JDBC

CAMERINFOLKS

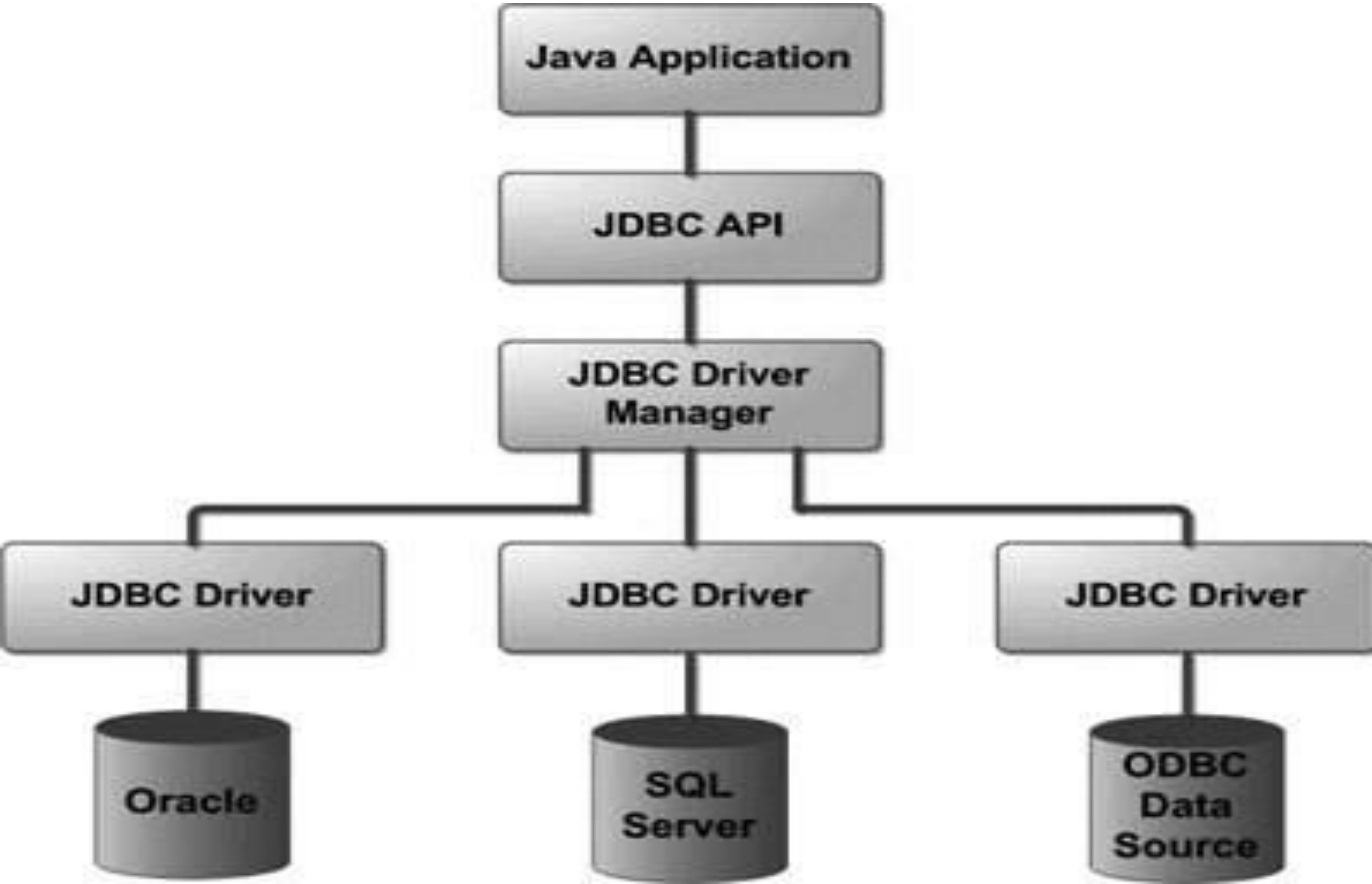
What is JDBC?

- JDBC stands for **J**ava **D**ata**b**ase **C**onnectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases.

Why use JDBC?

- Before JDBC, ODBC API was the database API to connect and execute query with the database. But, ODBC API uses ODBC driver which is written in C language (i.e. platform dependent and unsecured). That is why Java has defined its own API (JDBC API) that uses JDBC drivers (written in Java language).

Architectural diagram



JDBC Driver Types

- JDBC Driver is a software component that enables java application to interact with the database.There are 4 types of JDBC drivers:
- JDBC-ODBC bridge driver
- Native-API driver (partially java driver)
- Network Protocol driver (fully java driver)
- Thin driver (fully java 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.

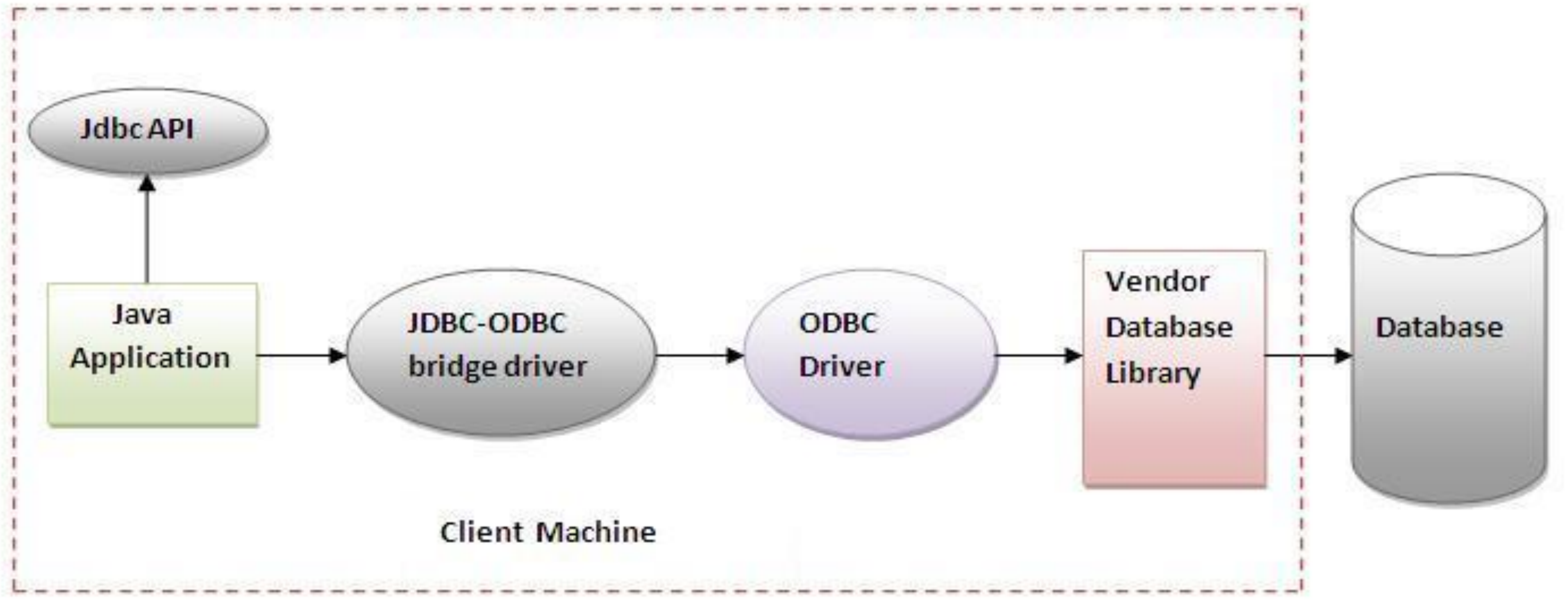


Figure- JDBC-ODBC Bridge Driver

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.

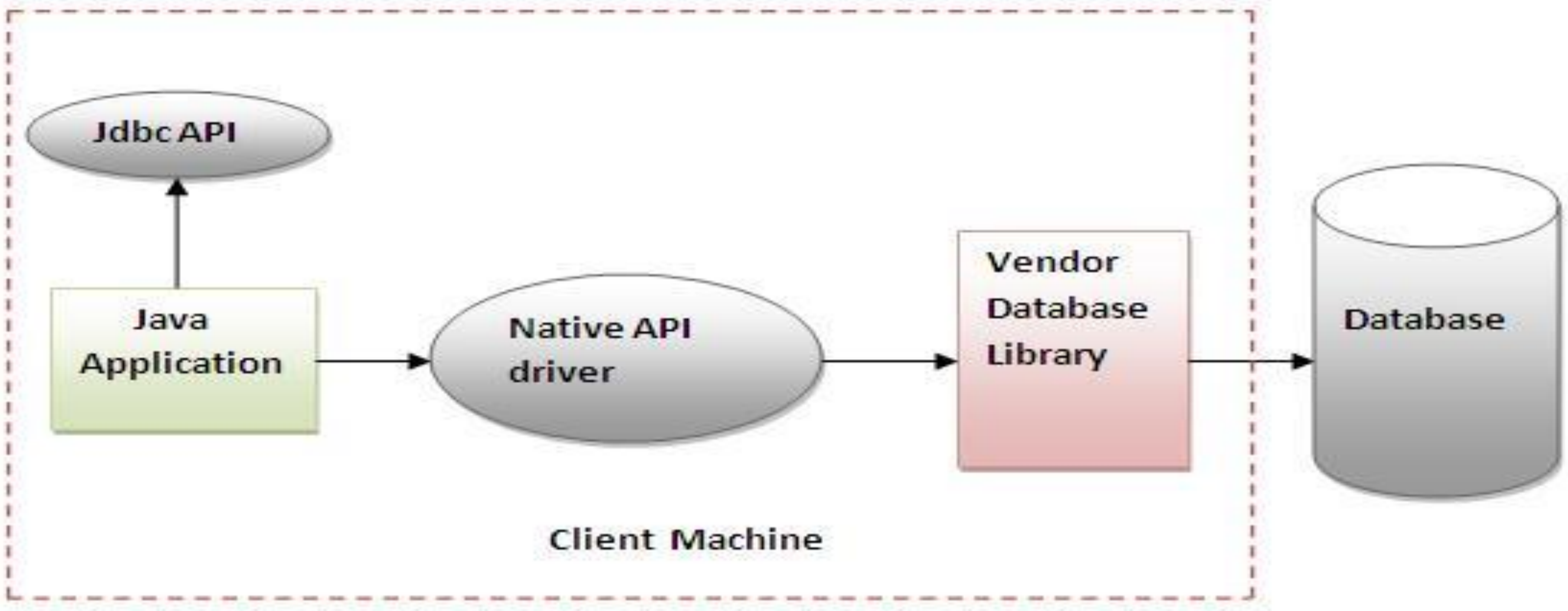


Figure- Native API Driver

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.

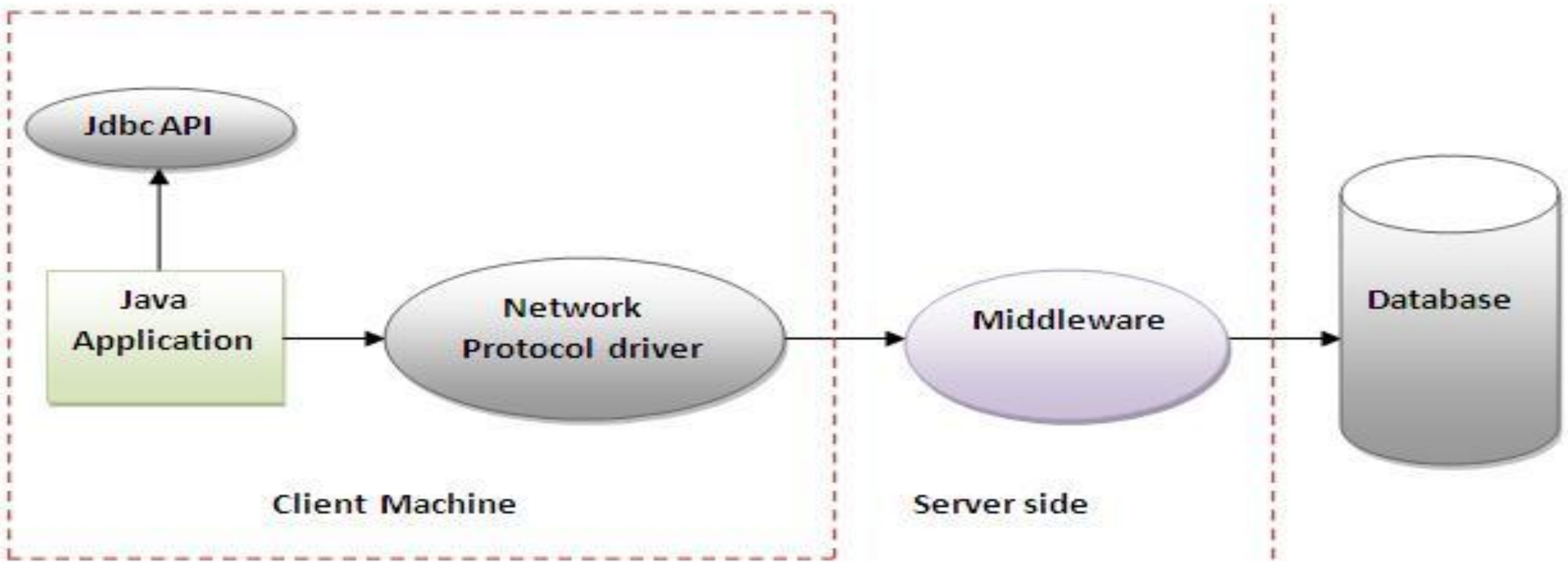


Figure- Network Protocol Driver

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.

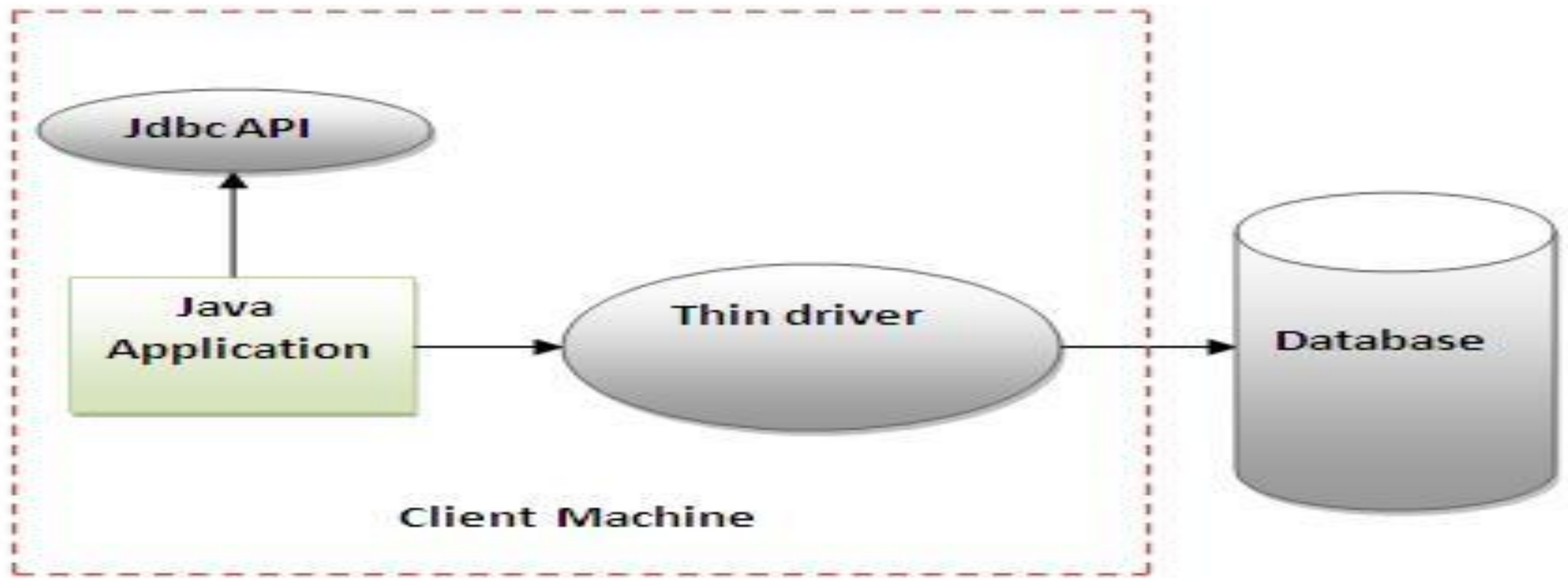


Figure- Thin Driver

5 Steps to connect to the database in java

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:

public static void forName(String className)throws ClassNotFoundException

2. Create the connection object.

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

SYNTAX:

a. public static Connection getConnection(String url)throws SQL Exception .

b. public static Connection getConnection(String url,String name, String password) throws SQLException .

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

public Statement createStatement()throws SQLException

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

public ResultSet executeQuery(String sql)throws SQLException .

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

public void close()throws SQLException

Example to Connect Java Application with MySQL database

```
import java.sql.*;
public class MySqlCon {
public static void main(String[] args) {
try{
Class.forName("com.mysql.cd.jdbc.Driver");
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/test","root","root");
Statement stmt=con.createStatement();
 String sql = "INSERT INTO employee " +     "VALUES ('manav',56)";
 stmt.executeUpdate(sql);
 System.out.println("Successfully inserted");
 ResultSet rs=stmt.executeQuery("select * from employee");
while(rs.next())
System.out.println(rs.getString(1)+""+rs.getInt(2));
sql="delete from employee where  empid=56";
stmt.executeUpdate(sql);
System.out.println("Successfully deleted");
 rs=stmt.executeQuery("select * from employee");
while(rs.next())
System.out.println(rs.getString(1)+""+rs.getInt(2));
con.close();
}catch(Exception e){ System.out.println(e);}  }  }
```