BÀI TẬP TUẦN 3

I. Thực hành tại lớp

1. Tạo Database sau trong MySQL

Yêu cầu: Viết câu lệnh tạo database SinhVien bằng file sql, import file để tạo bảng trong MySQL

HOCPHI	**MSSV**	**Int(5)**
	HO	**Varchar(10)**
	TEN	**Varchar(10)**
	HOCPHI	**Int(3)**

MSSV	HO	TEN	HOCPHI
05201	**Duong**	**Hoang**	**199**
05203	**Nguyen**	**Quan**	**199**
05205	**Ly**	**Quang**	**199**

2. Viết chương trình bằng ngôn ngữ Java để kết nối với Database trên, hiển thị học phí của tất cả sinh viên ra màn hình

2.1.Add thư viện driver kết nối

Để kết nối ứng dụng Java với MySQL Database, cần add lib mysqlconnector.jar file (https://dev.mysql.com/downloads/connector/j/)

Chọn menu Tools.Library để hiển thị cửa sổ Library Manager, chọn button New Library để tạo mới 1 thư viện cần dùng cho ứng dụng, chọn button Add JAR/Folder rồi duyệt tìm và chọn file thư viện miêu tả driver "MySQL Connector/J" được dùng trong ứng dụng

- Lớp Driver cho MySQL Database là **com.mysql.jdbc.Driver**.

- Địa chỉ kết nối cho MySQL Database là **jdbc:mysql://localhost:3306/sinhvien**. Trong đó:

- **jdbc** là API
- **mysql** là cơ sở dữ liệu
- **localhost** là tên server mà MySQL đang chạy (chúng ta cũng có thể sử dụng địa chỉ IP tại đây)
- **3306** là port mặc định đối với MACOS, hoặc 8888 đối với Window
- **sinhvien** là tên của cơ sở dữ liệu (tùy theo cơ sở dữ liệu muốn sử dụng)

- Username mặc định cho MySQL Database là **root**. Pass để kết nối ở file hướng dẫn cài đặt

2.2. Các bước để thiết lập một kết nối JDBC với một Database MySQL

Bước 1: Import các JDBC package

Để sử dụng JDBC package cập nhật và xóa dữ liệu trong các bảng SQL:

```
import java.sql.* ;  // cho cac chuong trinh JDBC chuan
import java.math.* ; // de ho tro BigDecimal va BigInteger
```

Bước 2: Đăng ký JDBC driver

Một trong hai cách sau:

Sử dụng phương thức Class.forName(): phổ biến

```
public static void Class.forName(String ten_lop) throws ClassNotFoundException
```

Ví dụ, để đăng ký Oracle driver, bạn sử dụng mẫu sau:

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

Sử dụng DriverManager.registerDriver()

Sử dụng khi một JDK không tuân theo JVM (được cung cấp bởi Microsoft chẳng hạn),

```
Driver myDriver = new oracle.jdbc.driver.OracleDriver();
  DriverManager.registerDriver( myDriver );
```

Bước 3: Tạo địa chỉ Database URL chính xác

Đây là địa chỉ trỏ tới cơ sở dữ liệu của bạn. Việc tạo một địa chỉ Database URL chính xác là rất quan trọng, và đây thường là bước hay xuất hiện lỗi trong khi thành lập một kết nối. Bảng dưới đây liệt kê một số tên JDBC driver và Database URL phổ biến:

Trong đó, **hostname** là tên server mà cơ sở dữ liệu đang chạy (có thể là localhost hoặc địa chỉ IP của bạn); **port Number** là số hiệu cổng và **databaseName** là tên của cơ sở dữ liệu.

RDBMS	Tên JDBC Driver	Định dạng URL
MySQL	com.mysql.jdbc.Driver	**jdbc:mysql://**hostname/ databaseName
ORACLE	oracle.jdbc.driver.OracleDriver	**jdbc:oracle:thin:**@hostname:port Number:databaseName
DB2	COM.ibm.db2.jdbc.net.DB2Driver	**jdbc:db2:**hostname:port Number/databaseName

Sybase	com.sybase.jdbc.SybDriver	**jdbc:sybase:Tds:**hostname: port Number/databaseName

Phần in đậm là phần bạn không nên thay đổi, bạn chỉ cần thay đổi phần còn lại theo yêu cầu của bạn.

Bước 4: Tạo đối tượng Connection

Sau khi đã tải driver, bạn có thể thiết lập một kết nối bởi sử dụng phương thức DriverManager.getConnection(). Phương thức này có ba mẫu:

Cách 1: getConnection(String url)

Mẫu phương thức này chỉ yêu cầu một Database URL. Tuy nhiên, trong trường hợp này, Database URL cũng đã bao cả username và password.

Ví dụ: sử dụng Thin driver của Oracle, với host name là localhost, cổng là 3306 và tên cơ sở dữ liệu là sinhvien, thì Database URL sẽ là:

```
jdbc:oracle:thin:username/password@localhost:3306:sinhvien
```

Do đó, kết nối trên có thể được tạo như sau:

```
String URL = "jdbc:oracle:thin:username/password@localhost:3306:sinhvien";
Connection conn = DriverManager.getConnection(URL);
```

Cách 2: getConnection(String url, Properties info)

Mẫu phương thức này yêu cầu một Database URL và một đối tượng Properties. Một đối tượng Properties giữ một tập hợp các cặp key-value. Mẫu này được sử dụng để truyền các thuộc tính tới driver trong một lời gọi tới phương thức getConnection().

Để tạo kết nối như ví dụ trên, bạn sử dụng:

```
import java.util.*;
String URL = "jdbc:oracle:thin:@localhost:3306:sinhvien";
Properties info = new Properties( );
info.put( "user", "username" );
info.put( "password", "password" );
Connection conn = DriverManager.getConnection(URL, info);
```

Cách 3: getConnection(String url, String user, String password): Phổ biến

Sử dụng phương thức này để truyền *một Database URL, một username và một password*.

```
String URL = "jdbc:oracle:thin:@vietjack:3306:sinhvien";
String USER = "username";
String PASS = "password"
Connection conn = DriverManager.getConnection(URL, USER, PASS);
```

Đóng kết nối JDBC

Để đóng kết nối đã mở ở trên, gọi phương thức close() có cú pháp như sau:

```
public void conn.close() throws SQLException
```

2.3. Statement Interface trong JDBC

Interface này cung cấp nhiều phương thức để thực thi các truy vấn với cơ sở dữ liệu và trả về kết quả mà nó tạo ra (ResultSet).

Tạo 1 statement

```
Statement stmt=con.createStatement();
```

Các loại statement

a. public ResultSet executeQuery(String sql)

Thực thi một lệnh SQL đã cho và trả về một đối tượng Resultset đơn. Tham số sql là một lệnh SQL.

b. public int executeUpdate(String sql)

Thực thi lệnh SQL đã cho, có thể là INSERT, UPDATE, DELETE hoặc một lệnh SQL mà không trả về bất kỳ cái gì như lệnh SQL DDL.

c. public boolean execute(String sql)

Thực thi lệnh SQL mà có thể trả về nhiều kết quả. Thương được sử dụng khi thực thi một store procedure trả về nhiều kết quả hoặc thực thi một chuỗi SQL chưa biết

d. public int[] executeBatch()

Được sử dụng để thực thi một nhóm các lệnh và nếu thành công thì trả về một mảng. Các phần tử trong mảng trả về được sắp xếp theo thứ tự tương ứng với lệnh trong batch. Các phần tử trong mảng được trả về này có thể là:

Một số >= 0, chỉ rằng lệnh được thực thi thành công và đó là số hàng bị tác động trong cơ sở dữ liệu.

Một giá trị SUCCESS_NO_INFO chỉ rằng lệnh được thực thi thành công nhưng số hàng bị tác động là chưa biết.

Một giá trị EXECUTE_FAILED chỉ rằng lệnh đã thất bại.

2.4. Hiện thực cho ví dụ trên

```
package com.local.jdbc;

import java.sql.*;
class ViDuJDBC{
public static void main(String args[]){
try{
// Buoc 1: Tai lop Driver
Class.forName("com.mysql.jdbc.Driver");

// Buoc 2: Tao doi tuong Connection
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/sinhvien","root","123456");
// sihhvien la ten co so du lieu, root la username va mat khau la 123456

// Buoc 3: Tao doi tuong Statement
Statement stmt=con.createStatement();

// Buoc 4: Thuc thi truy van
ResultSet rs=stmt.executeQuery("SELECT * FROM hocphi");

while(rs.next())
System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getString(3)+ rs.getInt(4));
```

```
// Buoc 5: Dong doi tuong Connection
con.close();


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


}
}
```