Java Database Connectivity with 5 Steps

1. [5 Steps to connect to the database in java](https://www.javatpoint.com/steps-to-connect-to-the-database-in-java)
   1. [Register the driver class](https://www.javatpoint.com/steps-to-connect-to-the-database-in-java#step1)
   2. [Create the connection object](https://www.javatpoint.com/steps-to-connect-to-the-database-in-java#step2)
   3. [Create the Statement object](https://www.javatpoint.com/steps-to-connect-to-the-database-in-java#step3)
   4. [Execute the query](https://www.javatpoint.com/steps-to-connect-to-the-database-in-java#step4)
   5. [Close the connection object](https://www.javatpoint.com/steps-to-connect-to-the-database-in-java#step5)

|  |
| --- |
|  |

1) Register the driver class

|  |
| --- |
| The **forName()** method of Class class is used to register the driver class. This method is used to dynamically load the driver class. |

Syntax of forName() method

1. **public** **static** **void** forName(String className)**throws** ClassNotFoundException

Note: Since JDBC 4.0, explicitly registering the driver is optional. We just need to put vender's Jar in the classpath, and then JDBC driver manager can detect and load the driver automatically.

Example to register the OracleDriver class

Here, Java program is loading oracle driver to esteblish database connection.

1. Class.forName("oracle.jdbc.driver.OracleDriver");

2) Create the connection object

|  |
| --- |
| The **getConnection()** method of DriverManager class is used to establish connection with the database. |

Syntax of getConnection() method

1. 1) **public** **static** Connection getConnection(String url)**throws** SQLException
2. 2) **public** **static** Connection getConnection(String url,String name,String password)
3. **throws** SQLException

Example to establish connection with the Oracle database

1. Connection con=DriverManager.getConnection(
2. "jdbc:oracle:thin:@localhost:1521:xe","system","password");

3) Create the Statement object

|  |
| --- |
| The createStatement() method of Connection interface is used to create statement. The object of statement is responsible to execute queries with the database. |

Syntax of createStatement() method

1. **public** Statement createStatement()**throws** SQLException

Example to create the statement object

1. Statement stmt=con.createStatement();

4) Execute the query

|  |
| --- |
| 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. |

Syntax of executeQuery() method

1. **public** ResultSet executeQuery(String sql)**throws** SQLException

Example to execute query

1. ResultSet rs=stmt.executeQuery("select \* from emp");
3. **while**(rs.next()){
4. System.out.println(rs.getInt(1)+" "+rs.getString(2));
5. }

5) Close the connection object

|  |
| --- |
| By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection. |

Syntax of close() method

1. **public** **void** close()**throws** SQLException

Example to close connection

1. con.close();