Lab Assignment 17

Objective 1: Java to MySQL Connection (Using the JDBC-ODBC bridge)

Problem Statement:- A Java program for MySQL Connection

Explanation/Steps:-

We will first be creating a table in mysql for our example:

Create database **Toy**;

```
CREATE TABLE `humans` (
            `ID` int(10) unsigned NOT NULL AUTO_INCREMENT,
            `LastName` varchar(90) NOT NULL,
            `FirstName` varchar(90) NOT NULL,
            `Gender` varchar(90) NOT NULL,
            `Address` varchar(255) NOT NULL,
            `Phone` varchar(45) NOT NULL,
            PRIMARY KEY (`ID`)
          )
```

- Database will contain all the tables present in an application.

First import the necessary libraries for Database connection, java.sql:

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

These are the necessary tools we will need:
Connection - This represents the connection to a specific database. This is used to establish the connection with the database drive.
Statement - This is used to execute the SQL statement.
ResultSet - This holds the table of the data that is generated from a sql statement. For statements other than select, this is not necessary used since the statement would not return a table. Then instantiate your connection variables

```
Connection con;
Statement stmt;
ResultSet rs;
```

create an exception handler

```
          try{
//Your Code Goes Here

}catch(Exception e){
System.err.println(e);
}
```

To make your exception handling more specific, you can add 2 more catch before Exception containing SQLException and ClassNotFoundException

In the try block, insert your connection code, First is the forced loading of the Database Driver in Class.forName() followed by the declaration of objects we instantiated earlier.

```
        //Load the JdbcOdbc Driver
        Class.forName("com.mysql.jdbc.Driver");

        //Specify the Database URL where the DNS will be and the user and password
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/Toy","username","password");

                //Initialize the statement to be used, specify if rows are scrollable
        stmt=con.createStatement();
        //ResultSet will hold the data retrieved
        rs = stmt.executeQuery("SELECT * FROM Humans");
```

Then we now get the data from the ResultSet object

```
//Display the results
        while(rs.next()){

        System.out.println(rs.getInt("ID")     +     "     "     +     rs.getString("LastName")     +     "     "     +
rs.getString("FirstName"));

        }
```

OR

```
        while(rs.next()){

        System.out.println(rs.getInt(1) + " " + rs.getString(2) + " " + rs.getString(3));

        }
```

*Note: You can use the column name or the column position to retrieve values from the resultset.

- For further reference for insert update and delete you can click here.

For a full source code listing:

```
import java.sql.*;

        public class ViewingMySQL {
        public static void main(String[] args) {

        Connection con;
        Statement stmt;
        ResultSet rs;
        try{

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

        con = driverManager.getConnection(""jdbc:mysql://localhost:3306/Toy","username","password");

        stmt = con.createStatement();
        while(rs.next()){
        System.out.println(rs.getInt("ID") + " " + rs.getString("LastName") + " " +
rs.getString("FirstName"));
        }
        }catch(Exception e){
        System.err.println(e);
        }
        }
        }
```

Sample code:-

```
//STEP 1. Import required packages

import java.sql.*;



public class FirstExample {

  // JDBC driver name and database URL

  static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";

  static final String DB_URL = "jdbc:mysql://localhost/EMP";



  // Database credentials

  static final String USER = "username";

  static final String PASS = "password";



  public static void main(String[] args) {

  Connection conn = null;

  Statement stmt = null;

  try{
```

```
//STEP 2: Register JDBC driver

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


//STEP 3: Open a connection

System.out.println("Connecting to database...");

conn = DriverManager.getConnection(DB_URL,USER,PASS);


//STEP 4: Execute a query

System.out.println("Creating statement...");

stmt = conn.createStatement();

String sql;

sql = "SELECT id, first, last, age FROM Employees";

ResultSet rs = stmt.executeQuery(sql);


//STEP 5: Extract data from result set

while(rs.next()){

   //Retrieve by column name

   int id = rs.getInt("id");

   int age = rs.getInt("age");

   String first = rs.getString("first");

   String last = rs.getString("last");


   //Display values

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

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

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

```
      System.out.println(", Last: " + last);
   }
   //STEP 6: Clean-up environment
   rs.close();
   stmt.close();
   conn.close();
}catch(SQLException se){
   //Handle errors for JDBC
   se.printStackTrace();
}catch(Exception e){
   //Handle errors for Class.forName
   e.printStackTrace();
}finally{
   //finally block used to close resources
   try{
      if(stmt!=null)
         stmt.close();
   }catch(SQLException se2){
   }// nothing we can do
   try{
      if(conn!=null)
         conn.close();
   }catch(SQLException se){
      se.printStackTrace();
   }//end finally try
}//end try
```

```
  System.out.println("Goodbye!");

}//end main

}//end FirstExample
```

Objective 2: Sample JDBC Program for Excel

Problem Statement:- A Java program for Excel database.

Code/Explanation:-

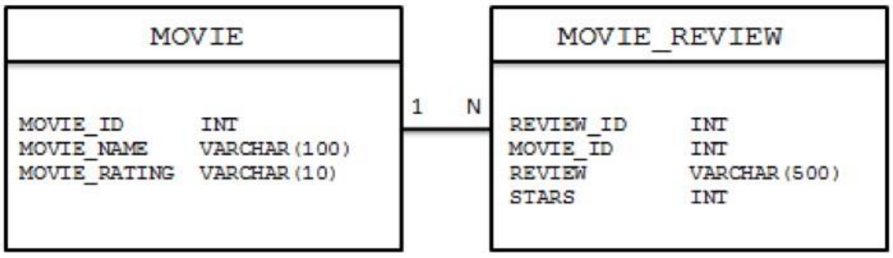
```
import java.sql.*;            // Use classes in java.sql package

public class ExcelSelectTest {

  public static void main(String[] args) {
    try (
      // Step 1: Allocate a database "Connection" object
      Connection conn = DriverManager.getConnection(
          "jdbc:odbc:ebookshopODBC"); // Access/Excel

      // Step 2: Allocate a "Statement" object in the Connection
      Statement stmt = conn.createStatement();
    ) {
      // Excel connection, by default, is read-only.
      // Need to turn it off to issue INSERT, UPDATE, ...
      conn.setReadOnly(false);

      // Step 3: Execute a SQL SELECT query, the query result
      // is returned in a "ResultSet" object.
      // Table name is the sheet's name in the form of [sheet-name$]
      String strSelect = "select title, price, qty from [books$]";
      System.out.println("The SQL query is: " + strSelect); // Echo For debugging

      ResultSet rset = stmt.executeQuery(strSelect);

      // Step 4: Process the ResultSet by scrolling the cursor forward via next().
      // For each row, retrieve the contents of the cells with getXxx(columnName).
      System.out.println("The records selected are:");
      int rowCount = 0;
      while(rset.next()) { // Move the cursor to the next row
        String title = rset.getString("title");
        double price = rset.getDouble("price");
        int    qty = rset.getInt("qty");
        System.out.println(title + ", " + price + ", " + qty);
        ++rowCount;
      }
      System.out.println("Total number of records = " + rowCount);

      // Try INSERT
      int returnCode = stmt.executeUpdate(
        "insert into [books$] values (1002, 'Java 101', 'Tan Ah Teck', 2.2, 2)");
      System.out.println(returnCode + " record(s) inserted.");

      // Try UPDATE
      returnCode = stmt.executeUpdate(
        "update [books$] set qty = qty+1 where id = 1002");
      System.out.println(returnCode + " record(s) updated.");

    } catch(SQLException ex) {
      ex.printStackTrace();
    }
    // Step 5: Close the resources - Done automatically by try-with-resources
  }
}
```

LAB EXERCISE:

A movie database application that allows a user to look up information about a movie and provide a review for a movie. Your application should provide the following features:

- List all movies that are rated a particular rating (PG, PG-13, R, etc) entered by the user.
- All a user to post a review for a particular movie. A review consists of review text and a star rating with
  - 5 stars = Excellent
  - 4 stars = Good
  - 3 stars = Neutral
  - 2 stars = Poor
  - 1 star = Very Poor
- List all reviews for a movie and include the average star rating for the movie

Your solution should read the database connection information (jdbc driver and connection url) from a properties file called database. Properties. You can create a console based application to implement this application . The only requirement is that the class containing the main method should be called `MovieDBApp`.


MOVIE
MOVIE_ID INT
MOVIE_NAME VARCHAR(100)
MOVIE_RATING VARCHAR(10)
1
N
MOVIE_REVIEW
REVIEW_ID INT
MOVIE_ID INT
REVIEW VARCHAR(500)
STARS INT