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.

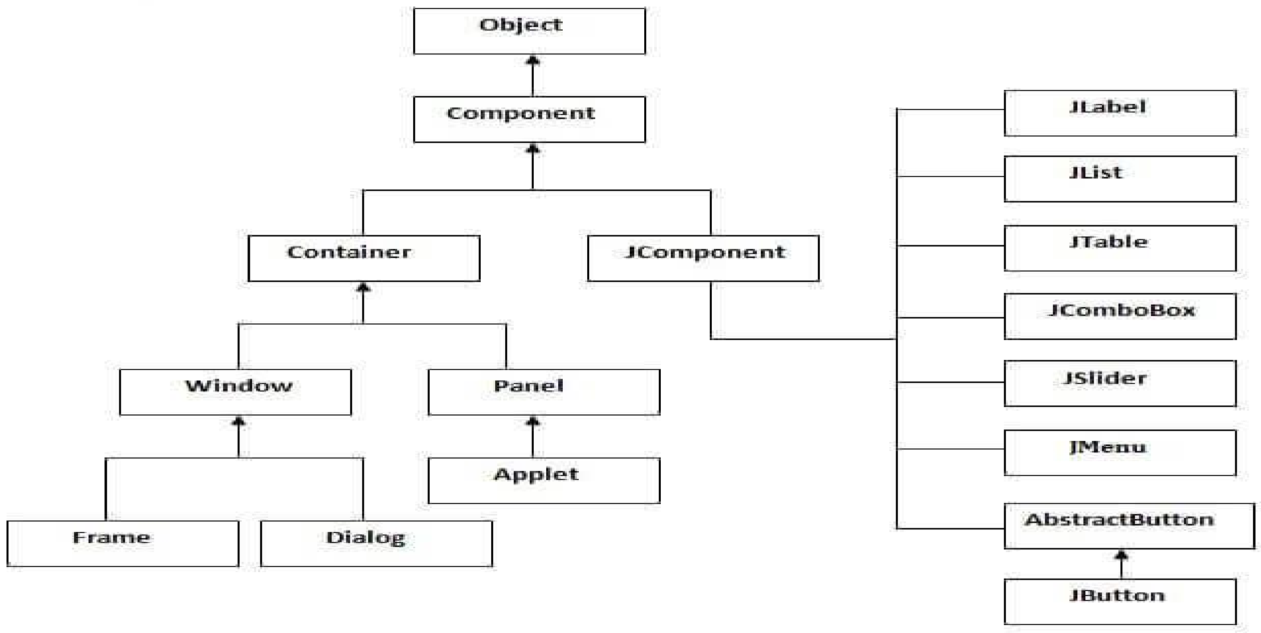
Java Swing Examples

There are two ways to create a frame:

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

Hierarchy of Java Swing classes

The hierarchy of java swing API is given below.



Java JButton

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

JButton class declaration

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

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

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.

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

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

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.

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

```
1.  public class JTextField extends JTextComponent implements SwingConstants
```

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

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

```
1.  public class JTextArea extends JTextComponent
```

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

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

```
1.  public class JPasswordField extends JTextField
```

Java JCheckBox

The JCheckBox class is used to create a checkbox. It is used to turn an option on (true) or off (false). Clicking on a CheckBox changes its state from "on" to "off" or from "off" to "on ".It inherits JToggleButton class.

JCheckBox class declaration

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

```
1.  public class JCheckBox extends JToggleButton implements Accessible
```

Java JRadioButton

The JRadioButton class is used to create a radio button. It is used to choose one option from multiple options. It is widely used in exam systems or quiz.

It should be added in ButtonGroup to select one radio button only.

JRadioButton class declaration

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

1. **public class** JRadioButton **extends** JToggleButton **implements** Accessible

Java JComboBox

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

JComboBox class declaration

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

public class JComboBox **extends** JComponent **implements** ItemSelectable, ListDataListener, ActionListener, Acce
ssible

Java JTable

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

JTable class declaration

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

Java JList

The object of JList class represents a list of text items. The list of text items can be set up so that the user can choose either one item or multiple items. It inherits JComponent class.

JList class declaration

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

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

Java JOptionPane

The JOptionPane class is used to provide standard dialog boxes such as message dialog box, confirm dialog box and input dialog box. These dialog boxes are used to display information or get input from the user. The JOptionPane class inherits JComponent class.

JOptionPane class declaration

1. `public class JOptionPane extends JComponent implements Accessible`

Java JScrollBar

The object of JScrollbar class is used to add horizontal and vertical scrollbar. It is an implementation of a scrollbar. It inherits JComponent class.

JScrollBar class declaration

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

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

Java JMenuBar, JMenu and JMenuItem

The JMenuBar class is used to display menubar on the window or frame. It may have several menus.

The object of JMenu class is a pull down menu component which is displayed from the menu bar. It inherits the JMenuItem class.

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

JMenuBar class declaration

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

JMenu class declaration

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

JMenuItem class declaration

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

Java JPopupMenu

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

JPopupMenu class declaration

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

```
1.      public class JPopupMenu extends JComponent implements Accessible, MenuElement
```

Java JCheckBoxMenuItem

JCheckBoxMenuItem class represents checkbox which can be included on a menu . A CheckBoxMenuItem can have text or a graphic icon or both, associated with it. MenuItem can be selected or deselected. MenuItems can be configured and controlled by actions.

Java JSeparator

The object of JSeparator class is used to provide a general purpose component for implementing divider lines. It is used to draw a line to separate widgets in a Layout. It inherits JComponent class.

Java JProgressBar

The JProgressBar class is used to display the progress of the task. It inherits JComponent class.

JProgressBar class declaration

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

```
1.      public class JProgressBar extends JComponent implements SwingConstants, Accessible
```

Java JTree

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

JTree class declaration

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

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

Java JColorChooser

The JColorChooser class is used to create a color chooser dialog box so that user can select any color. It inherits JComponent class.

JColorChooser class declaration

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

1. public class JColorChooser extends JComponent implements Accessible

Java JTabbedPane

The JTabbedPane class is used to switch between a group of components by clicking on a tab with a given title or icon. It inherits JComponent class.

JTabbedPane class declaration

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

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

Java JSlider

The Java JSlider class is used to create the slider. By using JSlider, a user can select a value from a specific range.

Java JSpinner

The object of JSpinner class is a single line input field that allows the user to select a number or an object value from an ordered sequence.

JSpinner class declaration

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

public class JSpinner extends JComponent implements Accessible

Java JDialog

The JDialog control represents a top level window with a border and a title used to take some form of input from the user. It inherits the Dialog class.

Unlike JFrame, it doesn't have maximize and minimize buttons.

JDialog class declaration

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

1. public class JDialog extends Dialog implements WindowConstants, Accessible, RootPaneContainer

Java JPanel

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

It doesn't have title bar.

JPanel class declaration

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

Java JFileChooser

The object of JFileChooser class represents a dialog window from which the user can select file. It inherits JComponent class.

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

Java JToggleButton

JToggleButton is used to create toggle button, it is two-states button to switch on or off.

Java JToolBar

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

Java JViewport

The JViewport class is used to implement scrolling. JViewport is designed to support both logical scrolling and pixel-based scrolling. The viewport's child, called the view, is scrolled by calling the JViewport.setViewPosition() method.

Java JFrame

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

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

Java JComponent

The JComponent class is the base class of all Swing components except top-level containers. Swing components whose names begin with "J" are descendants of the JComponent class. For example, JButton, JScrollPane, JPanel, JTable etc. But, JFrame and JDialog don't inherit JComponent class because they are the child of top-level containers.

The JComponent class extends the Container class which itself extends Component. The Container class has support for adding components to the container.

Java JLayeredPane

The JLayeredPane class is used to add depth to swing container. It is used to provide a third dimension for positioning component and divide the depth-range into several different layers.

```
1.      public class JLayeredPane extends JComponent implements Accessible
```

Java JDesktopPane

The JDesktopPane class, can be used to create "multi-document" applications. A multi-document application can have many windows included in it. We do it by making the contentPane in the main window as an instance of the JDesktopPane class or a subclass. Internal windows add instances of JInternalFrame to the JdesktopPane instance. The internal windows are the instances of JInternalFrame or its subclasses.

Java JEditorPane

JEditorPane class is used to create a simple text editor window. This class has setContentType() and setText() methods.

setContentType("text/plain"): This method is used to set the content type to be plain text.

setText(text): This method is used to set the initial text content.

Java JScrollPane

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

Java JSplitPane

JSplitPane is used to divide two components. The two components are divided based on the look and feel implementation, and they can be resized by the user. If the minimum size of the two components is greater than the size of the split pane, the divider will not allow you to resize it.

The two components in a split pane can be aligned left to right using JSplitPane.HORIZONTAL_SPLIT, or top to bottom using JSplitPane.VERTICAL_SPLIT. When the user is resizing the components the minimum size of the components is used to determine the maximum/minimum position the components can be set to.

Java JTextPane

JTextPane is a subclass of JEditorPane class. JTextPane is used for styled document with embedded images and components. It is text component that can be marked up with attributes that are represented graphically. JTextPane uses a DefaultStyledDocument as its default model.

Java JTextPane

JTextPane is a subclass of JEditorPane class. JTextPane is used for styled document with embedded images and components. It is text component that can be marked up with attributes that are represented graphically. JTextPane uses a DefaultStyledDocument as its default model.

How to use ToolTip in Java Swing

You can create a tool tip for any JComponent with **setToolTipText()** method. This method is used to set up a tool tip for the component.

For example, to add tool tip to PasswordField, you need to add only one line of code:

```
1.      field.setToolTipText("Enter your Password");
```

How to change TitleBar icon in Java AWT and Swing

The setIconImage() method of Frame class is used to change the icon of Frame or Window. It changes the icon which is displayed at the left side of Frame or Window.

The Toolkit class is used to get instance of Image class in AWT and Swing.

Toolkit class is the abstract super class of every implementation in the Abstract Window Toolkit(AWT). Subclasses of Toolkit are used to bind various components. It inherits Object class.

How to make an executable jar file in Java

The **jar (Java Archive)** tool of JDK provides the facility to create the executable jar file. An executable jar file calls the main method of the class if you double click it.

To create the executable jar file, you need to create **.mf file**, also known as manifest file.

Introduction to JDBC (Java Database Connectivity)

*JDBC is an API(Application programming interface) used in java programming to interact with databases. The classes and interfaces of JDBC allow the application to send requests made by users to the specified database.*JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database. There are four types of JDBC drivers:

- JDBC-ODBC Bridge Driver,
- Native Driver,
- Network Protocol Driver, and
- Thin Driver

What is API

API (Application programming interface) is a document that contains a description of all the features of a product or software. It represents classes and interfaces that software programs can follow to communicate with each other. An API can be created for applications, libraries, operating systems, etc.

Purpose of JDBC

Enterprise applications created using the JAVA EE technology need to interact with databases to store application-specific information. So, interacting with a database requires efficient database connectivity, which can be achieved by using the ODBC(Open database connectivity) driver. This driver is used with JDBC to interact or communicate with various kinds of databases such as Oracle, MS Access, Mysql, and SQL server database.

Components of JDBC

There are generally four main components of JDBC through which it can interact with a database. They are as mentioned below:

1. JDBC API: It provides various methods and interfaces for easy communication with the database. It provides two packages as follows, which contain the java SE and Java EE platforms to exhibit WORA(write once run anywhere) capabilities.

```
java.sql.*;
```

It also provides a standard to connect a database to a client application.

2. JDBC Driver manager: It loads a database-specific driver in an application to establish a connection with a database. It is used to make a database-specific call to the database to process the user request.

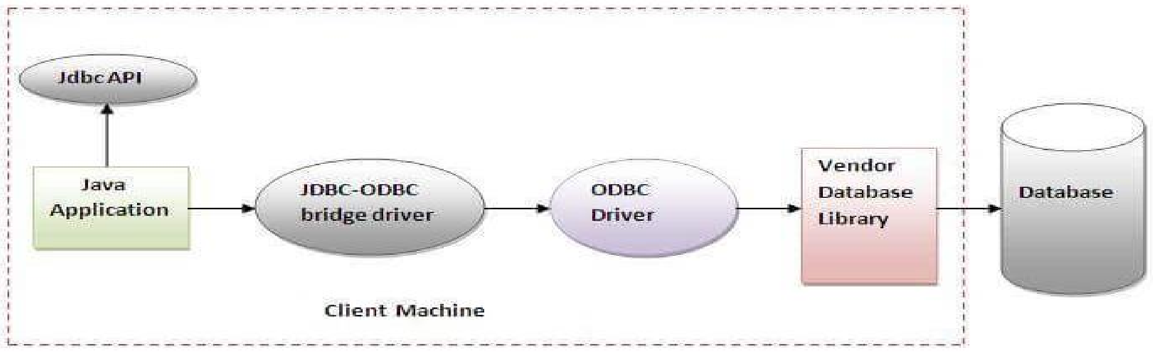
3. JDBC Test suite: It is used to test the operation(such as insertion, deletion, updation) being performed by JDBC Drivers.

4. JDBC-ODBC Bridge Drivers: It connects database drivers to the database. This bridge translates the JDBC method call to the ODBC function call. It makes use of the **sun.jdbc.odbc** package which includes a native library to access ODBC characteristics.

JDBC 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.



In Java 8, the JDBC-ODBC Bridge has been removed.

Oracle does not support the JDBC-ODBC Bridge from Java 8. Oracle recommends that you use JDBC drivers provided by the vendor of your database instead of the JDBC-ODBC Bridge.

Advantages:

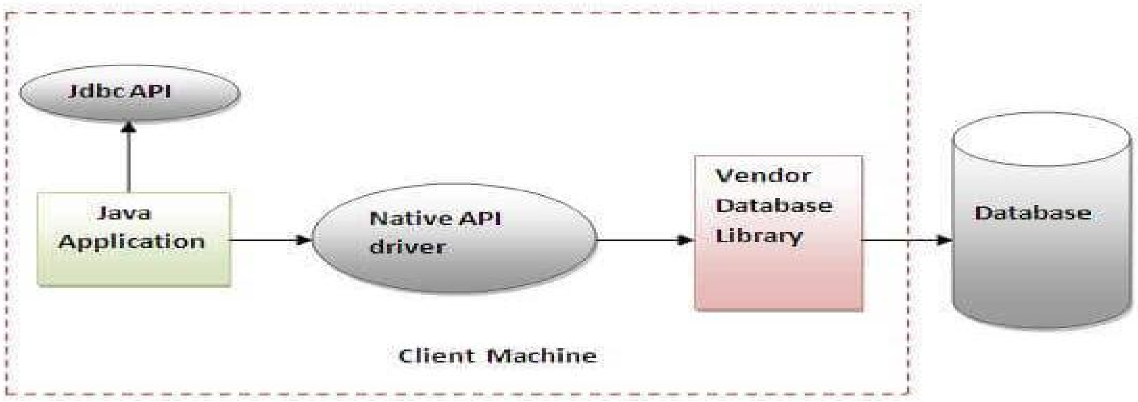
- easy to use.
- can be easily connected to any database.

Disadvantages:

- Performance degraded because JDBC method call is converted into the ODBC function calls.
- The ODBC driver needs to be installed on the client machine.

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.



Advantage:

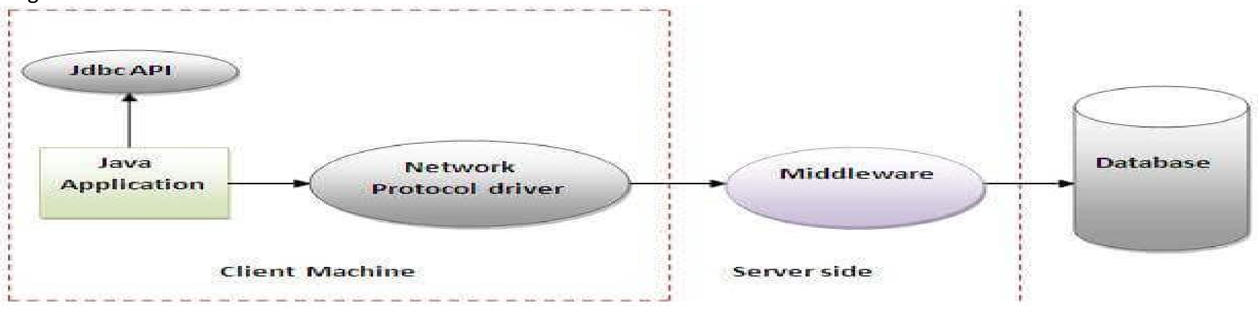
- performance upgraded than JDBC-ODBC bridge driver.

Disadvantage:

- The Native driver needs to be installed on the each client machine.
- The Vendor client library needs to be installed on client machine.

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.



Advantage:

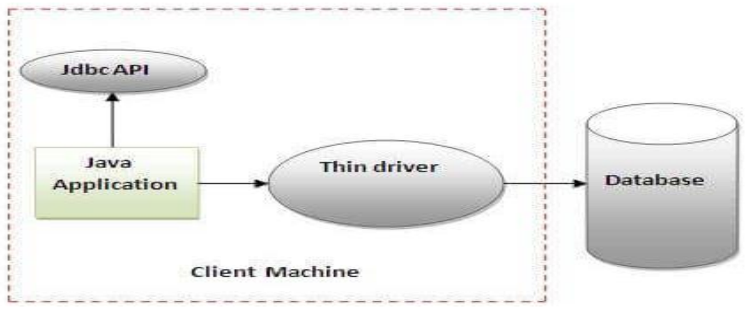
- No client side library is required because of application server that can perform many tasks like auditing, load balancing, logging etc.

Disadvantages:

- Network support is required on client machine.
- Requires database-specific coding to be done in the middle tier.
- Maintenance of Network Protocol driver becomes costly because it requires database-specific coding to be done in the middle tier.

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.



Advantage:

- Better performance than all other drivers.
- No software is required at client side or server side.

Disadvantage:

- Drivers depend on the Database.

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:** The driver class for the mysql database is **com.mysql.jdbc.Driver**.
2. **Connection URL:** The connection URL for the mysql database is **jdbc:mysql://localhost:3306/sonoo** where jdbc is the API, mysql is the database, localhost is the server name on which mysql is running, we may also use IP address, 3306 is the port number and sonoo is the database name. We may use any database, in such case, we need to replace the sonoo with our database name.
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.

Let's first create a table in the mysql database, but before creating table, we need to create database first.

```
create database sonoo;
use sonoo;
create table emp(id int(10),name varchar(40),age int(3));
```

Example to Connect Java Application with mysql database

In this example, sonoo is the database name, root is the username and password both.

```
import java.sql.*;
class MysqlCon{
public static void main(String args[]){
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/sonoo","root","root");
//here sonoo is database name, root is username and password
Statement stmt=con.createStatement();
ResultSet rs=stmt.executeQuery("select * from emp");
while(rs.next())
System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getString(3));
con.close();
}catch(Exception e){ System.out.println(e);}
}
}
```

The above example will fetch all the records of emp table.

To connect java application with the mysql database, **mysqlconnector.jar** file is required to be loaded.