UNIT 3
JAVA DATABASE CONNECTIVITY (JDBC)

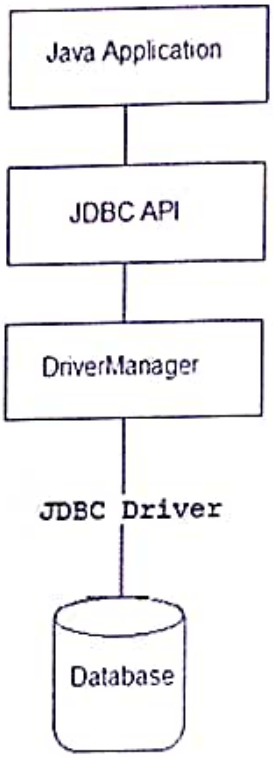
JAVA DATABASE CONNECTIVITY

Java Database Connectivity(JDBC) is an **Application Programming Interface(API)** used to connect Java application with Database. JDBC is used to interact with various type of Database such as Oracle, MS Access, My SQL and SQL Server. JDBC can also be defined as the platform-independent interface between a relational database and Java programming. It allows java program to execute SQL statement and retrieve result from database.

Software Required: JAVA Editors, JDK, Appletviewer Tool, JAVA enabled Web Browser, Eclipse, JDBC Driver (mysql-connector-java-5.1.18-bin)

Package: java.applet.*, java.awt.*, java.awt.event.*,java.sql.*

JDBC Architecture



JDBC Driver

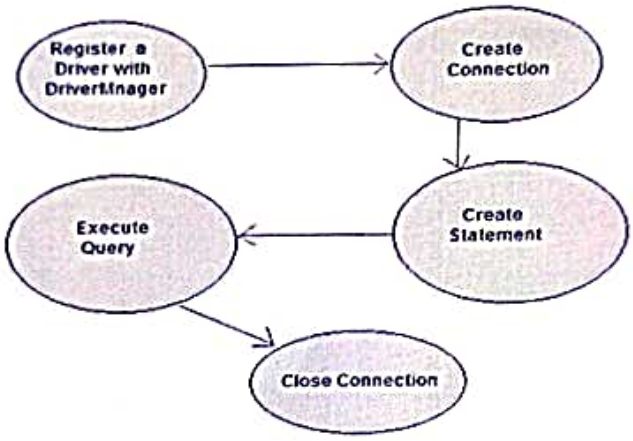
JDBC Driver is required to process SQL requests and generate result. The following are the different types of driver available in JDBC.

- Type-1 Driver or JDBC-ODBC bridge
- Type-2 Driver or Native API Partly Java Driver
- Type-3 Driver or Network Protocol Driver
- Type-4 Driver or Thin Driver

Steps to connect a Java Application to Database

The following 5 steps are the basic steps involve in connecting a Java application with Database using JDBC.

1. Register the Driver
2. Create a Connection
3. Create SQL Statement
4. Execute SQL Statement
5. Closing the connection



1. Register the Driver

Class.forName() is used to load the driver class explicitly.

Example:

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

2. Create a Connection

getConnection() method of DriverManager class is used to create a connection

Syntax

getConnection(String url)

getConnection(String url, String username, String password)

getConnection(String url, Properties info)

Example:

Connection con = DriverManager.getConnection ("jdbc:mysql://localhost:3306/dbname ", "username", "password");

3. Create SQL Statement

createStatement() method is invoked on current Connection object to create a SQL Statement.

Syntax

public Statement createStatement() throws SQLException

Example:

Statement s=con.createStatement();

4. Execute SQL Statement

executeQuery() method of Statement interface is used to execute SQL statements.

Syntax

public ResultSet executeQuery(String query) throws SQLException

Example

```
ResultSet rs=s.executeQuery("select * from user");
 while(rs.next())
 {
  System.out.println(rs.getString(1)+" "+rs.getString(2));
}
```

5. Closing the connection

After executing SQL statement you need to close the connection and release the session. The `close()` method of **Connection** interface is used to close the connection.

Syntax

public void close() throws SQLException

Example

con.close();

Practical No. 14

Develop a database application that uses any JDBC driver.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLExcaption;
import java.sql.Statement;

Public Class Practical14
{
    Public Static void main(String args[])
    {
        Connection Con = null;
        Statement Stmt = null;
        ResultSet rs = null;
        try
        {
            Class.forName("com.mysql.jdbc.
                        Driver");
            Con.DriverManager.getConnection
                ("jdbc.mysql://localhost:3306/
                Student_Info", "root", "");

System.out.println("Connection Successful");
Stmt = Con.createStatement();
System.out.println("Reading from
                table...");
String Sql = "Select * from Student_
                details";
rs = Stmt.executeQuery(Sql);
while(rs.next())
{
```

```
System.out.println("Id: "+rs.getInt(1)+
    "Name: "+rs.getString(2)+"Age: "
    +rs.getString(3)+"\n");
}
}
Catch (ClassNotFoundException e)
{
    System.out.println(e);
}
Catch (SQLException e)
{
  System.out.println(e);
}
finally
{
    try
    {
      if(rs!=null)
        rs.close();
      if(Stmt!=null)
        Stmt.close();
      if(con!=null);
        con.close();
    }
    Catch (SQLException e)
    {
        System.out.println(e);
    }
}
}
}
```

Observation :-

By implementing java.sql Packages and Establishing Connection with the database we can read all the information from the database and display them.

* Output :-

Connection Successfull

Reading from table...

ID: 1 Name : XYZ Age : 22

Practical No. 15

Develop a UI that performs the following SQL operations: 1) Insert 2)Delete 3)Update.

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;

//<Applet code="Practical15.class" width="500"
        height="500" ></APPLET>

Public class Practical15 extends Applet
        implements ActionListener
{
    Label l1, l2, l3;
    TextField id, name, age;
    Button ins, up, del;

    Connection Con;
    Statement Stmt;

    Public void init()
    {
        l1 = new Lable("Id:");
        l2 = new lable("Name:");
        l3 = new label("Age:");

        id = new TextField();
        name = new TextField();
        age = new TextField();

        ins = new Button("Insert");
        up = new Button("Update");
        del = new Button("Delete");
```

```
add(l7);
add(id);

add(l2);
add(name);

add(l3);
add(age);

add(ins);
add(up);
add(del);

ins.addActionListener(this);
up.addActionListener(this);
del.addActionListener(this);

try {
    Class.forName("com.mysql.jdbc.
                   Driver");
con = DriverManager.getConnection("jdbc:
      mysql://localhost:3306/Student_
      info","root","");

Stmt = con.createStatement();
}
catch(ClassNotFoundException e)
{
    e.printStackTrace();
}
catch(SQLException se)
{
    se.printStackTrace();
}
```

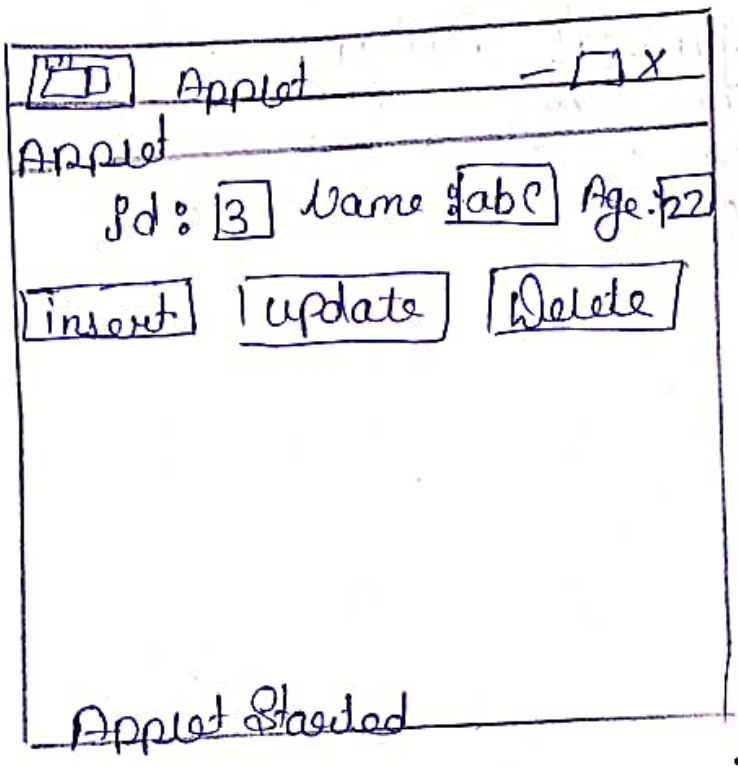
```
Public void actionPerformed(ActionEvent ae)
{
    String s = ae.getActionCommand();
    String query = null;
    String sid = id.getText();
    int a = Integer.valueOf(sid);
    String sname = name.getText();
    String sage = age.getText();
    if(s.equals("Insert"))
    {
        query = "Insert INTO Student_details
        Values("+sid+" "+sname+", "+sage+")";
        System.out.println("Inserted Successfully..");
    }
    else if(s.equals("Update"))
    {
        query = "update Student_details SET
        name = '"+sname+"' WHERE id="+a";
        System.out.println("Updated Successfully");
    }
    else if(s.equals("Delete"))
    {
        query = "DELETE from Student_details
        WHERE id = "+a";
        System.out.println("deleted Successfully");
    }
    try{
        Stmt.executeUpdate(query);
    }
    catch(SQLException e)
    { e.printStackTrace();
    }
}
```

```
public void destroy ()
{
    try {
        if (Stmt != null)
            Stmt.close();
        if (con != null)
            con.close();
    }
    Catch (SQLException e) {
        e.printStackTrace();
    }
}
```

* Observation :-

By implementation ActionListener interface, and adding label and textbox, we can add details and through click on insert button information is inserted, click on update button data will be updated and click on delete button data will be deleted.

Output :-



Inserted Successfully.

Practical No. 16

Write a program to present a set of choice for user to select a product & display the price of product.

```
import java.applet.Applet;
import java.awt.Choice;
import java.awt.Label;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

Public class Practical16 extends Applet
        implements ItemListener {
    Choice c;
    Label ans;
    Statement stmt;
    Connection con;
    ResultSet rs;

    @override
    public void init() {
        c=new Choice();
        add(c);
        ans=new Label();
        add(ans);

        c.addItemListener(this);
        try{
            Class.forName("com.mysql.jdbc.
                                Driver");
            con=DriverManager.getConnection("jdbc:
            mysql://localhost:3306/product_info",
                                "root","");
```

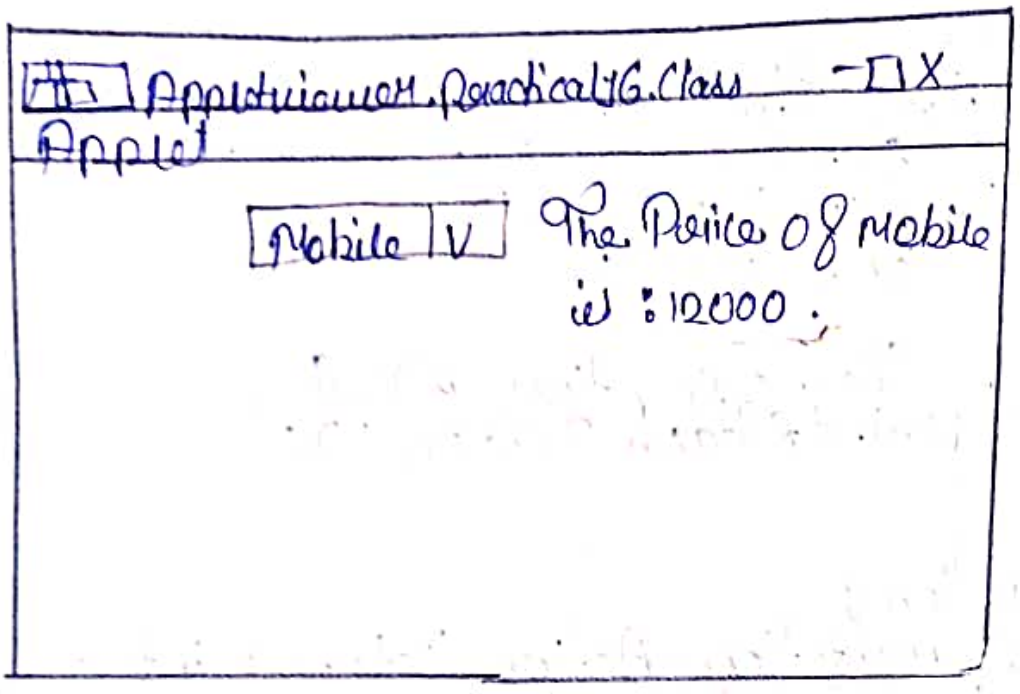
```
Stmt = con.CreateStatement();
rs = Stmt.executeQuery("SELECT
     name from product_details");
while(rs.next())
{
  c.add(rs.getString());
}
}catch(ClassNotFoundException e){
     e.printStackTrace();
}catch(SQLException e){
  e.printStackTrace();
}
}
@Override
Public void itemStateChanged(ItemEvent e)
{
  try{
     rs = Stmt.executeQuery("SELECT price,
     name from product_details WHERE
     name='"+c.getSelectedItem()+"'");
while(rs.next())
{
  ans.setText("The price of "+rs.
  getString(2)+" is :"+rs.getString(1));
}
}catch(SQLException e1){
  e1.printStackTrace();
}
}
Public void destroy()
{try{
     rs.close();
     Stmt.close();
```

```
        con.close();
    }
    catch (SQLException e)
    {
        e.printStackTrace();
    }
}
}
```

* Observation :-

By implementing ItemListener interface, through adding Label and Choice and fetching name of product in choice through database. On Selecting, we can get price of selected product in label.

Output :


Applet Viewer: Practical16.Class
Applet
Mobile
V
The Price of Mobile is : 12000.