Nikit Gokhe

Class – Comp D1

Roll No. 224024

GR No. 21810522

**ASSIGNMENT NO.10**

**AIM:**

Implement MYSQL database connectivity with Java for Database navigation operations such as insert, delete, update etc. using ODBC/JDBC.

**OBJECTIVE:**

To MYSQL database connectivity with JAVA.

**THEORY:**

Java database connectivity (JDBC) is a standard application programming interface (API) specification that is implemented by different database vendors to allow Java programs to access their database management systems. The JDBC API consists of a set of interfaces and classes written in the Java programming language. Interestingly, JDBC is not an acronym and thus stands for nothing but popular as *Java Database Connectivity*. According to Sun (now merged in Oracle) JDBC is a trademark term and not an acronym. It was named to be redolent ofODBC.

JDBC implementation comes in form of a database driver for a particular DBMS vendor. A Java program that uses the JDBC API loads the specified driver for a particular DBMS before it actually connects to a database

To start using JDBC (Java database connectivity) first you would need a database system, and then a JDBC (Java database connectivity) driver for your database. During this article JDBC (Java database connectivity) will be explained with MySQL, but you can use it with any database system (such as Microsoft SQL, Oracle, IBM DB2, PostgresSQL), provided that you have a JDBC (Java database connectivity) driver for your database system.

Suppose you have MySQL and a functional Java environment installed on your system, where you write and execute Java programs. For MySQL, download MySQL Connector/J the official JDBC (Java database connectivity) driver for MySQL. You will get a JDBC driver mysql-connector-java-\*\*\*-bin.jar in zipped driver bundle. Place this JAR in Java's classpath because this JAR is the JDBC driver for MySQL and will be used by JDBC (Java database connectivity).

After having environment ready to experiment JDBC connectivity you need to create a database and at least one table in the database containing a few records, we name the example database EXPDB, and table inside EXPDB is EXPTABLE. I create this database with root

privileges, if you do so it's fine but if you create a database with different user then ensure that you have permissions to create, update and drop tables in the database you created (to perform above stated operations by JDBC - Java database connectivity). Now, let's create EXPDB database, a table EXPTABLE in EXPDB, and insert a few records as follows. Following steps we are executing manually, later we will connect to this database with the help of JDBC - Java databaseconnectivity.

# Connect toMySQL

1. **CreateDatabase**
2. **UseDatabase**
3. **Create atable**
4. **InsertRecords**
5. **ListRecords**

**JDBC Connection - Using JDBC**

So far we have gone through basic JDBC (Java database connectivity) concepts and created a trivial MySQL database to connect to it through JDBC within from a Java program. To access the database through Java program and JDBC, you would require following items in addition.

# JDBC Connection - DataBase URL orString

Database URL for JDBC connection or JDBC connection string follows more or less similar syntax to that of ordinary URLs. It tells the protocol used to connect to database, subprotocol, location of database, port number on which database listens client requests, and database name. The example syntax may look like jdbc:mysql://localhost:3306/EXPDB. Aforementioned URL specifies a MySQL database named EXPDB running on localhost on port 3306.0

# JDBC Connection - DriverClass

As we have obtained the JDBC driver in form of a JAR file (mysql-connector-java-\*\*\*-bin.jar) in which the driver for MySQL database is located. This driver needs to be registered in order to access EXPDB. Driver file name for MySQL is com.mysql.jdbc.Driver. This file has to be loaded into memory before you get connected to database, else you will result into java.sql.SQLException: No suitable driverexception.

# 3. JDBC Connection - Database User Name andPassword

To get a JDBC connection to the database you would require the username and password, it is the same username and password which we used, while connecting to MySQL.

# Java Program for JDBC Connection

Now we will start writing Java program to connect to EXPDB (our example database) through JDBC (Java database Connectivity) and perform INSERT and SELECT operations for demonstration. To illustrate this piece of work, we would take following steps, and finally collect all pieces of code to assemble the completeprogram.

# Register JDBC DriverClass

Registering JDBC driver class with the DriverManager means loading JDBC driver class in memory. You can load JDBC driver class in two ways. One way is to load the JDBC driver class in Java program is asfollows:

# Connect to Database through JDBCDriver

In order to connect to example database EXPDB you need to open a database connection in Java program.

# Execute SQL Statements through JDBCConnection

Now that you have a JDBC Connection object conn, you would like to execute SQL statements through the conn (JDBC connection) object. A connection in JDBC is a session with a specific database, where SQL statements are executed and results are returned within the context of a connection. To execute SQL statements you would need a Statement object that you would acquire by invoking createStatement() method onconn.

# Close JDBCConnection

JDBC connection to database is a session; it has been mentioned earlier. As soon as you close the session you are no longer connected to database; therefore, you would not be able to perform any operation on database. Closing connection must be the very last step when you are done with all database operations.

**INPUT:-**

package connectivity; import java.sql.\*; import java.io.\*;

import javax.sql.\*; class Connectivity {

public static void main(String args[]){ Connection con;

Statement connectionState; ResultSet rs;

intch;

boolean flag=true; String decision; intno;

try {

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

con=DriverManager.getConnection("jdbc:mysql://localhost:3306/sampl e","root","Rajshah$11");

System.out.println("Statement object created"); do {

the table");

System.out.println(); System.out.println("Menu:");

System.out.println("1.Insert Record into the table"); System.out.println("2.Update the Existing Record."); System.out.println("3.Delete existing record "); System.out.println("4.Display all the records from

System.out.println("5.Exit"); System.out.println("Enter your choice:"); BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

ch=Integer.parseInt(br.readLine()); switch(ch){

case1:

System.out.println("Enter Employee No:"); no = Integer.parseInt(br.readLine()); System.out.println("Enter Employee

Name:");

Salary:");

String name = br.readLine();

System.out.println("Enter Employee

int sal = Integer.parseInt(br.readLine());

values(?,?,?)"; con.prepareStatement(sql);

String sql="Insert into employee1 PreparedStatement p =

p.setInt(1,no); p.setString(2,name); p.setInt(3,sal); p.executeUpdate();

System.out.println("Record Added"); break;

case 2:

connectionState=con.createStatement(); while(flag) {

System.out.println("Enter Employee no. for the record you wish to update:");

no = Integer.parseInt(br.readLine()); System.out.println("Enter new

name:");

Salary:");

name = br.readLine();

System.out.println("Enter new

sal = Integer.parseInt(br.readLine()); sql = "Update employee1 set

employee1.name='"+name+"',employee1.sal="+sal+" where(((employee1.no)="+no+"))";

System.out.println(sql); int rows =

connectionState.executeUpdate(sql);

System.out.println(rows+"rows

successfully updated"); update more data:y/n:");

System.out.println("Do you want to

decision=br.readLine().toLowerCase(); if(decision.charAt(0)=='n')

flag=false;

}

break; case 3:

connectionState = con.createStatement();

System.out.println("Enter Employee no. for the record you wish to update:");

no = Integer.parseInt(br.readLine()); sql = "Delete from employee1 where

employee1.no ="+no+"";

Successfully");

connectionState.execute(sql); System.out.println("Record Deleted

break;

case 4:

connectionState = con.createStatement(); sql = "Select \* from employee1";

rs = connectionState.executeQuery(sql); while(rs.next()) {

System.out.println("\n"); System.out.println("\t"+rs.getInt(1));

System.out.println("\t"+rs.getString(2));

System.out.println("\t"+rs.getInt(3));

}

break; case 5:

System.exit(0); default:

System.out.println("Invalid Choice"); break;

}

}

while (ch!=4);

}

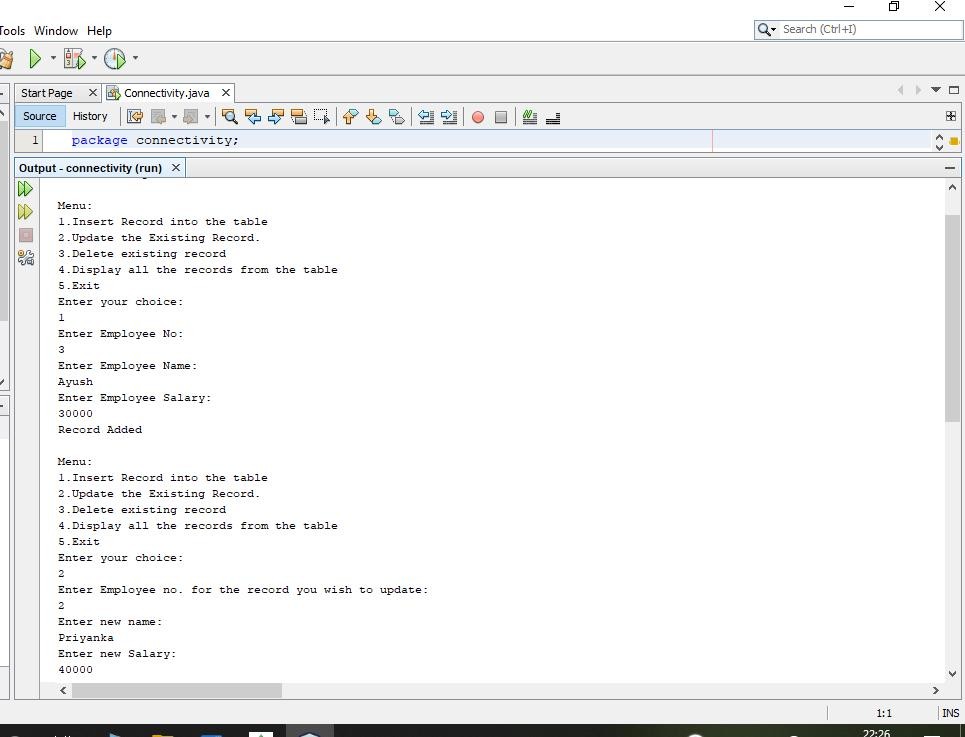
catch(Exception e) { System.out.println(e);

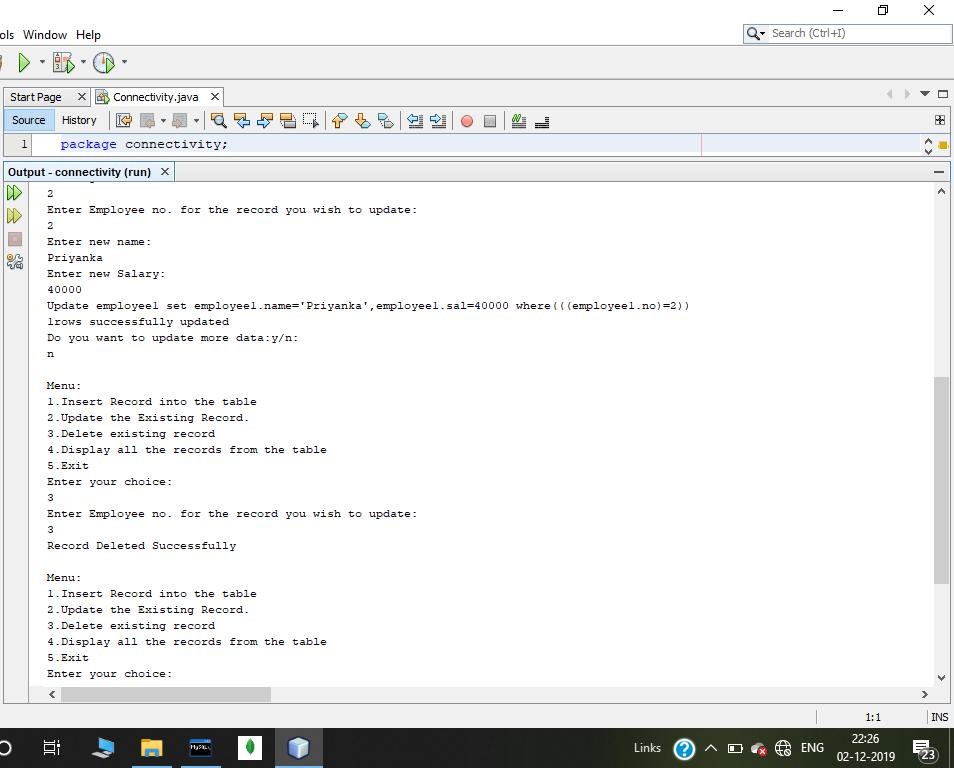
}

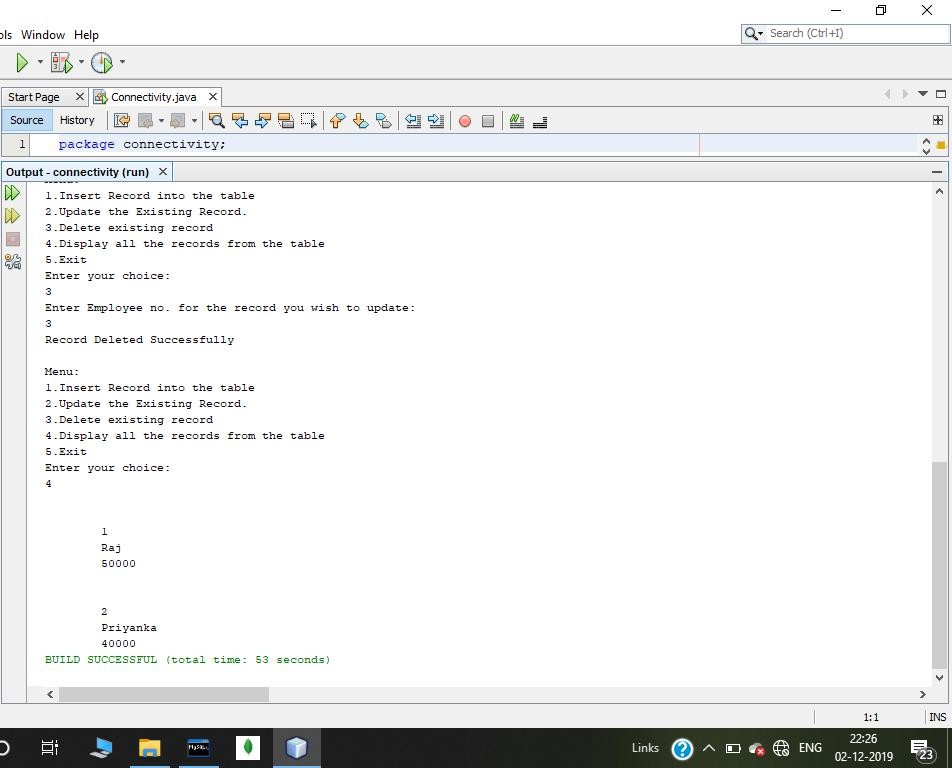
}

}

**OUTPUT:-**



****



**REMARK:-**

MYSQL connectivity implemented successfully.