextArea();

        area.setBounds(20,80,300, 300);

        area.addKeyListener(this);

        add(l);add(area);

        setSize(400,400);

        setLayout(null);

        setVisible(true);

    }

    public void keyPressed(KeyEvent e) {

        l.setText("Key Pressed");

    }

    public void keyReleased(KeyEvent e) {

        l.setText("Key Released");

    }

    public void keyTyped(KeyEvent e) {

        l.setText("Key Typed");

    }

    public static void main(String[] args) {

        new KeyListenerExample();

    }

}

## Methods of WindowListener interface

The signature of 7 methods found in WindowListener interface are given below:

1. public abstract void windowActivated(WindowEvent e);
2. public abstract void windowClosed(WindowEvent e);
3. public abstract void windowClosing(WindowEvent e);
4. public abstract void windowDeactivated(WindowEvent e);
5. public abstract void windowDeiconified(WindowEvent e);
6. public abstract void windowIconified(WindowEvent e);
7. public abstract void windowOpened(WindowEvent e);

## Java WindowListener Example

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 arg0) {

    System.out.println("activated");

}

public void windowClosed(WindowEvent arg0) {

    System.out.println("closed");

}

public void windowClosing(WindowEvent arg0) {

    System.out.println("closing");

    dispose();

}

public void windowDeactivated(WindowEvent arg0) {

    System.out.println("deactivated");

}

public void windowDeiconified(WindowEvent arg0) {

    System.out.println("deiconified");

}

public void windowIconified(WindowEvent arg0) {

    System.out.println("iconified");

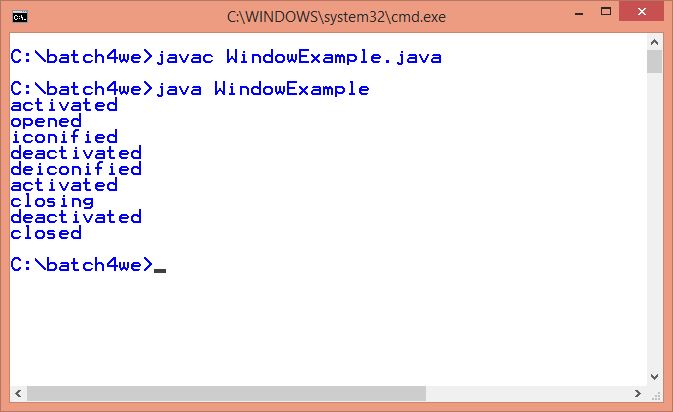
}

public void windowOpened(WindowEvent arg0) {

    System.out.println("opened");

}

}



# Java Swing

### 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**. |



Design components

* The Model contains only the pure application data, it contains no logic describing how to present the data to a user.
* The View presents the model’s data to the user. The view knows how to access the model’s data, but it does not know what this data means or what the user can do to manipulate it.
* The Controller exists between the view and the model. It listens to events triggered by the view (or another external source) and executes the appropriate reaction to these events

### Hierarchy of Java Swing classes



### 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 {

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

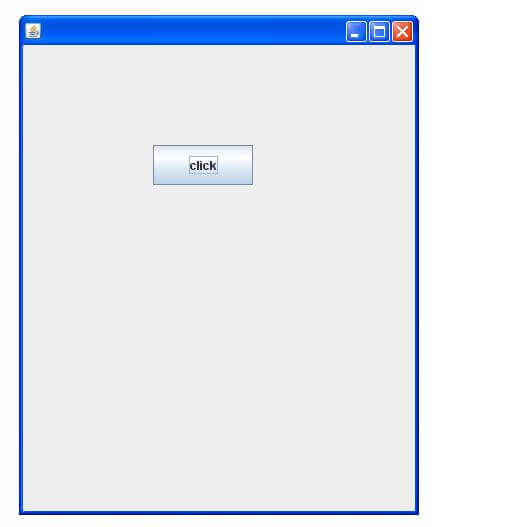
f.setSize(400,500);//400 width and 500 height

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

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

}

}



import javax.swing.\*;

public class Simple2 extends JFrame{//inheriting JFrame

JFrame f;

Simple2(){

JButton b=new JButton("click");//create button

b.setBounds(130,100,100, 40);

add(b);//adding button on frame

setSize(400,500);

setLayout(null);

setVisible(true);

}

public static void main(String[] args) {

new Simple2();

}}

## JpasswordField

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);

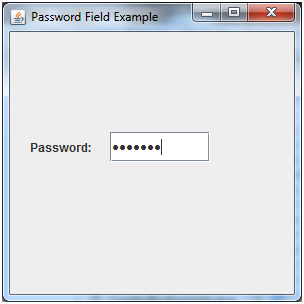
            f.setSize(300,300);

            f.setLayout(null);

            f.setVisible(true);

}

}



import javax.swing.\*;

public class RadioButtonExample {

JFrame f;

RadioButtonExample(){

f=new JFrame();

JRadioButton r1=new JRadioButton("A) Male");

JRadioButton r2=new JRadioButton("B) Female");

r1.setBounds(75,50,100,30);

r2.setBounds(75,100,100,30);

ButtonGroup bg=new ButtonGroup();

bg.add(r1);bg.add(r2);

f.add(r1);f.add(r2);

f.setSize(300,300);

f.setLayout(null);

f.setVisible(true);

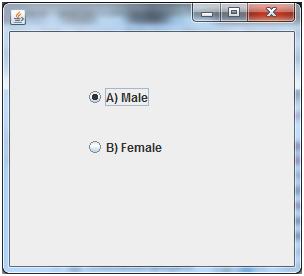
}

public static void main(String[] args) {

    new RadioButtonExample();

}

}



Java JComboBox Example

import javax.swing.\*;

public class ComboBoxExample {

JFrame f;

ComboBoxExample(){

    f=new JFrame("ComboBox Example");

    String country[]={"India","Aus","U.S.A","England","Newzealand"};

    JComboBox cb=new JComboBox(country);

    cb.setBounds(50, 50,90,20);

    f.add(cb);

    f.setLayout(null);

    f.setSize(400,500);

    f.setVisible(true);

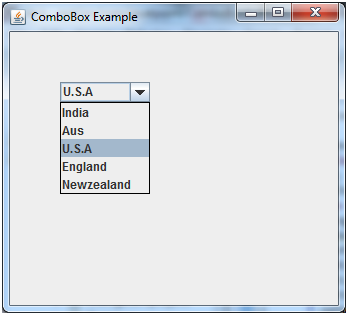
}

public static void main(String[] args) {

    new ComboBoxExample();

}

}



**Jtable:**

**import javax.swing.\*;**

public class TableExample {

    JFrame f;

    TableExample(){

    f=new JFrame();

    String data[][]={ {"101","latha","50000"},

                          {"102","geeta","55000"},

                          {"101","Sarika","52000"}};

    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);

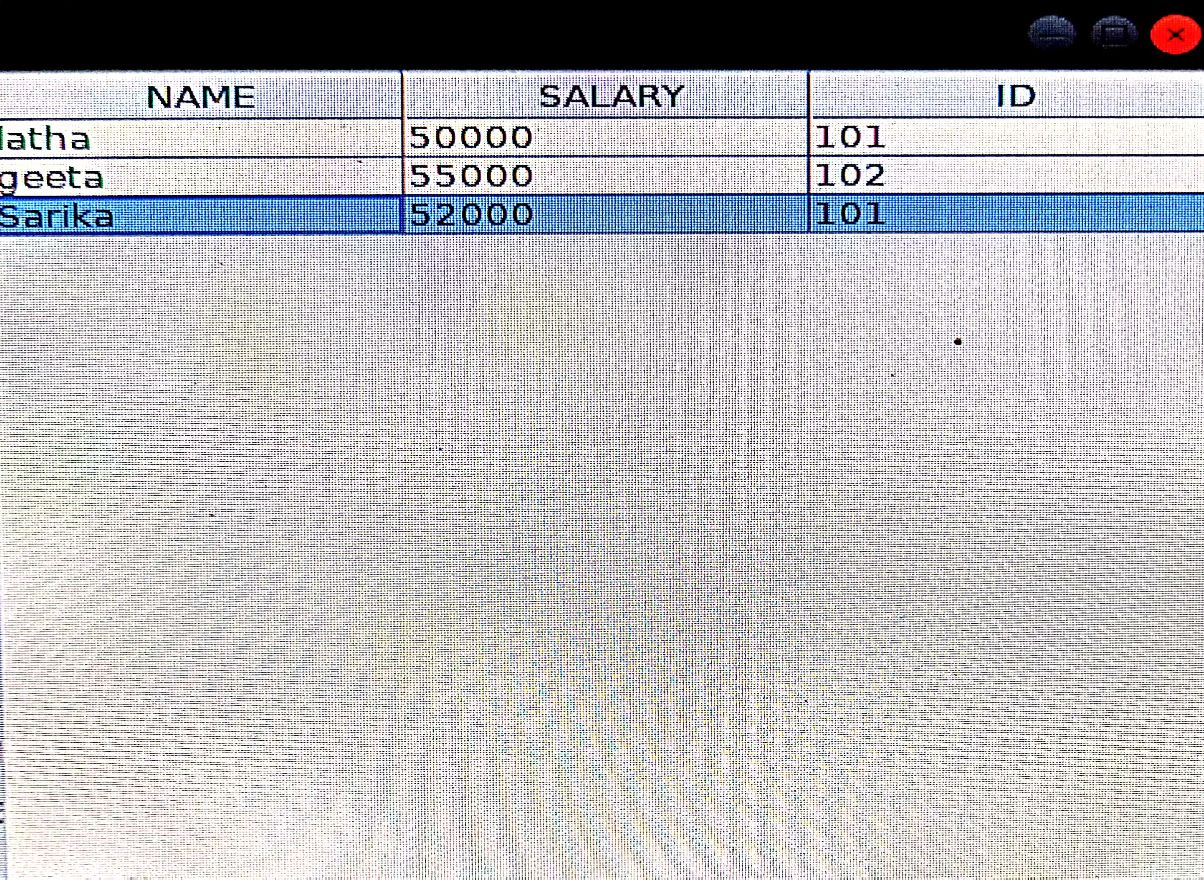
}

public static void main(String[] args) {

    new TableExample();

}

}

****

**Java LayoutManagers**

The LayoutManagers are used to arrange components in a particular manner. LayoutManager is an interface that is implemented by all the classes of layout managers. There are following classes that represents the layout managers:

java.awt.BorderLayout

java.awt.FlowLayout

java.awt.GridLayout

import java.awt.\*;

import javax.swing.\*;

public class Border {

JFrame f;

Border(){

    f=new JFrame();

    JButton b1=new JButton("NORTH");;

    JButton b2=new JButton("SOUTH");;

    JButton b3=new JButton("EAST");;

    JButton b4=new JButton("WEST");;

    JButton b5=new JButton("CENTER");;

    f.add(b1,BorderLayout.NORTH);

    f.add(b2,BorderLayout.SOUTH);

    f.add(b3,BorderLayout.EAST);

    f.add(b4,BorderLayout.WEST);

    f.add(b5,BorderLayout.CENTER);

    f.setSize(300,300);

    f.setVisible(true);

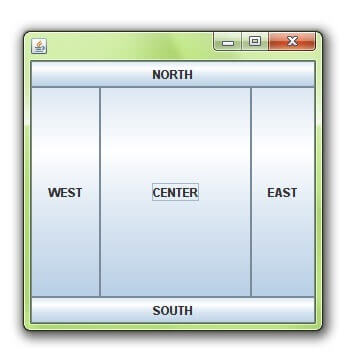
}

public static void main(String[] args) {

    new Border();

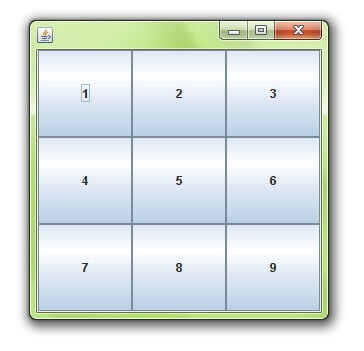
}

}



# GridLayout

The GridLayout is used to arrange the components in rectangular grid. One component is displayed in each rectangle.



import java.awt.\*;

import javax.swing.\*;

public class MyGridLayout{

JFrame f;

MyGridLayout(){

    f=new JFrame();

    JButton b1=new JButton("1");

    JButton b2=new JButton("2");

    JButton b3=new JButton("3");

    JButton b4=new JButton("4");

    JButton b5=new JButton("5");

        JButton b6=new JButton("6");

        JButton b7=new JButton("7");

    JButton b8=new JButton("8");

        JButton b9=new JButton("9");

    f.add(b1);f.add(b2);f.add(b3);f.add(b4);f.add(b5);

    f.add(b6);f.add(b7);f.add(b8);f.add(b9);

    f.setLayout(new GridLayout(3,3));

    //setting grid layout of 3 rows and 3 columns

    f.setSize(300,300);

    f.setVisible(true);

}

public static void main(String[] args) {

    new MyGridLayout();

}

}

# Java FlowLayout

The FlowLayout is used to arrange the components in a line, one after another (in a flow). It is the default layout of applet or panel.

### Fields of FlowLayout class

public static final int LEFT

public static final int RIGHT

public static final int CENTER

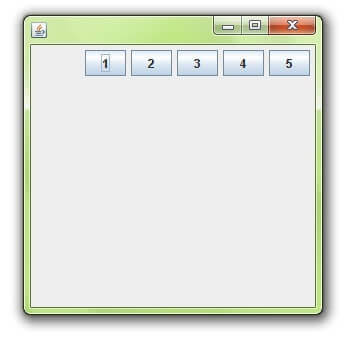
public static final int LEADING

public static final int TRAILING

### Constructors of FlowLayout class

1. FlowLayout(): creates a flow layout with centered alignment and a default 5 unit horizontal and vertical gap.
2. FlowLayout(int align): creates a flow layout with the given alignment and a default 5 unit horizontal and vertical gap.
3. FlowLayout(int align, int hgap, int vgap): creates a flow layout with the given alignment and the given horizontal and vertical gap.

### Example of FlowLayout class



import java.awt.\*;

import javax.swing.\*;

public class MyFlowLayout{

JFrame f;

MyFlowLayout(){

    f=new JFrame();

    JButton b1=new JButton("1");

    JButton b2=new JButton("2");

    JButton b3=new JButton("3");

    JButton b4=new JButton("4");

    JButton b5=new JButton("5");

    f.add(b1);f.add(b2);f.add(b3);f.add(b4);f.add(b5);

    f.setLayout(new FlowLayout(FlowLayout.RIGHT));

    //setting flow layout of right alignment

    f.setSize(300,300);

    f.setVisible(true);

}

public static void main(String[] args) {

    new MyFlowLayout();

}

}

# Java I/O

1) System.out: standard output stream

2) System.in: standard input stream

3) System.err: standard error stream

### OutputStream

Java application uses an output stream to write data to a destination; it may be a file, an array, peripheral device or socket

### InputStream

Java application uses an input stream to read data from a source; it may be a file, an array, peripheral device or socket.



## OutputStream class

|  |  |
| --- | --- |
| public void write(int)throws IOException | is used to write a byte to the current output stream. |
| 2) public void write(byte[])throws IOException | is used to write an array of byte to the current output stream. |
| 3) public void flush()throws IOException | flushes the current output stream. |
| 4) public void close()throws IOException | is used to close the current output stream. |

### OutputStream Hierarchy



## InputStream class

InputStream class is an abstract class. It is the superclass of all classes representing an input stream of bytes.

### Useful methods of InputStream

|  |  |
| --- | --- |
| Method | Description |
| 1) public abstract int read()throws IOException | reads the next byte of data from the input stream. It returns -1 at the end of the file. |
| 2) public int available()throws IOException | returns an estimate of the number of bytes that can be read from the current input stream. |
| 3) public void close()throws IOException | is used to close the current input stream. |

### InputStream Hierarchy



import java.io.FileOutputStream;

public class FileOutputStreamExample {

    public static void main(String args[]){

           try{

             FileOutputStream fout=new FileOutputStream("testout.txt");

             fout.write(65);

             fout.close();

             System.out.println("success...");

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

      }

}

**Output:**

Success...

The content of a text file testout.txt is set with the data A.

testout.txt

A

## Java FileOutputStream example 2: write string

import java.io.FileOutputStream;

public class FileOutputStreamExample {

    public static void main(String args[]){

           try{

             FileOutputStream fout=new FileOutputStream("a.txt");

             String s="Hello Rgukt”;

             byte b[]=s.getBytes();//converting string into byte array

             fout.write(b);

             fout.close();

             System.out.println("success...");

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

      }

}

Output:

Success...

a.txt

Hello Rgukt

# FileInputStream

## Java FileInputStream example 1: read single character

import java.io.FileInputStream;

public class DataStreamExample {

     public static void main(String args[]){

          try{

            FileInputStream fin=new FileInputStream("a.txt");

            int i=fin.read();

            System.out.print((char)i);

            fin.close();

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

         }

        }

**Output:**

H

## Java FileInputStream example 2: read all characters

package com.javatpoint;

import java.io.FileInputStream;

public class DataStreamExample {

     public static void main(String args[]){

          try{

            FileInputStream fin=new FileInputStream("a.txt");

            int i=0;

            while((i=fin.read())!=-1){

             System.out.print((char)i);

            }

            fin.close();

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

         }

        }

**Output:**

Hello Rgukt

## Java BufferedOutputStream class methods

|  |  |
| --- | --- |
| Method | Description |
| void write(int b) | It writes the specified byte to the buffered output stream. |
| void write(byte[] b, int off, int len) | It write the bytes from the specified byte-input stream into a specified byte [array](https://www.javatpoint.com/array-in-java), starting with the given offset |
| void flush() | It flushes the buffered output stream. |

import java.io.\*;

public class BufferedOutputStreamExample{

public static void main(String args[])throws Exception{

     FileOutputStream fout=new FileOutputStream("b.txt");

     BufferedOutputStream bout=new BufferedOutputStream(fout);

     String s="Welcome to Rgukt.";

     byte b[]=s.getBytes();

     bout.write(b);

     bout.flush();

     bout.close();

     fout.close();

     System.out.println("success");

}

}

**Output:**

Success

b.txt

Welcome to Rgukt.

import java.io.\*;

public class BufferedInputStreamExample{

 public static void main(String args[]){

  try{

    FileInputStream fin=new FileInputStream("b.txt");

    BufferedInputStream bin=new BufferedInputStream(fin);

    int i;

    while((i=bin.read())!=-1){

     System.out.print((char)i);

    }

    bin.close();

    fin.close();

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

 }

}

Here, we are assuming that you have following data in "b.txt" file:

Welcome to Rgukt.

Output:

Welcome to Rgukt.

# Java JDBC

|  |
| --- |
| JDBC Driver is a software component that enables java application to interact with the database. There are 4 types of JDBC drivers:   1. JDBC-ODBC bridge driver 2. Native-API driver (partially java driver) 3. Network Protocol driver (fully java driver) 4. Thin driver (fully java driver) |

### 1) JDBC-ODBC bridge driver

|  |
| --- |
| The JDBC-ODBC bridge driver uses ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls. This is now discouraged because of thin driver. |



### 2) Native-API driver

|  |
| --- |
| The Native API driver uses the client-side libraries of the database. The driver converts JDBC method calls into native calls of the database API. It is not written entirely in java. |



### 3) Network Protocol driver

The Network Protocol driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol. It is fully written in java.



### 4) Thin driver

|  |
| --- |
| The thin driver converts JDBC calls directly into the vendor-specific database protocol. That is why it is known as thin driver. It is fully written in Java language. |



# Java Database Connectivity with MySQL

To connect Java application with the MySQL database, we need to follow 5 following steps.

In this example we are using MySql as the database. So we need to know following informations for the mysql database:

1. Driver class: com.mysql.jdbc.Driver.
2. Connection URL: jdbc:mysql://localhost:3306/sonoo
3. Username: The default username for the mysql database is root.
4. Password: It is the password given by the user at the time of installing the mysql database. In this example, we are going to use root as the password.

**Inserting:**

import java.sql.\*;

class Insert{

public static void main(String args[]){

try{

class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection( "jdbc:mysql://localhost:3306/latha","root","Arjun@5808");

System.out.println("database connected");

PreparedStatement stmt=con.prepareStatement("insert into student values(?,?)");

stmt.setInt(1,160313);

stmt.setString(2,"Amulya");

int i=stmt.executeUpdate();

System.out.println(i+" records inserted");

con.close();

}catch(Exception e){ System.out.println("not connected with database");}

}

}

**Retrieving:**

import java.sql.\*;

public class Retriving {

public static void main(String[] args) {

try{

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection( "jdbc:mysql://localhost:3306/latha","root","Arjun@5808");

System.out.println("database connected");

Statement stmt=con.createStatement();

ResultSet rs=stmt.executeQuery("select \* from student");

while(rs.next()) {

System.out.println(rs.getInt(1)+" "+rs.getString(2));

}

con.close();

}catch(Exception e){ System.out.println("not connected with database");

}}

}