

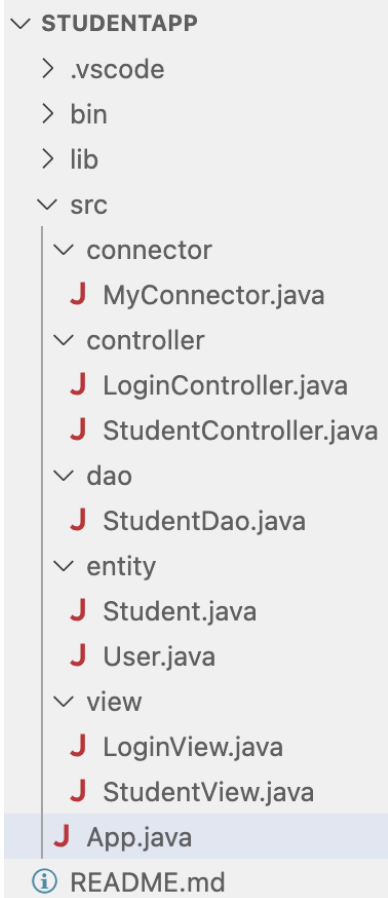
# Ứng dụng quản lý Sinh viên

**Đề bài:** Viết chương trình quản lý sinh viên trong Java , sử dụng Swing để tạo giao diện và áp dụng mô hình MVC, sử dụng cơ sở dữ liệu MySQL để lưu trữ thông tin. Mỗi đối tượng sinh viên có các thuộc tính sau: mssv, hoten, tuoi, diachi và dtb (điểm trung bình). Với các chức năng sau:

1. Sử dụng mô hình MVC.
2. Tạo màn hình đăng nhập.
3. Thêm mới một sinh viên.
4. Chỉnh sửa thông tin.
5. Xóa sinh viên
6. Sắp xếp sinh viên theo dtb.
7. Sắp xếp sinh viên theo tên.
8. Hiển thị danh sách sinh viên.

## Hướng dẫn

1. Tạo một project với cấu trúc của project trên vsCode như sau:



---

Tầng model bao gồm các package dao và entity.

- Lớp **User.java** để lưu thông tin người dùng.
- Lớp **Student.java** để lưu thông tin cho mỗi sinh viên.
- Lớp **StudentDao.java** chứa các phương thức quản lý sinh viên như thêm, sửa, xóa, sắp xếp, đọc, ghi sinh viên.

Tầng view bao gồm package view.

- Lớp **LoginView.java** tạo màn hình login.
- Lớp **StudentView.java** tạo màn hình quản lý sinh viên.

Tầng controler bao gồm package controler.

- Lớp **LoginController.java** xử lý các sự kiện từ LoginView.java.
- Lớp **StudentController.java** xử lý các sự kiện từ StudentView.java.

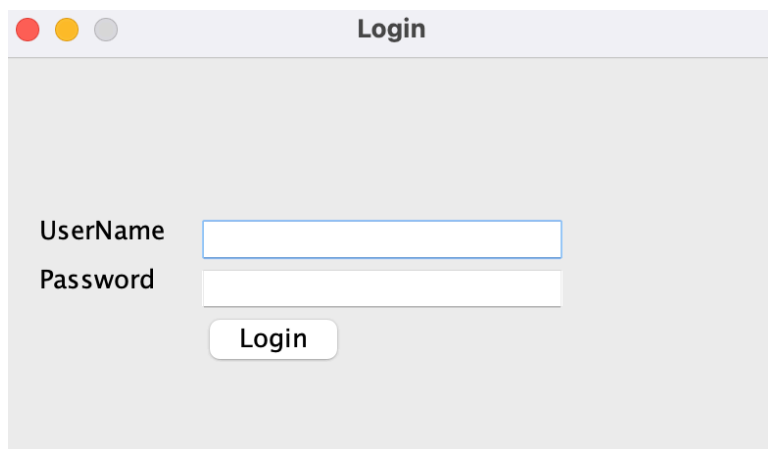
Package connector gồm lớp **MyConnector.java** chứa thông tin liên quan đến kết nối database.

Lớp **App.java** chứa hàm main để khởi chạy ứng dụng.

## 1. Tạo chức năng login

Tạo màn hình login chứa thông tin sau:

- Trường username.
- Trường password.
- Login button.



The image shows a Java Swing window titled "Login". It features a light gray background and a title bar with three standard window control buttons (red, yellow, and gray). The main content area contains two text input fields. The first field is labeled "UserName" and the second is labeled "Password". Below the password field is a button labeled "Login".

---

## Tạo lớp **User.java**

```
package entity;

public class User {
    private String userName;
    private String password;

    public User() {
    }

    public User(String userName, String password) {
        super();
        this.userName = userName;
        this.password = password;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

---

## Tạo lớp **LoginView.java**

```
package view;
import java.awt.event.*;
import javax.swing.*;

import entity.User;

public class LoginView extends JFrame implements ActionListener {
    private JLabel userNameLabel;
    private JLabel passwordlabel;
    private JPasswordField passwordField;
    private JTextField userNameField;
    private JButton loginBtn;

    public LoginView() {
        initComponents();
    }

    private void initComponents() {
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        userNameLabel = new JLabel("UserName");
        passwordlabel = new JLabel("Password");
        userNameField = new JTextField(15);
        passwordField = new JPasswordField(15);
        loginBtn = new JButton();

        loginBtn.setText("Login");
        loginBtn.addActionListener(this);

        // tạo spring layout
        SpringLayout layout = new SpringLayout();
        JPanel panel = new JPanel();

        // tạo đối tượng panel để chứa các thành phần của màn hình login
        panel.setSize(400, 300);
        panel.setLayout(layout);
        panel.add(userNameLabel);
        panel.add(passwordlabel);
        panel.add(userNameField);
        panel.add(passwordField);
        panel.add(loginBtn);
    }
}
```

```

// cài đặt vị trí các thành phần trên màn hình login
String WEST = SpringLayout.WEST;
String NORTH = SpringLayout.NORTH;

layout.putConstraint(WEST, userNameLabel, 20, WEST, panel);
layout.putConstraint(NORTH, userNameLabel, 80, NORTH, panel);
layout.putConstraint(WEST, passwordlabel, 20, WEST, panel);
layout.putConstraint(NORTH, passwordlabel, 105, NORTH, panel);
layout.putConstraint(WEST, userNameField, 80, WEST, userNameLabel);
layout.putConstraint(NORTH, userNameField, 80, NORTH, panel);
layout.putConstraint(WEST, passwordField, 80, WEST, passwordlabel);
layout.putConstraint(NORTH, passwordField, 105, NORTH, panel);
layout.putConstraint(WEST, loginBtn, 80, WEST, passwordlabel);
layout.putConstraint(NORTH, loginBtn, 130, NORTH, panel);

// add panel tới JFrame
this.add(panel);
this.pack();

// cài đặt các thuộc tính cho JFrame
this.setTitle("Login");
this.setSize(400, 300);
this.setResizable(false);
}

public void showMessage(String message) {
    JOptionPane.showMessageDialog(this, message);
}

public User getUser() {
    return new User(userNameField.getText(),
        String.valueOf(passwordField.getPassword()));
}

public void actionPerformed(ActionEvent e) {
}

public void addLoginListener(ActionListener listener) {
    loginBtn.addActionListener(listener);
}
}

```

---

## Tạo lớp **LoginController.java**

```
package controller;
import java.awt.event.*;
import entity.User;
import view.LoginView;
import view.StudentView;

public class LoginController {
    private LoginView loginView;
    private StudentView studentView;

    public LoginController(LoginView view) {
        this.loginView = view;
        view.addLoginListener(new LoginListener());
    }

    public void showLoginView() {
        loginView.setVisible(true);
    }

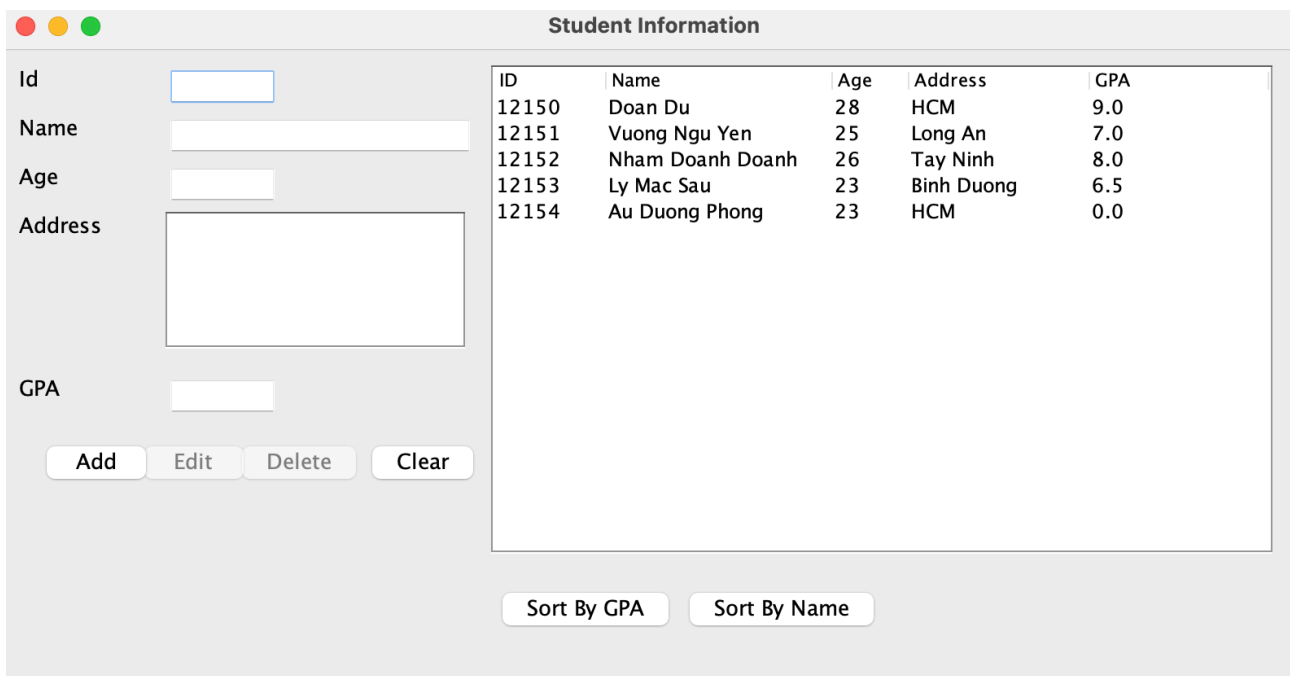
    class LoginListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            User user = loginView.getUser();
            if (checkUser(user)) {
                studentView = new StudentView();
                StudentController sc = new StudentController(studentView);
                sc.showStudentView();
                loginView.setVisible(false);
            } else {
                loginView.showMessageDialog("Username or password are incorrect!");
            }
        }

        public boolean checkUser(User user) {
            if (user != null) {
                if ("admin".equals(user.getUserName()) && "admin".equals(user.
                    getPassword()))
                    return true;
            }
            return false;
        }
    }
}
```

## 2. Tạo chức năng quản lý sinh viên

Tạo màn hình quản lý sinh viên chứa các thông tin sau:

- Các trường tương ứng với các thuộc tính của sinh viên.
- Button Add.
- Button Edit.
- Button Delete.
- Button Clear.
- Bảng hiển thị danh sách sinh viên.
- Button “Sort By Name”
- Button “Sort By GPA”



ID	Name	Age	Address	GPA
12150	Doan Du	28	HCM	9.0
12151	Vuong Ngu Yen	25	Long An	7.0
12152	Nham Doanh Doanh	26	Tay Ninh	8.0
12153	Ly Mac Sau	23	Binh Duong	6.5
12154	Au Duong Phong	23	HCM	0.0

---

## Tạo lớp **Student.java**

```
package entity;

public class Student {
    private int id, age;
    private String name, address;
    private float gpa;

    public Student() {

    }

    public Student(int id, String name, int age, String address, float gpa) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.address = address;
        this.gpa = gpa;
    }

    public int getId() {return id;}

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

    public String getName() {return name;}

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

    public int getAge() {return age;}

    public void setAge(int age) {this.age = age;}

    public String getAddress() {return address;}

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

    public float getGpa() {return gpa;}

    public void setGpa(float gpa) {this.gpa = gpa;}
}
```



---

## Tạo lớp **StudentDao.java**

```
package dao;

import java.sql.*;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

import connector.MyConnector;
import entity.Student;

public class StudentDao {

    private List<Student> listStudents;

    public StudentDao() {
        this.listStudents = readListStudents();
        if (listStudents == null) {
            listStudents = new ArrayList<Student>();
        }
    }

    // Đọc danh sách sinh viên từ Database
    public List<Student> readListStudents() {

        List<Student> list = new ArrayList<Student>();

        MyConnector myConnector = new MyConnector();
        myConnector.initConnector();

        try {
            Statement statement = myConnector.connection.createStatement();
            ResultSet resultSet = statement.executeQuery("SELECT * FROM SINHVIEN");

            while(resultSet.next()){
                int id = resultSet.getInt("MSSV");
                String name = resultSet.getString("TEN");
                int age = resultSet.getInt("TUOI");
                String addr = resultSet.getString("DIACHI");
                float gpa = resultSet.getFloat("GPA");
            }
        }
    }
}
```

```

        Student student = new Student(id, name, age, addr,gpa);
        list.add(student);
    }

    myConnector.connection.close();

} catch (SQLException e) {
    e.printStackTrace();
}
return list;
}

// thêm sinh viên vào Database
public boolean add(Student student) {
    boolean flag = false;

    MyConnector myConnector = new MyConnector();
    myConnector.initConnector();

    try {
        PreparedStatement statement =
            myConnector.connection.prepareStatement("INSERT INTO SINHVIEN(MSSV, TEN,
                TUOI, DIACHI, GPA) VALUES (?, ?, ?, ?, ?)");

        statement.setInt(1,student.getId());
        statement.setString(2,student.getName());
        statement.setInt(3,student.getAge());
        statement.setString(4,student.getAddress());
        statement.setFloat(5,student.getGpa());

        statement.executeUpdate();

        myConnector.connection.close();

        listStudents.add(student);

        flag = true;
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return flag;
}

```

```

// Chỉnh sửa thông tin sinh viên
public boolean edit(Student student) {
    boolean flag = false;
    int size = listStudents.size();
    for (int i = 0; i < size; i++) {
        if (listStudents.get(i).getId() == student.getId()) {

            MyConnector myConnector = new MyConnector();
            myConnector.initConnector();

            try {
                PreparedStatement statement =
                    myConnector.connection.prepareStatement("UPDATE SINHVIEN SET TEN =
                        ?, TUOI = ?, DIACHI = ?, GPA = ? WHERE MSSV = ?;");

                statement.setString(1, student.getName());
                statement.setInt(2, student.getAge());
                statement.setString(3, student.getAddress());
                statement.setFloat(4, student.getGpa());
                statement.setInt(5, student.getId());

                statement.executeUpdate();

                myConnector.connection.close();

                listStudents.get(i).setName(student.getName());
                listStudents.get(i).setAge(student.getAge());
                listStudents.get(i).setAddress(student.getAddress());
                listStudents.get(i).setGpa(student.getGpa());

                flag = true;
            } catch (SQLException e) {
                e.printStackTrace();
            }
            break;
        }
    }

    return flag;
}

```

```

// xóa sinh viên ra khỏi Database
public boolean delete(Student student) {
    boolean isFound = false;

    int size = listStudents.size();
    for (int i = 0; i < size; i++) {
        if (listStudents.get(i).getId() == student.getId()) {
            student = listStudents.get(i);
            isFound = true;
            break;
        }
    }
    if (isFound) {

        MyConnector myConnector = new MyConnector();
        myConnector.initConnector();

        try {
            PreparedStatement statement =
                myConnector.connection.prepareStatement("DELETE FROM SINHVIEN WHERE
                    MSSV = ?;");

            statement.setInt(1, student.getId());

            statement.executeUpdate();

            myConnector.connection.close();

            listStudents.remove(student);

        } catch (SQLException e) {
            e.printStackTrace();
        }
        return true;
    }
    return false;
}

```

```

// sắp xếp danh sách sinh viên theo GPA theo tứ tự tăng dần
public void sortStudentByGPA() {
    Collections.sort(listStudents, new Comparator<Student>() {
        public int compare(Student student1, Student student2) {
            if (student1.getGpa() > student2.getGpa()) {
                return 1;
            }
            return -1;
        }
    });

    //...
}

// sắp xếp danh sách sinh viên theo tên tăng dần
public void sortStudentByName() {
    Collections.sort(listStudents, new Comparator<Student>() {
        public int compare(Student student1, Student student2) {
            return student1.getName().compareTo(student2.getName());
        }
    });

    //...
}

public List<Student> getListStudents() {
    return listStudents;
}

public void setListStudents(List<Student> listStudents) {
    this.listStudents = listStudents;
}
}

```

---

## Tạo lớp **StudentView.java**

```
package view;
import java.util.List;
import java.awt.Dimension;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
import javax.swing.table.DefaultTableModel;
import entity.Student;

public class StudentView extends JFrame implements ActionListener,
    ListSelectionListener {
    private static final long serialVersionUID = 1L;
    private JButton addStudentBtn;
    private JButton editStudentBtn;
    private JButton deleteStudentBtn;
    private JButton clearBtn;
    private JButton sortStudentGPABtn;
    private JButton sortStudentNameBtn;
    private JScrollPane jScrollPaneStudentTable;
    private JScrollPane jScrollPaneAddress;
    private JTable studentTable;

    private JLabel idLabel;
    private JLabel nameLabel;
    private JLabel ageLabel;
    private JLabel addressLabel;
    private JLabel gpaLabel;

    private JTextField idField;
    private JTextField nameField;
    private JTextField ageField;
    private JTextArea addressTA;
    private JTextField gpaField;

    // định nghĩa các cột của bảng student
    private String [] columnNames = new String [] {
        "ID", "Name", "Age", "Address", "GPA"};
    // định nghĩa dữ liệu mặc định của bảng student là rỗng
    private Object data = new Object [][] {};
```

```

public StudentView() {
    initComponents();
}

private void initComponents() {
    setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    // khởi tạo các phím chức năng
    addStudentBtn = new JButton("Add");
    editStudentBtn = new JButton("Edit");
    deleteStudentBtn = new JButton("Delete");
    clearBtn = new JButton("Clear");
    sortStudentGPABtn = new JButton("Sort By GPA");
    sortStudentNameBtn = new JButton("Sort By Name");
    // khởi tạo bảng student
    jScrollPaneStudentTable = new JScrollPane();
    studentTable = new JTable();

    // khởi tạo các label
    idLabel = new JLabel("Id");
    nameLabel = new JLabel("Name");
    ageLabel = new JLabel("Age");
    addressLabel = new JLabel("Address");
    gpaLabel = new JLabel("GPA");

    // khởi tạo các trường nhập dữ liệu cho student
    idField = new JTextField(5);
    nameField = new JTextField(15);
    ageField = new JTextField(5);
    addressTA = new JTextArea();
    addressTA.setColumns(15);
    addressTA.setRows(5);
    jScrollPaneAddress = new JScrollPane();
    jScrollPaneAddress.setViewportViewView(addressTA);
    gpaField = new JTextField(5);

    // cài đặt các cột và data cho bảng student
    studentTable.setModel(new DefaultTableModel((Object[][] data, columnNames));
    jScrollPaneStudentTable.setViewportViewView(studentTable);
    jScrollPaneStudentTable.setPreferredSize(new Dimension (480, 300));

    // tạo spring layout
    SpringLayout layout = new SpringLayout();

```

```

// tạo đối tượng panel để chứa các thành phần của màn hình quản lý Student
JPanel panel = new JPanel();
panel.setSize(800, 420);
panel.setLayout(layout);
panel.add(jScrollPaneStudentTable);

panel.add(addStudentBtn);
panel.add(editStudentBtn);
panel.add(deleteStudentBtn);
panel.add(clearBtn);
panel.add(sortStudentGPABtn);
panel.add(sortStudentNameBtn);

panel.add(idLabel);
panel.add(nameLabel);
panel.add(ageLabel);
panel.add(addressLabel);
panel.add(gpaLabel);

panel.add(idField);
panel.add(nameField);
panel.add(ageField);
panel.add(jScrollPaneAddress);
panel.add(gpaField);

// cài đặt vị trí các thành phần trên màn hình login
String WEST = SpringLayout.WEST;
String NORTH = SpringLayout.NORTH;

layout.putConstraint(WEST, idLabel, 10, WEST, panel);
layout.putConstraint(NORTH, idLabel, 10, NORTH, panel);
layout.putConstraint(WEST, nameLabel, 10, WEST, panel);
layout.putConstraint(NORTH, nameLabel, 40, NORTH, panel);
layout.putConstraint(WEST, ageLabel, 10, WEST, panel);
layout.putConstraint(NORTH, ageLabel, 70, NORTH, panel);
layout.putConstraint(WEST, addressLabel, 10, WEST, panel);
layout.putConstraint(NORTH, addressLabel, 100, NORTH, panel);
layout.putConstraint(WEST, gpaLabel, 10, WEST, panel);
layout.putConstraint(NORTH, gpaLabel, 200, NORTH, panel);

layout.putConstraint(WEST, idField, 100, WEST, panel);
layout.putConstraint(NORTH, idField, 10, NORTH, panel);
layout.putConstraint(WEST, nameField, 100, WEST, panel);

```



```

        layout.putConstraint(NORTH, nameField, 40, NORTH, panel);
        layout.putConstraint(WEST, ageField, 100, WEST, panel);
        layout.putConstraint(NORTH, ageField, 70, NORTH, panel);
        layout.putConstraint(WEST, jScrollPaneAddress, 100, WEST, panel);
        layout.putConstraint(NORTH, jScrollPaneAddress, 100, NORTH, panel);
        layout.putConstraint(WEST, gpaField, 100, WEST, panel);
        layout.putConstraint(NORTH, gpaField, 200, NORTH, panel);

        layout.putConstraint(WEST, jScrollPaneStudentTable, 300, WEST, panel);
        layout.putConstraint(NORTH, jScrollPaneStudentTable, 10, NORTH, panel);

        layout.putConstraint(WEST, addStudentBtn, 20, WEST, panel);
        layout.putConstraint(NORTH, addStudentBtn, 240, NORTH, panel);
        layout.putConstraint(WEST, editStudentBtn, 60, WEST, addStudentBtn);
        layout.putConstraint(NORTH, editStudentBtn, 240, NORTH, panel);
        layout.putConstraint(WEST, deleteStudentBtn, 60, WEST, editStudentBtn);

        layout.putConstraint(NORTH, clearBtn, 240, NORTH, panel);
        layout.putConstraint(WEST, clearBtn, 80, WEST, deleteStudentBtn);

        layout.putConstraint(NORTH, deleteStudentBtn, 240, NORTH, panel);
        layout.putConstraint(WEST, sortStudentGPABtn, 300, WEST, panel);
        layout.putConstraint(NORTH, sortStudentGPABtn, 330, NORTH, panel);
        layout.putConstraint(WEST, sortStudentNameBtn, 115, WEST, sortStudentGPABtn);
        layout.putConstraint(NORTH, sortStudentNameBtn, 330, NORTH, panel);

        this.add(panel);
        this.pack();
        this.setTitle("Student Information");
        this.setSize(800, 420);

        // disable Edit and Delete buttons
        editStudentBtn.setEnabled(false);
        deleteStudentBtn.setEnabled(false);

        // enable Add button
        addStudentBtn.setEnabled(true);
    }

    public void showMessage(String message) {
        JOptionPane.showMessageDialog(this, message);
    }

```

```

// hiển thị danh sách sinh viên vào bảng studentTable
public void showListStudents(List<Student> list) {
    int size = list.size();
    // tạo một mảng 2 chiều với số dòng là số lượng sinh viên (size)
    // và số cột là 5.
    Object [][] students = new Object[size][5];
    for (int i = 0; i < size; i++) {
        students[i][0] = list.get(i).getId();
        students[i][1] = list.get(i).getName();
        students[i][2] = list.get(i).getAge();
        students[i][3] = list.get(i).getAddress();
        students[i][4] = list.get(i).getGpa();
    }
    studentTable.setModel(new DefaultTableModel(students, columnNames));
}

// điền thông tin của hàng được chọn từ bảng sinh viên
// vào các trường tương ứng của sinh viên.
public void fillStudentFromSelectedRow() {
    // lấy chỉ số của hàng được chọn
    int row = studentTable.getSelectedRow();
    if (row >= 0) {
        idField.setText(studentTable.getModel().getValueAt(row, 0).toString());
        nameField.setText(studentTable.getModel().getValueAt(row, 1).toString());
        ageField.setText(studentTable.getModel().getValueAt(row, 2).toString());
        addressTA.setText(studentTable.getModel().getValueAt(row, 3).toString());
        gpaField.setText(studentTable.getModel().getValueAt(row, 4).toString());

        // enable Edit and Delete buttons
        editStudentBtn.setEnabled(true);
        deleteStudentBtn.setEnabled(true);

        // disable Add button
        addStudentBtn.setEnabled(false);
    }
}

// xóa thông tin sinh viên
public void clearStudentInfo() {
    idField.setText("");
    nameField.setText("");
    ageField.setText("");
    addressTA.setText("");
    gpaField.setText("");
}

```

```

        // disable Edit and Delete buttons
        editStudentBtn.setEnabled(false);
        deleteStudentBtn.setEnabled(false);

        // enable Add button
        addStudentBtn.setEnabled(true);
    }

    // hiển thị thông tin sinh viên
    public void showStudent(Student student) {
        idField.setText("" + student.getId());
        nameField.setText(student.getName());
        ageField.setText("" + student.getAge());
        addressTA.setText(student.getAddress());
        gpaField.setText("" + student.getGpa());

        // enable Edit and Delete buttons
        editStudentBtn.setEnabled(true);
        deleteStudentBtn.setEnabled(true);

        // disable Add button
        addStudentBtn.setEnabled(false);
    }

    // lấy thông tin sinh viên
    public Student getStudentInfo() {
        // validate student
        if (!validateId() || !validateName() || !validateAge() || !validateAddress()
            || !validateGPA())
            return null;
        try {
            Student student = new Student();
            student.setId(Integer.parseInt(idField.getText()));
            student.setName(nameField.getText().trim());
            student.setAge(Byte.parseByte(ageField.getText().trim()));
            student.setAddress(addressTA.getText().trim());
            student.setGpa(Float.parseFloat(gpaField.getText().trim()));
            return student;
        } catch (Exception e) {showMessage(e.getMessage());}
        return null;
    }
}

```

```
private boolean validateId() {
    String id = idField.getText();
    if (id == null || "".equals(id.trim())) {
        idField.requestFocus();
        showMessage("Id cannot be empty!");
        return false;
    }
    return true;
}

private boolean validateName() {
    String name = nameField.getText();
    if (name == null || "".equals(name.trim())) {
        nameField.requestFocus();
        showMessage("Name cannot be empty!");
        return false;
    }
    return true;
}

private boolean validateAddress() {
    String address = addressTA.getText();
    if (address == null || "".equals(address.trim())) {
        addressTA.requestFocus();
        showMessage("Address cannot be empty!");
        return false;
    }
    return true;
}

private boolean validateAge() {
    try {
        Byte age = Byte.parseByte(ageField.getText().trim());
        if (age < 0 || age > 100) {
            ageField.requestFocus();
            showMessage("Age wrong format! It should be between 0 and 100.");
            return false;
        }
    } catch (Exception e) {
        ageField.requestFocus();
        showMessage("Age wrong format!");
        return false;
    }
    return true;
}
```

```

private boolean validateGPA() {
    try {
        Float gpa = Float.parseFloat(gpaField.getText().trim());
        if (gpa < 0 || gpa > 10) {
            gpaField.requestFocus();
            showMessage("GPA wrong format! It should be between 0 and 10.");
            return false;
        }
    } catch (Exception e) {
        gpaField.requestFocus();
        showMessage("GPA wrong format!");
        return false;
    }
    return true;
}

public void actionPerformed(ActionEvent e) {
}

public void valueChanged(ListSelectionEvent e) {
}

public void addAddStudentListener(ActionListener listener) {
    addStudentBtn.addActionListener(listener);
}

public void addEdiStudentListener(ActionListener listener) {
    editStudentBtn.addActionListener(listener);
}

public void addDeleteStudentListener(ActionListener listener) {
    deleteStudentBtn.addActionListener(listener);
}

public void addClearListener(ActionListener listener) {
    clearBtn.addActionListener(listener);
}

public void addSortStudentGPAListener(ActionListener listener) {
    sortStudentGPABtn.addActionListener(listener);
}

public void addSortStudentNameListener(ActionListener listener) {
    sortStudentNameBtn.addActionListener(listener);
}

public void addListStudentSelectionListener(ListSelectionListener listener) {
    studentTable.getSelectionModel().addListSelectionListener(listener);
}
}

```

---

## Tạo lớp **StudentController.java**

```
package controller;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
import dao.StudentDao;
import entity.Student;
import view.StudentView;

public class StudentController {
    private StudentDao studentDao;
    private StudentView studentView;

    public StudentController(StudentView view) {
        this.studentView = view;
        studentDao = new StudentDao();
        view.addAddStudentListener(new AddStudentListener());
        view.addEditStudentListener(new EditStudentListener());
        view.addDeleteStudentListener(new DeleteStudentListener());
        view.addClearListener(new ClearStudentListener());
        view.addSortStudentGPAListener(new SortStudentGPAListener());
        view.addSortStudentNameListener(new SortStudentNameListener());
        view.addListStudentSelectionListener(new ListStudentSelectionListener());
    }

    public void showStudentView() {
        List<Student> studentList = studentDao.getListStudents();
        studentView.setVisible(true);
        studentView.showListStudents(studentList);
    }

    // Cài đặt sự kiện click button "Add"
    class AddStudentListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            Student student = studentView.getStudentInfo();
            if (student != null) {
                if(studentDao.add(student)){
                    studentView.showStudent(student);
                    studentView.showListStudents(studentDao.getListStudents());
                }
            }
        }
    }
}
```

```

        studentView.showMessage("Add successfully!");
    }else
        studentView.showMessage("Cannot add!");
    }
}

// Cài đặt sự kiện click button "Edit"
class EditStudentListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        Student student = studentView.getStudentInfo();
        if (student != null) {
            if(studentDao.edit(student)){
                studentView.showStudent(student);
                studentView.showListStudents(studentDao.getListStudents());
                studentView.showMessage("Update successfully!");
            }else
                studentView.showMessage("Cannot update!");
        }
    }
}

// Cài đặt sự kiện click button "Delete"
class DeleteStudentListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        Student student = studentView.getStudentInfo();
        if (student != null) {
            if(studentDao.delete(student)){
                studentView.clearStudentInfo();
                studentView.showListStudents(studentDao.getListStudents());
                studentView.showMessage("Remove successfully!");
            }else
                studentView.showMessage("Cannot remove!");
        }
    }
}

// Cài đặt sự kiện click button "Clear"
class ClearStudentListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        studentView.clearStudentInfo();
    }
}

```

```

// Cài đặt sự kiện click button "Sort By GPA"
class SortStudentGPAListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        studentDao.sortStudentByGPA();
        studentView.showListStudents(studentDao.getListStudents());
    }
}

// Cài đặt sự kiện click button "Sort By Name"
class SortStudentNameListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        studentDao.sortStudentByName();
        studentView.showListStudents(studentDao.getListStudents());
    }
}

//Cài đặt sự kiện chọn sinh viên trong bảng sinh viên
class ListStudentSelectionListener implements ListSelectionListener {
    public void valueChanged(ListSelectionEvent e) {
        studentView.fillStudentFromSelectedRow();
    }
}
}

```



---

## Tạo lớp **MyConnector.java**

```
package connector;
import java.sql.*;

public class MyConnector {

    // Tạo chuỗi kết nối JDBC
    private String jdbcUrl = "jdbc:mysql://localhost:3306/testDB";

    public Connection connection = null;

    public void initConnector(){
        try {
            connection = DriverManager.getConnection(jdbcUrl, "root", "123456");
        } catch (SQLException e) {e.printStackTrace();}
    }
}
```

## Tạo lớp **App.java**

```
import java.awt.EventQueue;
import javax.swing.JFrame;
import controller.LoginController;
import view.LoginView;

public class App extends JFrame {
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                LoginView view = new LoginView();
                LoginController controller = new LoginController(view);
                // hiển thị màn hình login
                controller.showLoginView();
            }
        });
    }
}
```