**i. Class.forName ():**  
This method loads the driver class files into memory during the run time. You don't have to use new objects or create objects. In the following example uses Class.forName () to load the Oracle driver as  follows:  
The syntax for this is Syntax: Class.forName(“oracle.jdbc.driver.OracleDriver”)

**ii.  DriverManager.registerDriver ()**

DriverManager is a Java built-in class with static member registers. Here, we have called the constructor of the driver class at the compile time. In the following example uses DriverManager.registerDriver () to register the Oracle driver. Let us have a look at it.

Syntax: DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver())

**3. Now, we have to register the driver in DriverManager.**

**4. Next, form a Connection using a Connection class object**

After loading the driver, we can establish a connection in the following ways

Connection con = DriverManager.getConnection(url,user,password)

It requires the following parameters:

1. user: The user name that can access the SQL prompt.
2. Password: The password you can use to access the SQL prompt.
3. Con: This is a reference to the connection interface.
4. URL: Uniform Resource Locator

We have looked at all the parameters required to maintain the connection. Let us see an example to understand it better:

**Example :**

String url = “ jdbc:oracle:thin:@localhost:5421:xe”

**5. We now need to create a statement.**

Since we have established the connection, we can work with the database. The CallableStatement, JDBCStatement and PreparedStatement interfaces define the methods we can use to send SQL commands and receive data from the database. The usage of the  JDBC statement is as follows:

Statement st = con.createStatement();

**6. Execution of the query**

The query here is a SQL query. Now you know that you can execute multiple types of queries, like queries to update/insert a table in the database. A query to get the data. We use the executeQuery () method of the Statement interface to execute a query to get a value from the database. This method returns an object in the  ResultSet. We can use the ResultSet object to extract the records from the table. We use the executeUpdate (SQL query) method to execute update/insert queries.

**7. Closing the connection**

In the end, we have sent the data to the specified location and are nearing the task's completion. When you close the connection, the Statement and ResultSet objects close automatically. We use the close () method of the Connection interface to close the connection.

It will be as follows:

con.close ();

**Implementation**

// A Java Program for Establishing Connection in JDBC

// Importing the database

import java.sql.\*;

// Importing the required classes

import java.util.\*;

// Main class of Java

class Main

{

   // The Main driver method

Public static void main(String a[])

   {

       // Creating a connection using Oracle DB

       String url = "jdbc:oracle:thin:@localhost:5241:xe";

       // Username and password to access DB

       // Custom initialization

       String user = "superuser";

       String pass = "54321";

       // Entering data

       Scanner k = new Scanner(System.in);

       System.out.println("enter employee\_name");

       String employee\_name = k.next();

       System.out.println("enter employee\_id");

       int employee\_id = k.nextInt();

       System.out.println("enter department");

       String department = k.next();

       // Inserting data using SQL query

       String sql = "insert into employee1 values('" + employee\_name + "'," + employee\_id + ",'" + department + "')";

       // Connection class object

       Connection connectionn = null;

       // Try block to check for exceptions

       try

       {

           // Registering drivers

           DriverManager.registerDriver(

               new oracle.jdbc.OracleDriver());

           // Reference to connection interface

           connectionn = DriverManager.getConnection(url, user,

                                                     pass);

           // Creating a statement

           Statement st = connectionn.createStatement();

           // Executing query

           int m = st.executeUpdate(sql);

           if (m == 1)

               System.out.println(

                   "inserted successfully : " + sql);

           else

               System.out.println("insertion failed");

           // Closing the connections

           connectionn.close();

       }

       // Catch block to handle exceptions

       catch (Exception ex)

       {

           // Display message when exceptions occurs

           System.err.println(ex);

       }

   }

}

Output:

Enter employee\_name

Ninja

Enter employee\_id

6523

Enter department

12A

inserted successfully:  insert into employee1 values(‘Ninja',6523, '12A')

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.sql.Statement;

public class JDBCExample {

static final String DB\_URL = "jdbc: mysql://localhost/TUTORIALSPOINT";

static final String USER = "guest";

static final String PASS = "guest123";

public static void main(String[] args) {

// Open a connection

try(Connection conn = DriverManager.getConnection(DB\_URL, USER, PASS);

Statement stmt = conn.createStatement();

) {

// Execute a query

System.out.println("Inserting records into the table...");

String sql = "INSERT INTO Registration VALUES (100, 'Zara', 'Ali', 18)";

stmt.executeUpdate(sql);

sql = "INSERT INTO Registration VALUES (101, 'Mahnaz', 'Fatma', 25)";

stmt.executeUpdate(sql);

sql = "INSERT INTO Registration VALUES (102, 'Zaid', 'Khan', 30)";

stmt.executeUpdate(sql);

sql = "INSERT INTO Registration VALUES(103, 'Sumit', 'Mittal', 28)";

stmt.executeUpdate(sql);

System.out.println("Inserted records into the table...");

} catch (SQLException e) {

e.printStackTrace();

}

}

}

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class JDBCExample {

static final String DB\_URL = "jdbc:mysql://localhost/TUTORIALSPOINT";

static final String USER = "guest";

static final String PASS = "guest123";

static final String QUERY = "SELECT id, first, last, age FROM Registration";

public static void main(String[] args) {

// Open a connection

try(Connection conn = DriverManager.getConnection(DB\_URL, USER, PASS);

Statement stmt = conn.createStatement();

) {

String sql = "DELETE FROM Registration " +

"WHERE id = 101";

stmt.executeUpdate(sql);

ResultSet rs = stmt.executeQuery(QUERY);

while(rs.next()){

//Display values

System.out.print("ID: " + rs.getInt("id"));

System.out.print(", Age: " + rs.getInt("age"));

System.out.print(", First: " + rs.getString("first"));

System.out.println(", Last: " + rs.getString("last"));

}

rs.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}