JDBC IMPLEMETATION STEPS

SEVEN STEPS TO JDBC

1. Importing the java.sql.* package
2. Loading the driver
3. Establishing the DB connection
4. Creating a statement
5. Executing a statement
6. Retrieving the results
7. Closing the connection & statement

1. Importing the java.sql.* package

- The package **java.sql.*** contains the JDBC base API classes.

Ex.

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

2. Loading the driver

- The **forName()** method of class, 'class' is used to **register the driver class.**
- It is a static method of class name 'class'
- This method is used to dynamically load the driver class.
- It takes only one argument. The argument must be a driver class.
- Return type : **void**

Ex.

String driver="sun.jdbc.odbc.JdbcOdbcDriver"; //JDBC Driver

Class.forName(driver**);**

3. Establishing the DB connection

- Next step is to connect the database at a specified URL.
- Eveny database is identified by a URL.
- This is done by calling the **getConnection() method** of DriverManager class.
- The **getConnection()** method takes only one argument
- JDBC connections are specified by a Uniform Resource Locator (URL), which has the format

Syntax

jdbc:<subprotocol>:<dsn name>

where,

subprotocol ← the kind of database connectivity

dsn ← Data Source Name (contains information about the database)

Ex.

```
String url="jdbc:odbc:college";     // db URL
Connection con=DriverManager.getConnection(url);
```

4. Creating a statement

- The **createStament()** method of the Connection class is used to create a Statement object, for **executing SQL statements**

- After making connection to the database, we are ready to create SQL statements
- This can be done by using the methods of statement object

Ex.

Statement st=**con.createStatement(); // runnable state**

5 & 6. Executing a Query Statement & Retrieving the Results

- The **executeQuery()** method of Statement interface is used to execute queries to the database.
- This method returns the **object of ResultSet** that can be used to get all the records of a table.

Ex.

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

6. Retrieving the results

- The results must be processed after executing the SQL statements
- Some statements (**insert, delete, update)** will return **only an integer value** containing the number of rows affected by that statement
- **SQL query (e.g: select)** is used to return the **result set** containing the results of the query
- The result set is made up of columns / rows.

6.1 Important Methods

next()

- It is an instance method of the ResultSet class
- Just check whether the **next row is available or not**
- It returns **current row** if next row is present else it returns null value
- Return type: **boolean**

7. Closing the connection & statement 

- The connection object, resultset & all the Statement objects contain a **close()** method which should be called to ensure that the database system frees all the associated resources properly.
- By closing connection object statment and ResultSet will be closed automatically.

close()

- The **close() method** of Connection interface is used to close the connection
- Return type : **void**

Ex.

```
con.close();
```