Session 10

Advanced JDBC Features

Phần I - Thực hiện trong 30 phút

1. *Mục tiêu*

- Hiểu bản chất thuộc tính của ResultSet
- Hiểu và sử dụng đối tượng PrepareStatement.
- Biết thực hiện lệnh theo lô.
- Hiểu biết về Transaction sử dụng trong JDBC, có thể khôi phục, commit hoặc giải phóng SavePoint
- Hiểu biết và sử dụng CallableStatement để thực thi lệnh gọi Store Procedure.

2. *Thực hiện*

Bài thực hành 1: Từ ví dụ ứng dụng quản lý sinh viên Lab-06, chỉnh sửa mã nguồn theo hướng dẫn sau để hiểu thuộc tính của ResultSet.

Chú giải:

ResultSet là đối tượng dùng để lưu trữ dữ liệu truy vấn từ server về local. Mặc định ResultSet không thể cập nhật hoặc cuộn tới, lui được, nó chỉ có thể tiến tới (gọi hàm next()). ResultSet có đặc điểm như sau:

Scrollable: khả năng cuộn tới, lui, vị trí đầu dòng, cuối dòng, truy cập bất kỳ dòng dữ liệu nào tại bất cứ thời điểm nào.

Updatable: mọi cập nhật dữ liệu như Sửa, Xóa, Thêm mới sau khi thực hiện trên ResultSet, nếu muốn nó có thể tự động cập nhật thay đổi đó lên trực tiếp database.

Holdable: nếu được cài đặt chế độ này, mọi thay đổi trên ResultSet sẽ được giữ lại chưa cập nhật lên database cho đến khi gọi commit().

Bước 1: Tạo class SinhVien

```
package demo.jp2.lab07;

import java.util.Scanner;

/**
 *
 * @author minhvufc
```

```
 */
public class SinhVien {

    private int id;
    private String rollNumber;
    private String name;
    private String address;
    private String phoneNumber;
    private int gender;

    public SinhVien() {
    }

    public SinhVien(int id, String rollNumber, String name, String address, String
phoneNumber, int gender) {
        this.id = id;
        this.rollNumber = rollNumber;
        this.name = name;
        this.address = address;
        this.phoneNumber = phoneNumber;
        this.gender = gender;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getRollNumber() {
        return rollNumber;
    }

    public void setRollNumber(String rollNumber) {
        this.rollNumber = rollNumber;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
```

```
    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public String getPhoneNumber() {
        return phoneNumber;
    }

    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }

    public int getGender() {
        return gender;
    }

    public void setGender(int gender) {
        this.gender = gender;
    }

    public void nhapThongTin() {
        System.out.println("Nhập thông tin sinh viên:");
        Scanner nhap = new Scanner(System.in);
        System.out.println("Mã sinh viên: ");
        this.rollNumber = nhap.nextLine();
        System.out.println("Họ và Tên: ");
        this.name = nhap.nextLine();
        System.out.println("Địa chỉ: ");
        this.address = nhap.nextLine();
        System.out.println("SĐT: ");
        this.phoneNumber = nhap.nextLine();
        System.out.println("Giới tính(Nam = 1 | Nữ = 0): ");
        this.gender = nhap.nextInt();
    }
}
```

Tạo lớp MainClass

```
package demo.jp2.lab07;

import java.sql.CallableStatement;
import java.sql.Connection;
```

```
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author minhvufc
 */
public class MainClass {

    Connection conn = null;
    SinhVien sv;

    //Bước 1: Tạo kết nối tới CSDL
    public void connect() {
        try {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            conn =
DriverManager.getConnection("jdbc:sqlserver://localhost:1433;databaseName=qlsv",
"sa", "1234567");
            System.out.println("Kết nối tới CSDL thành công");
        } catch (ClassNotFoundException | SQLException ex) {
            Logger.getLogger(MainClass.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    //Bước 2: Tạo hàm hiển thị lên menu
    private int showMenu() {
        System.out.println("===== MENU =====");
        System.out.println("1. Nhập thông tin sinh viên");
        System.out.println("2. Danh sách sinh viên");
        System.out.println("3. Sửa sinh viên");
        System.out.println("4. Xóa sinh viên");
        System.out.println("5. Thêm nhóm sinh viên");
        System.out.println("6. Tìm sinh viên");
        System.out.println("7. Thoát");
        System.out.println("Vui lòng chọn từ 1 -> 7");
        Scanner nhap = new Scanner(System.in);
        return nhap.nextInt();
    }
```

```
//Bước 3: Tạo hàm nhập thông tin
public void nhapThongTin() throws SQLException {
    conn.setAutoCommit(false); // KHÓA tự động commit dữ liệu lên server
    boolean isStop = false;
    String sql = "INSERT INTO tblsinhvien("
            + "rollnumber, name, address, phone, gender) "
            + "VALUES (?,?,?,?,?)";
    PreparedStatement ps = conn.prepareStatement(sql);
    do {
        SinhVien objS = new SinhVien();
        objS.nhapThongTin();
        ps.setString(1, objS.getRollNumber());
        ps.setString(2, objS.getName());
        ps.setString(3, objS.getAddress());
        ps.setString(4, objS.getPhoneNumber());
        ps.setInt(5, objS.getGender());
        ps.addBatch();

        System.out.println("Bạn muốn nhập tiếp không (1: Có | 2: Không)");
        Scanner nhap = new Scanner(System.in);
        if (nhap.nextInt() != 1) {
            isStop = true;
        }
    } while (!isStop);
    // Không nhập nữa mới commit thay đổi
    int updateCount[] = ps.executeBatch();
    conn.commit(); // MỞ KHÓA tự động commit
}

//Bước 4: Tạo hàm nhập thông tin sinh viên cần sửa
//Dữ liệu nhập vào được lưu ở biến SinhVien đã khai báo ở trên
private void suaThongTin() {
    System.out.println("Nhập thông tin sửa:");
    Scanner nhap = new Scanner(System.in);
    sv = new SinhVien();
    System.out.println("Nhập Rollnumber sinh viên cần sửa: ");
    sv.setRollNumber(nhap.nextLine());
    System.out.println("Họ và Tên: ");
    sv.setName(nhap.nextLine());
    System.out.println("Địa chỉ: ");
    sv.setAddress(nhap.nextLine());
    System.out.println("SĐT: ");
    sv.setPhoneNumber(nhap.nextLine());
    System.out.println("Giới tính(Nam = 1 | Nữ = 0): ");
    sv.setGender(nhap.nextInt());
}
```

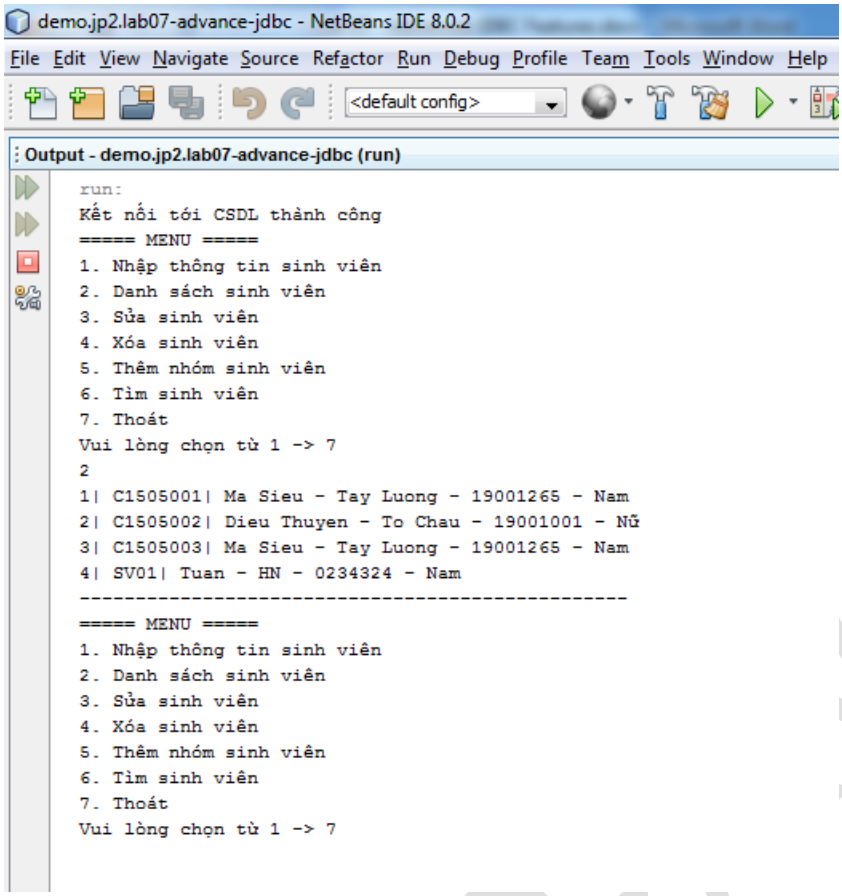
```
    //Bước 5: Hàm load dữ liệu từ database và hiển thị ra màn hình
    public void load() throws SQLException {
        String sql = "SELECT * FROM tblsinhvien";
        PreparedStatement pstmt = conn.prepareStatement(sql);
        ResultSet result = pstmt.executeQuery();

        while (result.next()) {
            String id = result.getString("id");
            String rollnumber = result.getString("rollnumber");
            String name = result.getString("name");
            String address = result.getString("address");
            String phone = result.getString("phone");
            String gender = "Nam";
            if (result.getInt("gender") == 0) {
                gender = "Nữ";
            }
            System.out.println(id + "| " + rollnumber + "| " + name + " - " + address +
" - " + phone + " - " + gender);
        }
        System.out.println("-----------------------------------------------");
    }

    //Bước 6: Hàm thực thi cập nhật dữ liệu
    public void update() throws SQLException {
        System.out.println("Cập nhật thông tin sinh viên");
        suaThongTin();
        String sql = "update tblsinhvien set name=?, address=?, phone=?, gender=?
where rollnumber=?";
        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, sv.getName());
        pstmt.setString(2, sv.getAddress());
        pstmt.setString(3, sv.getPhoneNumber());
        pstmt.setBoolean(4, (sv.getGender() == 1));
        pstmt.setString(5, sv.getRollNumber());

        int i = pstmt.executeUpdate();
        if (i > 0) {
            System.out.println("Sửa thành công");
        } else {
            System.out.println("Thất bại, vui lòng kiểm tra lại dữ liệu");
        }
    }

    //Bước 7: Hàm xóa thông tin sinh viên
    public void delete() throws SQLException {
        System.out.println("Nhập mã sinh viên muốn xóa: ");
```

```
        Scanner nhap = new Scanner(System.in);
        String rollNumber = nhap.nextLine();
        String sql = "DELETE FROM tblsinhvien WHERE "
                + "rollnumber = '" + rollNumber + "'";
        Statement st = conn.createStatement();
        int row = st.executeUpdate(sql);
        if (row > 0) {
            System.out.println("Xóa thành công");
        } else {
            System.out.println("Thất bại, vui lòng kiểm tra lại dữ liệu");
        }
    }

    //Bước 8: Hàm lấy về thông tin sinh viên theo rollnumber
    public void getStudentByID() throws SQLException {
        System.out.println("Nhập Rollnumber sinh viên muốn hiển thị: ");
        Scanner nhap = new Scanner(System.in);
        String roll = nhap.nextLine();

        String sql = "{CALL sp_getstudent(?)}";
        CallableStatement cs = conn.prepareCall(sql);
        cs.setString(1, roll);
        ResultSet rs = cs.executeQuery();

        if (rs.next()) {
            System.out.println("Tìm thấy sinh viên có roll:"+roll);
            System.out.println("Rollnumber:"+rs.getString(2));
            System.out.println("Name:"+rs.getString(3));
            System.out.println("Address:"+rs.getString(4));
            System.out.println("Phone:"+rs.getString(5));
            System.out.println("Gender:"+(rs.getBoolean(6) ? "Nam":"Nữ"));
        }else{
            System.out.println("Không tìm thấy sinh viên nào có roll:"+roll);
        }

    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        MainClass main = new MainClass();
        main.connect();
        int choose = 0;

        do {
            choose = main.showMenu();
```

```
            switch (choose) {
                case 1: {
                    try {
                        main.nhapThongTin();
                    } catch (SQLException ex) {
                        Logger.getLogger(MainClass.class.getName()).log(Level.SEVERE,
null, ex);
                    }
                }
                break;
                case 2: {
                    try {
                        main.load();
                    } catch (SQLException ex) {
                        Logger.getLogger(MainClass.class.getName()).log(Level.SEVERE,
null, ex);
                    }
                }
                break;
                case 3: {
                    try {
                        main.update();
                    } catch (SQLException ex) {
                        Logger.getLogger(MainClass.class.getName()).log(Level.SEVERE,
null, ex);
                    }
                }
                break;
                case 4: {
                    try {
                        main.delete();
                    } catch (SQLException ex) {
                        Logger.getLogger(MainClass.class.getName()).log(Level.SEVERE,
null, ex);
                    }
                }
                break;
                case 5: {
                    try {
                        main.nhapThongTin();
                    } catch (SQLException ex) {
                        Logger.getLogger(MainClass.class.getName()).log(Level.SEVERE,
null, ex);
                    }
                }
                break;
```

```
                case 6: {
                    try {
                        main.getStudentByID();
                    } catch (SQLException ex) {
                        Logger.getLogger(MainClass.class.getName()).log(Level.SEVERE,
null, ex);
                    }
                }
                break;
                case 7:
                    System.exit(0);
            }
        } while (choose > 0 && choose < 7);

        try {
            // Đóng kết nối
            main.conn.close();
        } catch (SQLException ex) {
            Logger.getLogger(MainClass.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

Bước 3: Build và chạy chương trình.



Bài thực hành 2: Sử dụng CachedRowset để tạo kết nối tới CSDL quản lý sinh viên trên, thực hiện các thao tác CRUD.

Chú giải:

CachedRowset là một interface được xây dựng sẵn cùng với Resultset. Khác với JDBCRowset, interface này là một disconnected rowset – tức là nó thực hiện kết nối tới CSDL, lấy về Resultset rồi ngắt kết nối, mọi thực thi thay đổi dữ liệu sẽ thao tác trên dữ liệu cục bộ, khi có lệnh gọi cập nhật thì CachedRowset mở lại kết nối tới database và thực thi thay đổi. Trong trường hợp không cần kết nối dữ liệu liên tục thì việc sử dụng CachedRowset sẽ làm giảm tải phía server CSDL rất nhiều.

Bước 1: Viết class DemoCachedRowset và hàm hiển thị menu.

```
package demo.jp2.lab07;
```

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.sql.rowset.CachedRowSet;
import javax.sql.rowset.RowSetFactory;
import javax.sql.rowset.RowSetProvider;

/**
 *
 * @author MinhVuFC
 */
public class DemoCachedRowset {

    // Thông số kết nối
    final static String URL = "jdbc:sqlserver://localhost:1433;databaseName=qlsv";
    final static String USERNAME = "sa";
    final static String PASSWORD = "1234567";

    // Các trường dữ liệu trong database
    final static String ID = "id";
    final static String ROLLNUMBER = "rollnumber";
    final static String NAME = "name";
    final static String ADDRESS = "address";
    final static String PHONE = "phone";
    final static String GENDER = "gender";

    private void getAllRowset(CachedRowSet rowset) throws SQLException {
        rowset.beforeFirst(); // Di chuyển con trỏ tới dòng đầu tiên;
        // In kết quả truy vấn ra màn hình

System.out.println("==========================================
==========");
        while (rowset.next()) {
            int rowNum = rowset.getRow();
            // String id = rowset.getString("id");
            String rollnumber = rowset.getString("rollnumber");
            String name = rowset.getString("name");
            String address = rowset.getString("address");
            String phone = rowset.getString("phone");
            String gender = rowset.getBoolean("gender") ?"Nam":"Nữ";
            System.out.println(rowNum + "#\t " + " " + rollnumber + "| " + name + " -
" + address + " - " + phone + " - " + gender);
        }
```

```
System.out.println("==========================================
==========");
    }

    /**
     * Hàm hiển thị danh sách sinh viên lấy từ CSLD
     *
     * @param rowset Biến interface truyền vào là đối tượng connected rowset
     */
    private void insert(CachedRowSet rowset) throws SQLException {
        // Khởi tạo đối tượng Sinh Viên và nhập dữ liệu
        SinhVien sinhVien = new SinhVien();
        System.out.println("Nhập thông tin sinh viên:");
        Scanner nhap = new Scanner(System.in);
        System.out.println("Mã sinh viên: ");
        sinhVien.setRollNumber(nhap.nextLine());
        System.out.println("Họ và Tên: ");
        sinhVien.setName(nhap.nextLine());
        System.out.println("Địa chỉ: ");
        sinhVien.setAddress(nhap.nextLine());
        System.out.println("SĐT: ");
        sinhVien.setPhoneNumber(nhap.nextLine());
        System.out.println("Giới tính(Nam = 1 | Nữ = 0): ");
        sinhVien.setGender(nhap.nextInt());

        rowset.moveToInsertRow(); // Di chuyển con trỏ tới vị trí THÊM
        rowset.updateNull(ID); // Trong CatchedRowset buộc phải có nếu ko sẽ ko thực
hiện dc
        rowset.updateString(ROLLNUMBER, sinhVien.getRollNumber());
        rowset.updateString(NAME, sinhVien.getName());
        rowset.updateString(ADDRESS, sinhVien.getAddress());
        rowset.updateString(PHONE, sinhVien.getPhoneNumber());
        rowset.updateInt(GENDER, sinhVien.getGender());
        rowset.insertRow(); // Lệnh thực thi thêm dữ liệu vào dữ liệu tạm của
CachedRowset
        rowset.moveToCurrentRow(); // Di chuyển con trỏ tới dòng hiện tại trước khi
thêm
        /*
         Khác với JDBCRowset, mọi thay đổi được thực thi ngay lập tức thì đối với
         CachedRowset ta cần phải gọi hàm acceptChanges() để việc thực thi mới
         thực sự thay đổi trên CSDL
         */
        rowset.acceptChanges();
    }

    private void update(CachedRowSet rowset) throws SQLException {
```

```
        // Di chuyển con trỏ Rowset xuống cuối cùng
        rowset.last();
        // Hiển thị số lượng bản ghi lấy được từ CSDL
        System.out.println("Tìm thấy: " + rowset.getRow() + " bản ghi");

        // 1. Nhập thứ tự sinh viên muốn sửa
        System.out.println("Cập nhật thông tin sinh viên");
        System.out.println("Nhập thứ tự sinh viên muốn sửa: ");
        Scanner nhap = new Scanner(System.in);
        int rowNum = nhap.nextInt();

        // 2. Di chuyển tới vị trí muốn sửa
        rowset.absolute(rowNum);

        // 3. Nhập thông tin Sinh Viên
        SinhVien sinhVien = new SinhVien();
        System.out.println("Nhập thông tin sửa:");
        nhap.nextLine(); // Xóa bộ nhớ đệm tạm - phòng bị trôi dòng
        System.out.println("Họ và Tên: ");
        sinhVien.setName(nhap.nextLine());
        System.out.println("Địa chỉ: ");
        sinhVien.setAddress(nhap.nextLine());

        // 4. Thực hiện chỉnh sửa dữ liệu trên Rowset
        rowset.updateString(NAME, sinhVien.getName());
        rowset.updateString(ADDRESS, sinhVien.getAddress());

        // 5. Gọi lệnh thi hành sự thay đổi tạm thời trên CachedRowset
        rowset.updateRow();
        rowset.acceptChanges(); // Thực thi thay đổi trên CSDL
        System.out.println("Cập nhật thành công");
    }

    private void delete(CachedRowSet rowset) throws SQLException {
        // 1. Nhập thứ tự sinh viên muốn sửa
        System.out.println("Cập nhật thông tin sinh viên");
        System.out.println("Nhập thứ tự sinh viên muốn sửa: ");
        Scanner nhap = new Scanner(System.in);
        int rowNum = nhap.nextInt();

        // 2. Di chuyển tới vị trí muốn sửa
        rowset.absolute(rowNum);

        // 3. Gọi lệnh thực thi xóa trên CSDL
        System.out.println("Xóa Sinh Viên tên là " + rowset.getString(NAME) + " tại địa
chỉ " + rowset.getString(ADDRESS));
        rowset.deleteRow();
```

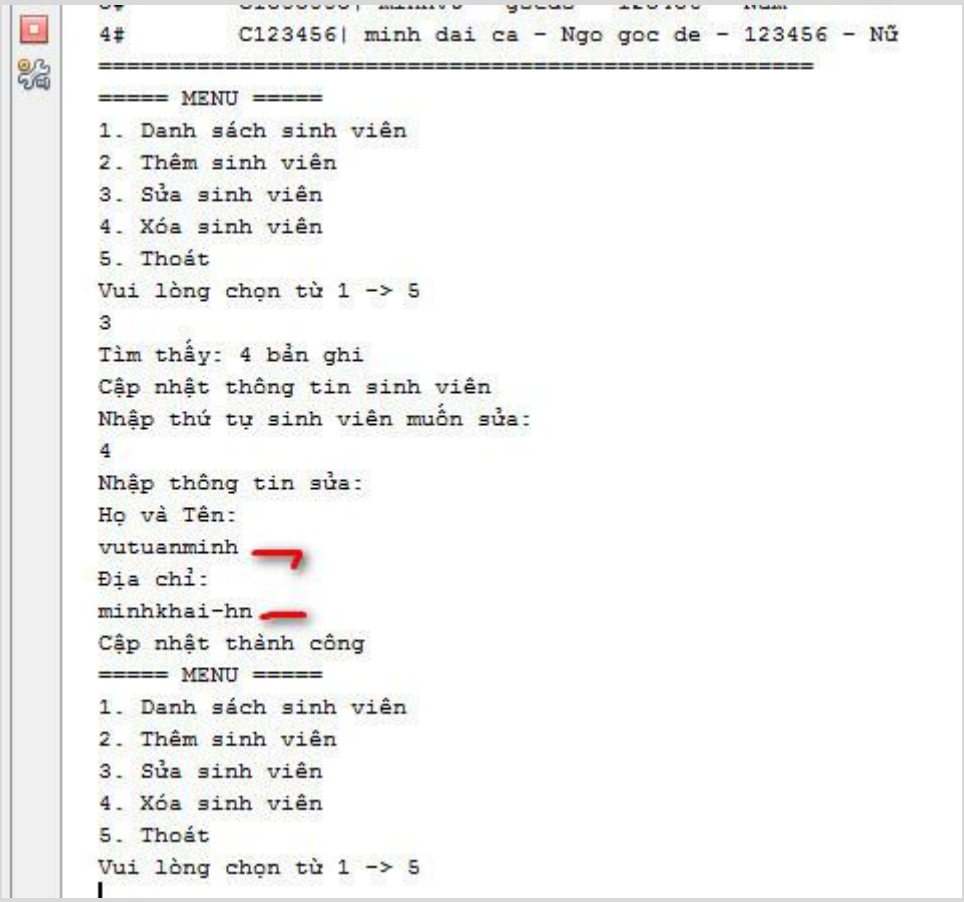
```
        rowset.acceptChanges(); // Thực thi thay đổi trên CSDL
    }

    private int showMenu() {
        System.out.println("===== MENU =====");
        System.out.println("1. Danh sách sinh viên");
        System.out.println("2. Thêm sinh viên");
        System.out.println("3. Sửa sinh viên");
        System.out.println("4. Xóa sinh viên");
        System.out.println("5. Thoát");
        System.out.println("Vui lòng chọn từ 1 -> 5");
        Scanner nhap = new Scanner(System.in);
        return nhap.nextInt();
    }

    /**
     * @param args the command line arguments
     * @throws java.sql.SQLException
     */
    public static void main(String[] args) throws SQLException {
        Connection conn = null;
        try {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            conn =
DriverManager.getConnection("jdbc:sqlserver://localhost:1433;databaseName=qlsv",
"sa", "1234567");
            conn.setAutoCommit(false); // Mặc định là true.
            System.out.println("Kết nối tới CSDL thành công");
        } catch (ClassNotFoundException | SQLException ex) {
            Logger.getLogger(MainClass.class.getName()).log(Level.SEVERE, null, ex);
        }
        // Khởi tạo interface từ lớp CachedRowSet;
        /*
         Cách 1: Tạo trực tiếp từ lớp CachedRowSetImpl
         Java đề xuất khả năng không sử dụng cách này trong tương lai
         */
//        CachedRowSet cachedRowset = new CachedRowSetImpl();
//        CachedRowSet cachedRowSet = new CachedRowSetImpl();

        // Cách 2: Sử dụng interface RowSetFactory
        RowSetFactory factory = RowSetProvider.newFactory();
        CachedRowSet cachedRowset = factory.createCachedRowSet();
        String sql = "SELECT * FROM tblsinhvien";
        cachedRowset.setCommand(sql);
        cachedRowset.execute(conn);
```

```
        DemoCachedRowset demo = new DemoCachedRowset();
        int choose;
        do {
            choose = demo.showMenu();
            switch (choose) {
                case 1:
                    demo.getAllRowset(cachedRowset);
                    break;
                case 2:
                    demo.insert(cachedRowset);
                    break;
                case 3:
                    demo.update(cachedRowset);
                    break;
                case 4:
                    demo.delete(cachedRowset);
                    break;
            }
        } while (choose > 0 && choose < 5);
        cachedRowset.close(); // Đóng kết nối tới CSDL
    }

}
```

Bước 4: Chạy chương trình và xem kết quả.

Lưu ý:

Hàm acceptChanges() được gọi khi cần thực thi thay đổi lên CSDL. Nếu chưa gọi thì dữ liệu sẽ không có bất cứ thay đổi gì như hình mô tả bên dưới.



Phần II - Bài tập tự làm

1. Viết chương trình quản lý hàng hóa gồm các thông tin sau:

1. Mã hàng hóa
2. Tên hàng
3. Nhà sản xuất
4. Số lượng (trong kho)
5. Giá gốc (VNĐ)
6. Thuế VAT

Menu của chương trình

========== MENU ==========

1. Nhập hàng
2. Xuất hàng
3. Kho

4. Cập nhật
5. Thoát

========== ***** ==========

Nhập hàng, cập nhật hàng yêu cầu sử dụng PrepareStatement để tạo truy vấn.

Tùy chọn "Kho" hiển thị danh sách các sản phẩm hiện có với toàn bộ thông tin.

Khi xuất hàng, phải trừ số lượng trong kho, nếu số lượng trong kho không đủ phải hiển thị cảnh báo và hủy giao dịch (sử dụng store procedure để kiểm soát).

2. Tạo ứng dụng quản lý nhà sách vận dụng các kiến thức nâng cao về JDBC (thực hiện lệnh theo lô, sử dụng PrepareStatement, StoreProcedure). Ứng dụng quản lý có các bảng với thông tin gợi ý như sau:

Nhà xuất bản

Tên

Địa chỉ

SĐT

Thể loại

Tên thể loại

Mô tả

Sách

Mã sách

Tên sách

Thể loại

NXB

Số trang

Giá tiền

3. Tạo ứng dụng consonle Java để quản lý sinh viên.

Yêu cầu 1:

Tạo database **dbQLSV**, bảng CSDL **tblSinhVien** gồm các trường sau:

1. int **id** - khóa chính, tăng dần.
2. varchar(8) rollNo - Mã sinh viên - Kiểu Unique
3. nvarchar(256) **sv_name** - tên sinh viên
4. nvarchar(521) **sv_address** - địa chỉ

5. varchar(11) **sv_phone** - Số điện thoại
6. varchar(64) **sv_email** - Email - Kiểu Unique

Yêu cầu 2:

Tạo menu như sau:

========= MENU ==========

1. Nhập dữ liệu sinh viên.
2. Hiển thị danh sách sinh viên.
3. Tìm kiếm sinh viên theo email
4. Tìm kiếm sinh viên theo tên
5. Sửa thông tin sinh viên theo mã Roll No.
6. Xóa thông tin sinh viên theo mã Roll No.
7. Thoát chương trình.

==========

Yêu cầu 3:

Sử dụng kiến thức về JDBC để tạo kết nối tới CSDL, nếu kết nối tới được thì hiển thị dưới menu dòng "**[Đã kết nối tới tới máy chủ]**", ngược lại phải hiển thị "**[Không thể kết nối tới máy chủ - vui lòng kiểm kết nối của bạn]**". Nếu ko kết nối được thì **không** cho thực hiện các chức năng trên menu.

Yêu cầu 4:

Thực hiện các chức năng của chương trình, thực hiện theo mô hình MVC chia 3 tầng:

1. Tầng 1: tạo các kết nối tới Database (Class.forname…., Connection….)
2. Tầng 2: datasource, xây dựng các hàm thực hiện các thao tác CRUD đối với các Entity như object SinhVien
3. Tầng 3: là tầng xử lý từ menu gọi tới và thao tác tới tầng 2 - datasource.

Yêu cầu 5:

Thực hiện tạo Store Procedure để giản tiện việc hardcode truy vấn SQL (bảo mật kém) và trông chương trình amater :D.