Shape, square

Description automatically generated

|  |  |  |
| --- | --- | --- |
| Giáo viên | : | Vũ Huấn |
| Sinh viên | : | Nhữ Đình Đức |
| Mã sinh viên | : | 211202522 |

***Đề Tài : Phần mềm quản lý sinh viên đơn giản***

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**UNIVERSITY OF TRANSPOST AND COMMUNICATIONS**

**NHÓM 39**

**BỘ MÔN CÔNG NGHỆ JAVA**

**LỚP CNTT6\_62 – N08**

*Hà Nội, tháng 05 năm 2023*

**Mục Lục**

Table of Contents

[**LỜI NÓI ĐẦU 3**](#_Toc134000319)

[**1. Tổng quan về đề tài 4**](#_Toc134000320)

[***1.1 Đề bài* 4**](#_Toc134000321)

[***1.2 Yêu cầu cơ bản* 4**](#_Toc134000322)

[***1.3 Sơ lược về khung chương trình* 4**](#_Toc134000323)

[**2. Cài đặt chương trình 6**](#_Toc134000324)

[**2.1 file pom 6**](#_Toc134000325)

[**2.2 Tạo chức năng đăng nhập 7**](#_Toc134000326)

[***2.2.1 Tạo lớp User.java* 7**](#_Toc134000327)

[***2.2.2 Tạo lớp UserSolution.java* 8**](#_Toc134000328)

[***2.2.3 Tạo lớp LoginView.java* 10**](#_Toc134000329)

[***2.2.4 Tạo lớp LoginController.java* 12**](#_Toc134000330)

[**2.3 Tạo chức năng quản lý sinh viên 13**](#_Toc134000331)

[***2.3.1 Tạo lớp Student.java* 13**](#_Toc134000332)

[***2.3.2 Tạo lớp StudentXML.java* 16**](#_Toc134000333)

[***2.3.3 Tạo lớp StudentSolution.java* 16**](#_Toc134000334)

[***2.3.4 Tạo lớp StudentView.java* 19**](#_Toc134000335)

[***2.5.6 Tạo lớp FileUtils.java* 35**](#_Toc134000336)

[***2.3.7 Tạo lớp App.java* 36**](#_Toc134000337)

[**2.4 Kết nối cơ sở dữ liệu (nếu có) 36**](#_Toc134000338)

[***2.4.1 Tạo file JDBCConnection.java* 36**](#_Toc134000339)

[***2.4.2 Tạo file StudentDao.java* 37**](#_Toc134000340)

[***2.4.3 Tạo lớp UserDao.java* 38**](#_Toc134000341)

[***2.4.4 Tạo lớp InsertStatement.java* 39**](#_Toc134000342)

[**3. Giao diện 40**](#_Toc134000343)

[**TÀI LIỆU THAM KHẢO 42**](#_Toc134000344)

# LỜI NÓI ĐẦU

Phần mềm quản lý sinh viên là một ứng dụng quan trọng trong lĩnh vực giáo dục. Nó giúp cho các trường đại học, cao đẳng và trung học phổ thông quản lý thông tin sinh viên một cách hiệu quả. Quản lý sinh viên bao gồm nhiều công việc như lưu trữ thông tin sinh viên, điểm số, khóa học, thời khóa biểu, tiền học phí, các thông tin về tình trạng học tập, hoạt động sinh viên và nhiều thông tin khác.

Tuy nhiên, việc quản lý sinh viên thủ công có thể dẫn đến những vấn đề như sai sót, mất thông tin, thất thoát hồ sơ và tiêu tốn nhiều thời gian và công sức. Đó là lý do tại sao phần mềm quản lý sinh viên trở thành một giải pháp tối ưu cho các trường giáo dục. Phần mềm quản lý sinh viên có nhiều lợi ích cho người quản lý cũng như sinh viên. Đầu tiên, nó giúp người quản lý tiết kiệm thời gian và tăng tính hiệu quả trong công việc. Thông qua phần mềm, thông tin sinh viên được lưu trữ một cách chính xác và có thể truy xuất nhanh chóng. Ngoài ra, phần mềm quản lý sinh viên còn giúp cho việc quản lý học phí, kế hoạch học tập, thông tin sinh viên dễ dàng hơn. Đối với sinh viên, phần mềm quản lý sinh viên cung cấp cho họ nhiều thông tin quan trọng như thời khóa biểu, lịch thi, điểm số, học phí, kế hoạch học tập. Nhờ đó, sinh viên có thể dễ dàng quản lý được thời gian và hoạt động học tập của mình một cách hiệu quả hơn. Trong quá trình phát triển phần mềm quản lý sinh viên, việc thiết kế giao diện, tối ưu hóa cơ sở dữ liệu và cải tiến tính năng là rất quan trọng. Tuy nhiên, điều quan trọng nhất vẫn là đảm bảo tính bảo mật và an toàn cho thông tin sinh viên. Đó cũng là lý do tại sao việc triển khai một hệ thống bảo mật hiệu quả cần được chú trọng.

Dưới đây, tôi sẽ hướng dẫn các bạn tạo một phần mềm ở mức đơn giản để quản lý sinh viên sử dụng ngôn ngữ hệ thống là java.

# 1. Tổng quan về đề tài

## ***1.1 Đề bài***

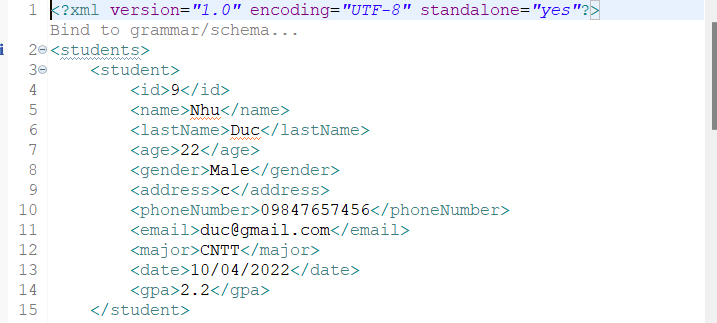
Viết chương trình đơn giản cho phần mềm **quản lý sinh viên trong Java**, sử dụng java Swing để tạo giao diện và áp dụng mô hình MVC. Trong đó, phần mềm chứa những chức năng cơ bản nhất.

## ***1.2 Yêu cầu cơ bản***

Một số yêu cầu cơ bản sau :

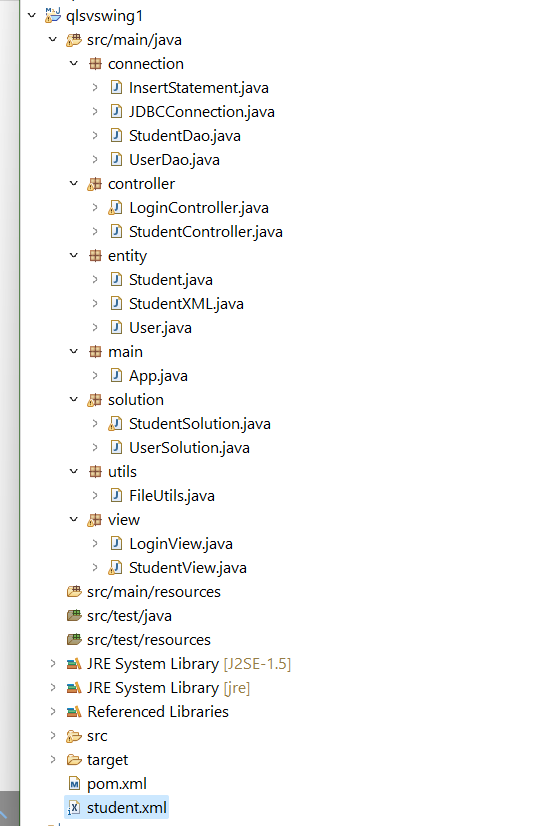
1. Sử dụng mô hình MVC
2. Tạo mà hình đăng nhập
3. Có chức năng cơ bản như thêm, sửa, xóa, sắp xép
4. Hiển thị danh sách sinh viên
5. Đọc và ghi sinh viên ra file (.xml)
6. Javacore cơ bản
7. Giao diện đơn giản với swing
8. Dùng cơ sở dữ liệu đi kèm (nếu có)

Trong đó sinh viên được lưu vào file **"student.xml"** với định dạng xml. Ví dụ:



## ***1.3 Sơ lược về khung chương trình***

Tạo maven project với cấu trúc sau :



Tầng M (model) bao gồm package ***solution*** và  ***entity* :**

* Lớp **User.java** để lưu thông tin người dùng.
* Lớp **UserSolution.java** chứa phương thức checkUser() để kiểm tra thông tin đang nhập.
* Lớp **Student.java** để lưu thông tin cho mỗi sinh viên.
* Lớp **StudentXML.java** để lưu thông tin danh sách sinh viên với định dạng XML vào file student.xml.
* Lớp **StudentSolution.java** chứa các phương thức quản lý sinh viên như thêm, sửa, xóa, sắp xếp, đọc, ghi sinh viên.

Tầng V (view) bao gồm package ***view***

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

Tầng C (controller) bao gồm package ***controller***

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

Các file khác:

* Lớp **FileUtils.java** được sử dụng để đọc ghi file.
* Lớp **App.java** chứa hàm main để khởi chạy ứng dụng.
* File **student.xml** được sử dụng để lưu danh sách sinh viên.
* Package **Connection** chứa các file về csdl

# 2. Cài đặt chương trình

## **2.1 file pom**

File pom.xml là một file cấu hình trong Maven, một công cụ quản lý dự án phát triển phần mềm trong Java. File này chứa thông tin về các thư viện phụ thuộc, plugin và cấu hình của dự án.

Các thông tin trong file pom.xml có thể được sử dụng để định cấu hình các khía cạnh khác nhau của dự án, bao gồm:

- Thông tin cơ bản của dự án, bao gồm tên, mô tả, phiên bản, tác giả và giấy phép.

- Thông tin về các phụ thuộc của dự án, bao gồm các thư viện, module hoặc các tài nguyên khác mà dự án cần phải sử dụng.

- Cấu hình của các plugin được sử dụng trong quá trình phát triển dự án, bao gồm các plugin để biên dịch mã nguồn, đóng gói ứng dụng, triển khai ứng dụng và các plugin khác.

- Cấu hình cho các khía cạnh khác của dự án, bao gồm các thiết lập cho định dạng mã nguồn, kiểm tra đơn vị, báo cáo, tạo bản sao lưu và nhiều hơn nữa.\

Trong bài tập này ,ta thêm các thư viện sau vào file pom.xml

* **jaxb-api-2.3.0.jar** : chuyển đối tượng thành xml và lưu vào file, đọc file và chuyển xml thành đối tượng.

Cụ thể :

<project *xmlns*="http://maven.apache.org/POM/4.0.0"

*xmlns:xsi*="http://www.w3.org/2001/XMLSchema-instance"

*xsi:schemaLocation*="http://maven.apache.org/POM/4.0.0

    http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>duc</groupId>

    <artifactId>qlsvswing1</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <packaging>jar</packaging>

    <name>qlsv</name>

    <url>http://maven.apache.org</url>

    <properties>

        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

    </properties>

    <dependencies>

        <!-- JAXB API -->

        <dependency>

            <groupId>javax.xml.bind</groupId>

            <artifactId>jaxb-api</artifactId>

            <version>2.3.1</version>

        </dependency>

        <!-- JAXB RI -->

        <dependency>

            <groupId>com.sun.xml.bind</groupId>

            <artifactId>jaxb-impl</artifactId>

            <version>2.3.4</version>

        </dependency>

        <dependency>

            <groupId>junit</groupId>

            <artifactId>junit</artifactId>

            <version>3.8.1</version>

            <scope>test</scope>

        </dependency>

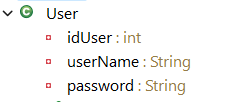
    </dependencies>

</project>

## **2.2 Tạo chức năng đăng nhập**

### ***2.2.1 Tạo lớp User.java***

Lớp có các thuộc tính sau :



package entity;

public class User {

    private *int* idUser;

    private *String* userName;

    private *String* password;

    public User() {

    }

    /\*\*

     \* @param *idUser*

     \* @param *userName*

     \* @param *password*

     \*/

    public User( *String* *userName*, *String* *password*) {

*super*();

*this*.userName = userName;

*this*.password = password;

    }

    public *int* getIdUser() {

        return idUser;

    }

    public *void* setIdUser(*int* *idUser*) {

*this*.idUser = idUser;

    }

    public *String* getUserName() {

        return *this*.userName;

    }

    public *void* setUserName(*String* *userName*) {

*this*.userName = userName;

    }

    public *String* getPassword() {

        return *this*.password;

    }

    public *void* setPassword(*String* *password*) {

*this*.password = password;

    }

}

### ***2.2.2 Tạo lớp UserSolution.java***

package solution;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import connection.JDBCConnection;

import entity.User;

public class UserSolution {

    private *Connection* conn;

    public *boolean* checkUser1(*String* *username*, *String* *pass*) throws *SQLException* {

          conn = JDBCConnection.getJDBCConnection();

          try {

*String* sql = "SELECT userName,password FROM user WHERE userName= '"+ username +"' AND password='" + pass + "'";

*PreparedStatement* statement =  conn.prepareStatement(sql);

*ResultSet* resultSet = statement.executeQuery();

              if (resultSet.next()) {

*User* user = **new** User( resultSet.getString("username"), resultSet.getString("password"));

                  if (user.getUserName().equals(username) && user.getPassword().equals(pass)) {

                      return true;

                   }

              }

          } catch (*SQLException* *e*) {

              e.printStackTrace();

          }

//        if (user != null) {

//            if ("".equals(user.getUserName())

//                    && "".equals(user.getPassword())) {

//                return true;

//            }

//        }

        return false;

    }

    public *boolean* checkUser() {

        try {

            if (checkUser1("ducnhuad", "ducnhu090323") ||checkUser1("trungdz", "trungmegai") ||checkUser1("tungdz", "tungtien123")) {

                return true;

            }

        } catch (*SQLException* *e*) {

            e.printStackTrace();

            return false;

        }return false;

    }

}

### ***2.2.3 Tạo lớp LoginView.java***

package view;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JPasswordField;

import javax.swing.JTextField;

import javax.swing.SpringLayout;

import javax.swing.WindowConstants;

import entity.User;

public class LoginView extends *JFrame* implements *ActionListener* {

    private static final *long* serialVersionUID = 1L;

    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

        layout.putConstraint(SpringLayout.WEST, userNameLabel, 60, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH, userNameLabel, 80, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, passwordlabel, 60, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH, passwordlabel, 105, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, userNameField, 80, SpringLayout.WEST, userNameLabel);

        layout.putConstraint(SpringLayout.NORTH, userNameField, 80, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, passwordField, 80, SpringLayout.WEST, passwordlabel);

        layout.putConstraint(SpringLayout.NORTH, passwordField, 105, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, loginBtn, 80, SpringLayout.WEST, passwordlabel);

        layout.putConstraint(SpringLayout.NORTH, loginBtn, 130, SpringLayout.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*.setLocationRelativeTo(null);

    }

    public *void* showMessage(*String* *message*) {

        JOptionPane.showMessageDialog(*this*, message);

    }

    public *User* getUser() {

        return **new** User( userNameField.getText(),

                String.copyValueOf(passwordField.getPassword()));

    }

    public *void* actionPerformed(*ActionEvent* *e*) {

    }

    public *void* addLoginListener(*ActionListener* *listener*) {

        loginBtn.addActionListener(listener);

    }

}

### ***2.2.4 Tạo lớp LoginController.java***

package controller;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import solution.UserSolution;

import entity.User;

import view.LoginView;

import view.StudentView;

public class LoginController {

    private *UserSolution* userSolution;

    private *LoginView* loginView;

    private *StudentView* studentView;

    public LoginController(*LoginView* *view*) {

*this*.loginView = view;

*this*.userSolution = **new** UserSolution();

        view.addLoginListener(**new** LoginListener());

    }

    public *void* showLoginView() {

        loginView.setVisible(true);

    }

    /\*\*

     \* Lớp LoginListener

     \* chứa cài đặt cho sự kiện click button "Login"

     \*

     \*/

    class LoginListener implements *ActionListener* {

        public *void* actionPerformed(*ActionEvent* *e*) {

            if (userSolution.checkUser()) {

                // nếu đăng nhập thành công, mở màn hình quản lý sinh viên

                studentView = **new** StudentView();

*StudentController* studentController = **new** StudentController(studentView);

                studentController.showStudentView();

                loginView.setVisible(false);

            } else {

                loginView.showMessage("username hoặc password không đúng.");

            }

        }

    }

}

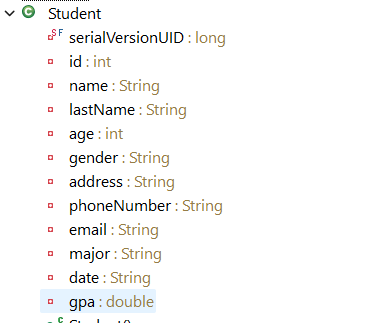
## **2.3 Tạo chức năng quản lý sinh viên**

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"

### ***2.3.1 Tạo lớp Student.java***

Lớp này gồm các thuộc tính sau :



package entity;

import java.io.Serializable;

import javax.xml.bind.annotation.XmlAccessType;

import javax.xml.bind.annotation.XmlAccessorType;

import javax.xml.bind.annotation.XmlRootElement;

@*XmlRootElement*(name = "student")

@*XmlAccessorType*(XmlAccessType.FIELD)

public class Student implements *Serializable* {

    private static final *long* serialVersionUID = 1L;

    private  *int* id;

    private *String* name;

    private *String* lastName;

    private *int* age;

    private *String* gender;

    private *String* address;

    private *String* phoneNumber;

    private *String* email;

    private *String* major;

    private *String* date;

    private *double* gpa;

    public Student() {

    }

    /\*\*

     \* @param *id*

     \* @param *name*

     \* @param *lastName*

     \* @param *age*

     \* @param *gender*

     \* @param *address*

     \* @param *phoneNumber*

     \* @param *email*

     \* @param *major*

     \* @param *date*

     \* @param *gpa*

     \*/

    public Student(*int* *id*, *String* *name*, *String* *lastName*, *int* *age*, *String* *gender*, *String* *address*, *String* *phoneNumber*,

*String* *email*, *String* *major*, *String* *date*, *double* *gpa*) {

*super*();

*this*.id = id;

*this*.name = name;

*this*.lastName = lastName;

*this*.age = age;

*this*.gender = gender;

*this*.address = address;

*this*.phoneNumber = phoneNumber;

*this*.email = email;

*this*.major = major;

*this*.date = date;

*this*.gpa = gpa;

    }

    public *String* getPhoneNumber() {

        return phoneNumber;

    }

    public *String* getEmail() {

        return email;

    }

    public *void* setEmail(*String* *email*) {

*this*.email = email;

    }

    public *void* setPhoneNumber(*String* *phoneNumber*) {

*this*.phoneNumber = phoneNumber;

    }

    public *int* getId() {

        return *this*.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 *double* getGpa() {

        return gpa;

    }

    public *void* setGpa(*double* *gpa*) {

*this*.gpa = gpa;

    }

    public *String* getLastName() {

        return lastName;

    }

    public *void* setLastName(*String* *lastName*) {

*this*.lastName = lastName;

    }

    public *String* getGender() {

        return gender;

    }

    public *void* setGender(*String* *gender*) {

*this*.gender = gender;

    }

    public *String* getMajor() {

        return major;

    }

    public *void* setMajor(*String* *major*) {

*this*.major = major;

    }

    public *String* getDate() {

        return date;

    }

    public *void* setDate(*String* *date*) {

*this*.date = date;

    }

}

### ***2.3.2 Tạo lớp StudentXML.java***

package entity;

import java.util.List;

import javax.xml.bind.annotation.XmlAccessType;

import javax.xml.bind.annotation.XmlAccessorType;

import javax.xml.bind.annotation.XmlRootElement;

@*XmlRootElement*(name = "students")

@*XmlAccessorType*(XmlAccessType.FIELD)

public class StudentXML {

    private *List*<Student> student;

    public *List*<Student> getStudent() {

        return student;

    }

    public *void* setStudent(*List*<Student> *student*) {

*this*.student = student;

    }

}

### ***2.3.3 Tạo lớp StudentSolution.java***

package solution;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

import entity.Student;

import entity.StudentXML;

import utils.FileUtils;

import view.StudentView;

/\*\*

 \* StudentSolution class

 \*

 \* @author duc

 \*/

public class StudentSolution {

    private static final *String* STUDENT\_FILE\_NAME = "student.xml";

    private *List*<Student> listStudents;

    public StudentSolution() {

*this*.listStudents = readListStudents();

        if (listStudents == null) {

            listStudents = **new** *ArrayList*<Student>();

        }

    }

    /\*\*

     \* Lưu các đối tượng student vào file student.xml

     \*

     \* @param *students*

     \*/

    public *void* writeListStudents(*List*<Student> *students*) {

*StudentXML* studentXML = **new** StudentXML();

        studentXML.setStudent(students);

        FileUtils.writeXMLtoFile(STUDENT\_FILE\_NAME, studentXML);

    }

    /\*\*

     \* Đọc các đối tượng student từ file student.xml

     \*

     \* @return list student

     \*/

    public *List*<Student> readListStudents() {

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

*StudentXML* studentXML = (StudentXML) FileUtils.readXMLFile(

                STUDENT\_FILE\_NAME, StudentXML.class);

        if (studentXML != null) {

            list = studentXML.getStudent();

        }

        return list;

    }

    public *void* add(*Student* *student*) {

*int* id = 1;

        if (listStudents != null && listStudents.size() > 0) {

            id = listStudents.size() + 1;

        }

        student.setId(id);

        listStudents.add(student);

        writeListStudents(listStudents);

    }

    /\*\*

     \* cập nhật student vào listStudents và lưu listStudents vào file

     \*

     \* @param *student*

     \*/

    public *void* edit(*Student* *student*) {

*int* size = listStudents.size();

        for (*int* i = 0; i < size; i++) {

            if (listStudents.get(i).getId() == student.getId()) {

                listStudents.get(i).setName(student.getName());

                listStudents.get(i).setLastName(student.getLastName());

                listStudents.get(i).setAge(student.getAge());

                listStudents.get(i).setGender(student.getGender());

                listStudents.get(i).setAddress(student.getAddress());

                listStudents.get(i).setPhoneNumber(student.getPhoneNumber());

                listStudents.get(i).setEmail(student.getEmail());

                listStudents.get(i).setMajor(student.getMajor());

                listStudents.get(i).setDate(student.getDate());

                listStudents.get(i).setGpa(student.getGpa());

                writeListStudents(listStudents);

                break;

            }

        }

    }

    /\*\*

     \* xóa student từ listStudents và lưu listStudents vào file

     \*

     \* @param *student*

     \*/

    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) {

            listStudents.remove(student);

            writeListStudents(listStudents);

            return true;

        }

        return false;

    }

    public *void* sortStudentByName() {

        Collections.sort(listStudents, **new** *Comparator*<Student>() {

            public *int* compare(*Student* *student1*, *Student* *student2*) {

                return student1.getLastName().compareTo(student2.getLastName());

            }

        });

    }

    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;

            }

        });

    }

    public *List*<Student> getListStudents() {

        return listStudents;

    }

    public *void* setListStudents(*List*<Student> *listStudents*) {

*this*.listStudents = listStudents;

    }

}

### ***2.3.4 Tạo lớp StudentView.java***

package view;

import java.awt.Dimension;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.util.List;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextArea;

import javax.swing.JTextField;

import javax.swing.SpringLayout;

import javax.swing.WindowConstants;

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 *JLabel* genderLabel;

    private *JTextField* gendertTextField;

    private *JLabel* lastNameLabel;

    private *JTextField* lastNameTextField;

    private *JLabel* phoneNumberLabel;

    private *JTextField* phoneNumberTextField;

    private *JLabel* emailLabel;

    private *JTextField* emailTextField;

    private *JLabel* majorLabel;

    private *JTextField* majorTextField;

    private *JLabel* dateLabel;

    private *JTextField* dateTextField;

    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","Last Name", "Age","Gender", "Address","PhoneNumber","Email","Major","CourseStart", "GPA"};

    // định nghĩa dữ liệu mặc định của bẳng 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");

        lastNameLabel = **new** JLabel("Last name");

        ageLabel = **new** JLabel("Age");

        phoneNumberLabel = **new** JLabel("PhoneNumber");

        majorLabel = **new** JLabel("Major");

        emailLabel = **new** JLabel("Email");

        dateLabel = **new** JLabel("CourseStart");

        addressLabel = **new** JLabel("Address");

        gpaLabel = **new** JLabel("GPA");

        // khởi tạo các trường nhập dữ liệu cho student

        idField = **new** JTextField(6);

        idField.setEditable(false);

        nameField = **new** JTextField(15);

        lastNameTextField = **new** JTextField(15);

        ageField = **new** JTextField(6);

        addressTA = **new** JTextArea();

        addressTA.setColumns(15);

        addressTA.setRows(5);

        phoneNumberTextField = **new** JTextField(15);

        emailTextField = **new** JTextField(15);

        majorTextField = **new** JTextField(15);

        dateTextField = **new** JTextField(15);

        jScrollPaneAddress = **new** JScrollPane();

        jScrollPaneAddress.setViewportView(addressTA);

        gpaField = **new** JTextField(6);

        genderLabel = **new** JLabel("Gender");

        gendertTextField= **new** JTextField(10);

        // cài đặt các cột và data cho bảng student

        studentTable.setModel(**new** DefaultTableModel((*Object*[][]) data, columnNames));

        jScrollPaneStudentTable.setViewportView(studentTable);

        jScrollPaneStudentTable.setPreferredSize(**new** Dimension (850,600));

         // 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(1200, 720);

        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(phoneNumberLabel);

        panel.add(lastNameLabel);

        panel.add(majorLabel);

        panel.add(emailLabel);

        panel.add(dateLabel);

        panel.add(idField);

        panel.add(nameField);

        panel.add(ageField);

        panel.add(jScrollPaneAddress);

        panel.add(gpaField);

        panel.add(phoneNumberTextField);

        panel.add(lastNameTextField);

        panel.add(majorTextField);

        panel.add(emailTextField);

        panel.add(dateTextField);

        panel.add(genderLabel);

        panel.add(gendertTextField);

        // cài đặt vị trí các thành phần trên màn hình login

        layout.putConstraint(SpringLayout.WEST, idLabel, 10, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH, idLabel, 10, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, idField, 110, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH, idField, 10, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, nameLabel, 10, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH, nameLabel, 40, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, nameField, 110, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH, nameField, 40, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, lastNameLabel, 10, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH, lastNameLabel, 70, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, lastNameTextField, 110, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH, lastNameTextField, 70, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, ageLabel, 10, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH, ageLabel, 100, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, ageField, 110, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH, ageField, 100, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, genderLabel, 10, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH, genderLabel, 130, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, gendertTextField, 110, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH,gendertTextField, 130, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, addressLabel, 10, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH, addressLabel, 160, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, jScrollPaneAddress, 110, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH, jScrollPaneAddress, 160, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, phoneNumberLabel, 10, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH, phoneNumberLabel, 260, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, phoneNumberTextField, 110, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH, phoneNumberTextField, 260, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, emailLabel, 10, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH, emailLabel, 290, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, emailTextField, 110, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH,emailTextField, 290, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, majorLabel, 10, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH, majorLabel, 320, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, majorTextField, 110, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH,majorTextField, 320, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, dateLabel, 10, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH, dateLabel, 350, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, dateTextField, 110, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH,dateTextField, 350, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, gpaLabel, 10, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH, gpaLabel, 380, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, gpaField, 110, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH,gpaField, 380, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, jScrollPaneStudentTable, 320, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH, jScrollPaneStudentTable, 10, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, addStudentBtn, 10, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH, addStudentBtn, 500, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, editStudentBtn, 60, SpringLayout.WEST, addStudentBtn);

        layout.putConstraint(SpringLayout.NORTH, editStudentBtn, 500, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, deleteStudentBtn, 60, SpringLayout.WEST, editStudentBtn);

        layout.putConstraint(SpringLayout.NORTH, clearBtn, 500, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, clearBtn, 80, SpringLayout.WEST, deleteStudentBtn);

        layout.putConstraint(SpringLayout.NORTH, deleteStudentBtn, 500, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, sortStudentGPABtn, 320, SpringLayout.WEST, panel);

        layout.putConstraint(SpringLayout.NORTH, sortStudentGPABtn, 630, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, sortStudentNameBtn, 145, SpringLayout.WEST, sortStudentGPABtn);

        layout.putConstraint(SpringLayout.NORTH, sortStudentNameBtn, 630, SpringLayout.NORTH, panel);

*this*.add(panel);

*this*.pack();

*this*.setTitle("Student Information");

*this*.setSize(1200, 720);

        // disable Edit and Delete buttons

        editStudentBtn.setEnabled(false);

        deleteStudentBtn.setEnabled(false);

        // enable Add button

        addStudentBtn.setEnabled(true);

*this*.setLocationRelativeTo(null);

    }

    public *void* showMessage(*String* *message*) {

        JOptionPane.showMessageDialog(*this*, message);

    }

    /\*\*

     \* hiển thị list student vào bảng studentTable

     \*

     \* @param *list*

     \*/

    public *void* showListStudents(*List*<Student> *list*) {

*int* size = list.size();

        // khởi tạo mảng 2 chiều students, trong đó:

        // số hàng: là kích thước của list student

*Object* [][] students = **new** *Object*[size][11];

        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).getLastName();

            students[i][3] = list.get(i).getAge();

            students[i][4] = list.get(i).getGender();

            students[i][5] = list.get(i).getAddress();

            students[i][6] = list.get(i).getPhoneNumber();

            students[i][7] = list.get(i).getEmail();

            students[i][8] = list.get(i).getMajor();

            students[i][9] = list.get(i).getDate();

            students[i][10] = list.get(i).getGpa();

        }

        studentTable.setModel(**new** DefaultTableModel(students, columnNames));

    }

    /\*\*

     \* điền thông tin của hàng được chọn từ bảng student

     \* vào các trường tương ứng của student.

     \*/

    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());

            lastNameTextField.setText(studentTable.getModel().getValueAt(row, 2).toString());

            ageField.setText(studentTable.getModel().getValueAt(row, 3).toString());

            gendertTextField.setText(studentTable.getModel().getValueAt(row, 4).toString());

            addressTA.setText(studentTable.getModel().getValueAt(row, 5).toString());

            phoneNumberTextField.setText(studentTable.getModel().getValueAt(row, 6).toString());

            emailTextField.setText(studentTable.getModel().getValueAt(row, 7).toString());

            majorTextField.setText(studentTable.getModel().getValueAt(row, 8).toString());

            dateTextField.setText(studentTable.getModel().getValueAt(row, 9).toString());

            gpaField.setText(studentTable.getModel().getValueAt(row, 10).toString());

            // enable Edit and Delete buttons

            editStudentBtn.setEnabled(true);

            deleteStudentBtn.setEnabled(true);

            // disable Add button

            addStudentBtn.setEnabled(false);

        }

    }

    /\*\*

     \* xóa thông tin student

     \*/

    public *void* clearStudentInfo() {

        idField.setText("");

        nameField.setText("");

        ageField.setText("");

        addressTA.setText("");

        gpaField.setText("");

        gendertTextField.setText("");

        phoneNumberTextField.setText("");

        majorTextField.setText("");

        emailTextField.setText("");

        dateTextField.setText("");

        lastNameTextField.setText("");

        // disable Edit and Delete buttons

        editStudentBtn.setEnabled(false);

        deleteStudentBtn.setEnabled(false);

        // enable Add button

        addStudentBtn.setEnabled(true);

    }

    /\*\*

     \* hiện thị thông tin student

     \*

     \* @param *student*

     \*/

    public *void* showStudent(*Student* *student*) {

        idField.setText("" + student.getId());

        nameField.setText(student.getName());

        lastNameTextField.setText(student.getLastName());

        ageField.setText("" + student.getAge());

        gendertTextField.setText( student.getGender());

        addressTA.setText(student.getAddress());

        phoneNumberTextField.setText(student.getPhoneNumber());

        emailTextField.setText(student.getEmail());

        majorTextField.setText(student.getMajor());

        dateTextField.setText(student.getDate());

        gpaField.setText("" + student.getGpa());

        // enable Edit and Delete buttons

        editStudentBtn.setEnabled(true);

        deleteStudentBtn.setEnabled(true);

        // disable Add button

        addStudentBtn.setEnabled(false);

    }

    /\*\*

     \* lấy thông tin student

     \*

     \* @return

     \*/

    public *Student* getStudentInfo() {

        if (!validateName() || !validateAge() || !validateAddress() ||  !validateGPA() || !validateDate() || !validatePhone() || !validateEmail()){

            return null;

        }

        try {

*Student* student = **new** Student();

            if (idField.getText() != null && !"".equals(idField.getText())) {

                student.setId(Integer.parseInt(idField.getText()));

            }

            student.setName(nameField.getText().trim());

            student.setLastName(lastNameTextField.getText().trim());

            student.setAge(Byte.parseByte(ageField.getText().trim()));

            student.setGender(gendertTextField.getText().trim());

            student.setAddress(addressTA.getText().trim());

            student.setPhoneNumber(phoneNumberTextField.getText().trim());

            student.setEmail(emailTextField.getText().trim());

            student.setMajor(majorTextField.getText().trim());

            student.setDate(dateTextField.getText().trim());

            student.setGpa(Double.parseDouble(gpaField.getText().trim()));

            return student;

        } catch (*Exception* *e*) {

            showMessage(e.getMessage());

        }

        return null;

    }

    private *boolean* validateEmail() {

        try {

*String* email = emailTextField.getText().trim();

*String* emailPattern = "^[a-zA-Z0-9\_+&\*-]+(?:\\." +

                    "[a-zA-Z0-9\_+&\*-]+)\*@" +

                    "(?:[a-zA-Z0-9-]+\\.)+[a-z" +

                    "A-Z]{2,7}$";

            if(!email.matches(emailPattern)  || email == null || "".equals(email)) {

                emailTextField.requestFocus();

                showMessage("gmail is not empty or not valid !");

                return false;

            }

        } catch (*Exception* *e*) {

            emailTextField.requestFocus();

            showMessage("Phone Number is not valid !");

            return false;

        }

        return true;

    }

    private *boolean* validateDate() {

*String* date = emailTextField.getText().trim();

            if( date == null || "".equals(date)) {

                emailTextField.requestFocus();

                showMessage("date is not empty !");

                return false;

            }

        return true;

    }

    private *boolean* validateName() {

*String* name = nameField.getText();

        if (name == null || "".equals(name.trim())) {

            nameField.requestFocus();

            showMessage("Name is not empty.");

            return false;

        }

        return true;

    }

    private *boolean* validatePhone() {

        try {

*String* phoneNumber = phoneNumberTextField.getText();

            if(phoneNumber == null || "".equals(phoneNumber.trim()) ) {

                phoneNumberTextField.requestFocus();

                showMessage("Phone Number is not empty !");

                return false;

            }

        }catch (*Exception* *e*) {

            phoneNumberTextField.requestFocus();

            showMessage("Phone Number is not valid!");

            return false;

        }

        return true;

    }

    private *boolean* validateAddress() {

*String* address = addressTA.getText();

        if (address == null || "".equals(address.trim())) {

            addressTA.requestFocus();

            showMessage("Address is not 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 is not valid");

                return false;

            }

        } catch (*Exception* *e*) {

            ageField.requestFocus();

            showMessage("Age is not valid!");

            return false;

        }

        return true;

    }

//    private boolean validateGender() {

//      String gender = gendertTextField.getText();

//      try {

//          if("".equals(gender.trim()) || !"Male".equals(gender.trim()) || !"Female".equals(gender.trim())) {

//              gendertTextField.requestFocus();

//              showMessage("Gender is not empty and gender must be 'Male' or 'Female' !");

//              return false;

//          }

//      }catch (Exception e) {

//          gendertTextField.requestFocus();

//          showMessage("Gender is not empty and gender must be 'Male' or 'Female' !");

//          return false;

//      }return true;

//    }

    private *boolean* validateGPA() {

        try {

*Float* gpa = Float.parseFloat(gpaField.getText().trim());

            if (gpa < 0 || gpa > 4) {

                gpaField.requestFocus();

                showMessage("GPA is not valid!");

                return false;

            }

        } catch (*Exception* *e*) {

            gpaField.requestFocus();

            showMessage("GPA is not valid!");

            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);

    }

    public *JTable* getStudentTable() {

        return studentTable;

    }

    public *void* setStudentTable(*JTable* *studentTable*) {

*this*.studentTable = studentTable;

    }

    public *String*[] getColumnNames() {

        return columnNames;

    }

    public *void* setColumnNames(*String*[] *columnNames*) {

*this*.columnNames = columnNames;

    }

}

***2.3.5 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 entity.Student;

import solution.StudentSolution;

import view.StudentView;

public class StudentController {

    private *StudentSolution* studentSolution;

    private *StudentView* studentView;

    public StudentController(*StudentView* *view*) {

*this*.studentView = view;

        studentSolution = **new** StudentSolution();

        view.addAddStudentListener(**new** AddStudentListener());

        view.addEdiStudentListener(**new** EditStudentListener());

        view.addDeleteStudentListener(**new** DeleteStudentListener());

        view.addClearListener(**new** ClearStudentListener());

        view.addSortStudentGPAListener(**new** SortStudentGPAListener());

        view.addSortStudentNameListener(**new** SortStudentNameListener());

        view.addListStudentSelectionListener(**new** ListStudentSelectionListener());

        view.addSearchBTnListener(**new** SearchSTudent());

    }

    public *void* showStudentView() {

*List*<Student> studentList = studentSolution.getListStudents();

        studentView.setVisible(true);

        studentView.showListStudents(studentList);

    }

    /\*\*

     \* Lớp AddStudentListener

     \* chứa cài đặt cho sự kiện click button "Add"

     \*/

    class AddStudentListener implements *ActionListener* {

        public *void* actionPerformed(*ActionEvent* *e*) {

*Student* student = studentView.getStudentInfo();

            if (student != null) {

                studentSolution.add(student);

                studentView.showStudent(student);

                studentView.showListStudents(studentSolution.getListStudents());

                studentView.showMessage("Thêm thành công!");

            }

        }

    }

    /\*\*

     \* Lớp EditStudentListener

     \* chứa cài đặt cho sự kiện click button "Edit"

     \*/

    class EditStudentListener implements *ActionListener* {

        public *void* actionPerformed(*ActionEvent* *e*) {

*Student* student = studentView.getStudentInfo();

            if (student != null) {

                studentSolution.edit(student);

                studentView.showStudent(student);

                studentView.showListStudents(studentSolution.getListStudents());

                studentView.showMessage("Cập nhật thành công!");

            } else {

                studentView.showMessage("Cập nhật thất bại!");

            }

        }

    }

    /\*\*

     \* Lớp DeleteStudentListener

     \* chứa cài đặt cho sự kiện click button "Delete"

     \*/

    class DeleteStudentListener implements *ActionListener* {

        public *void* actionPerformed(*ActionEvent* *e*) {

*Student* student = studentView.getStudentInfo();

            if (student != null) {

                studentSolution.delete(student);

                studentView.clearStudentInfo();

                studentView.showListStudents(studentSolution.getListStudents());

                studentView.showMessage("Xóa thành công!");

            }

        }

    }

    /\*\*

     \* Lớp ClearStudentListener

     \* chứa cài đặt cho sự kiện click button "Clear"

     \*/

    class ClearStudentListener implements *ActionListener* {

        public *void* actionPerformed(*ActionEvent* *e*) {

            studentView.clearStudentInfo();

        }

    }

    /\*\*

     \* Lớp SortStudentGPAListener

     \* chứa cài đặt cho sự kiện click button "Sort By GPA"

     \*/

    class SortStudentGPAListener implements *ActionListener* {

        public *void* actionPerformed(*ActionEvent* *e*) {

            studentSolution.sortStudentByGPA();

            studentView.showListStudents(studentSolution.getListStudents());

        }

    }

    /\*\*

     \* Lớp SortStudentGPAListener

     \* chứa cài đặt cho sự kiện click button "Sort By Name"

     \*/

    class SortStudentNameListener implements *ActionListener* {

        public *void* actionPerformed(*ActionEvent* *e*) {

            studentSolution.sortStudentByName();

            studentView.showListStudents(studentSolution.getListStudents());

        }

    }

    /\*\*

     \* Lớp ListStudentSelectionListener

     \* chứa cài đặt cho sự kiện chọn student trong bảng student     \*/

    class ListStudentSelectionListener implements *ListSelectionListener* {

        public *void* valueChanged(*ListSelectionEvent* *e*) {

            studentView.fillStudentFromSelectedRow();

        }

    }

}

### ***2.5.6 Tạo lớp FileUtils.java***

package utils;

import java.io.File;

import javax.xml.bind.JAXBContext;

import javax.xml.bind.JAXBException;

import javax.xml.bind.Marshaller;

import javax.xml.bind.Unmarshaller;

public class FileUtils {

    /\*\*

     \* Chuyển đổi đối tượng object về định dạng XML

     \* Sau đo lưu vào fileName

     \*

     \* @param *fileName*

     \* @param *object*

     \*/

    public static *void* writeXMLtoFile(*String* *fileName*, *Object* *object*) {

        try {

            // tạo đối tượng JAXB Context

*JAXBContext* jaxbContext = JAXBContext.newInstance(object.getClass());

            // Create đối tượng Marshaller

*Marshaller* jaxbMarshaller = jaxbContext.createMarshaller();

            // formating

            jaxbMarshaller.setProperty(Marshaller.JAXB\_FORMATTED\_OUTPUT, Boolean.TRUE);

            // lưu nội dung XML vào file

*File* xmlFile = **new** File(fileName);

            jaxbMarshaller.marshal(object, xmlFile);

        } catch (*JAXBException* *e*) {

            e.printStackTrace();

        }

    }

    /\*\*

     \* Đọc nội dung fileName, sau đó chuyển đổi nội dung của file

     \* thành đối tượng có kiểu là clazz.

     \*

     \* @param *fileName*

     \* @param *clazz*

     \* @return

     \*/

    public static *Object* readXMLFile(*String* *fileName*, *Class*<*?*> *clazz*) {

        try {

*File* xmlFile = **new** File(fileName);

*JAXBContext* jaxbContext = JAXBContext.newInstance(clazz);

*Unmarshaller* jaxbUnmarshaller = jaxbContext.createUnmarshaller();

            return jaxbUnmarshaller.unmarshal(xmlFile);

        } catch (*JAXBException* *e*) {

            e.printStackTrace();

        }

        return null;

    }

}

### ***2.3.7 Tạo lớp App.java***

package main;

import java.awt.EventQueue;

import controller.LoginController;

import view.LoginView;

public class App {

    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();

            }

        });

    }

}

## **2.4 Kết nối cơ sở dữ liệu (nếu có)**

File này có thể có hoặc không .

### ***2.4.1 Tạo file JDBCConnection.java***

package connection;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class JDBCConnection {

    public static *Connection* getJDBCConnection() {

        final *String* url = "jdbc:mysql://localhost:3306/studentmanage";

        final *String* user = "root";

        final *String* password = "ducnhu1234";

         try {

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

            return DriverManager.getConnection(url, user, password);

            } catch (*SQLException* *e*) {

                e.printStackTrace();

            } catch (*ClassNotFoundException* *e*) {

                e.printStackTrace();

        }

        return null;

     }

}

### ***2.4.2 Tạo file StudentDao.java***

package connection;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

import entity.Student;

public class StudentDao {

    public *List*<Student> getAllStudent(){

*List*<Student> students = **new** *ArrayList*<>();

*Connection* con = JDBCConnection.getJDBCConnection();

*String* sql = "Select \*from student";

        try {

*PreparedStatement* preparedStatement = con.prepareStatement(sql);

*ResultSet* rs = preparedStatement.executeQuery();

            while (rs.next()) {

*Student* student = **new** Student();

                student.setId(rs.getInt("id"));

                student.setName(rs.getString("name"));

                student.setLastName(rs.getString("lastName"));

                student.setAge(rs.getInt("age"));

                student.setGender(rs.getString("gender"));

                student.setAddress(rs.getString("address"));

                student.setPhoneNumber(rs.getString("phoneNumber"));

                student.setEmail(rs.getString("email"));

                student.setMajor(rs.getString("major"));

                student.setDate(rs.getString("date"));

                student.setGpa(rs.getDouble("gpa"));

                students.add(student);

            }

        } catch (*SQLException* *e*) {

            e.printStackTrace();

        }

        return students;

    }

}

### ***2.4.3 Tạo lớp UserDao.java***

package connection;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

import entity.User;

public class UserDao {

    public *List*<User> getAllUser() {

*List*<User> a = **new** *ArrayList*<>();

*Connection* connection = JDBCConnection.getJDBCConnection();

*String* sql = "select \* from user";

     try {

*PreparedStatement* preparedStatement = connection.prepareStatement(sql);

*ResultSet* rs =  preparedStatement.executeQuery();

         while ( rs.next()) {

*User* u = **new** User();

             //u.setIdUser(rs.getInt("idUser"));

             u.setUserName(rs.getString("userName"));

             u.setPassword(rs.getString("password"));

             a.add(u);

         }

     } catch (*SQLException* *e*) {

            // TODO Auto-generated catch block

            e.printStackTrace();

        }

     return a;

    }

}

### ***2.4.4 Tạo lớp InsertStatement.java***

package connection;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.SQLException;

import entity.Student;

import entity.User;

public class InsertStatement {

    public static *void* insertStudent(*Student* *student*) {

*Connection* connection = JDBCConnection.getJDBCConnection();

*String* sql = "INSERT INTO student (id, name, lastName, age, gender, address, phoneNumber, email, major, date, gpa) VALUES(?,?,?,?,?,?,?,?,?,?,?)";

     try {

*PreparedStatement* pst = connection.prepareStatement(sql);

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

            pst.setString(2, student.getName());

            pst.setString(3, student.getLastName());

            pst.setInt(4, student.getAge());

            pst.setString(5, student.getGender());

            pst.setString(6, student.getAddress());

            pst.setString(7, student.getPhoneNumber());

            pst.setString(8, student.getEmail());

            pst.setString(9, student.getMajor());

            pst.setString(10, student.getDate());

            pst.setDouble(11, student.getGpa());

            pst.executeUpdate();

     } catch (*SQLException* *e*) {

           System.out.println("Insert fail !");

            e.printStackTrace();

        }

    }

    public static *void* insertUser(*User* *user*) {

*Connection* connection = JDBCConnection.getJDBCConnection();

*String* sql = "insert into user ( userName, password) values (?,?)";

     try {

*PreparedStatement* pst = connection.prepareStatement(sql);

            //pst.setInt(1, user.getIdUser());

            pst.setString(1,user.getUserName() );

            pst.setString(2,user.getPassword() );

            pst.executeUpdate();

     } catch (*SQLException* *e*) {

           System.out.println("Insert fail !");

            e.printStackTrace();

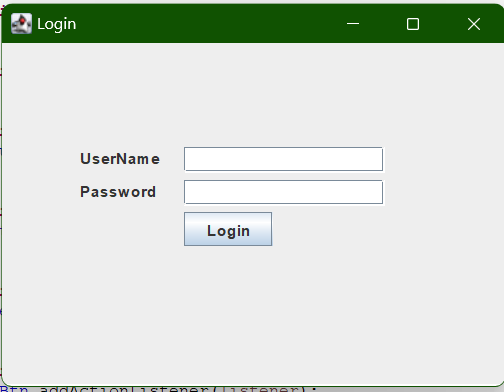
        }

    }

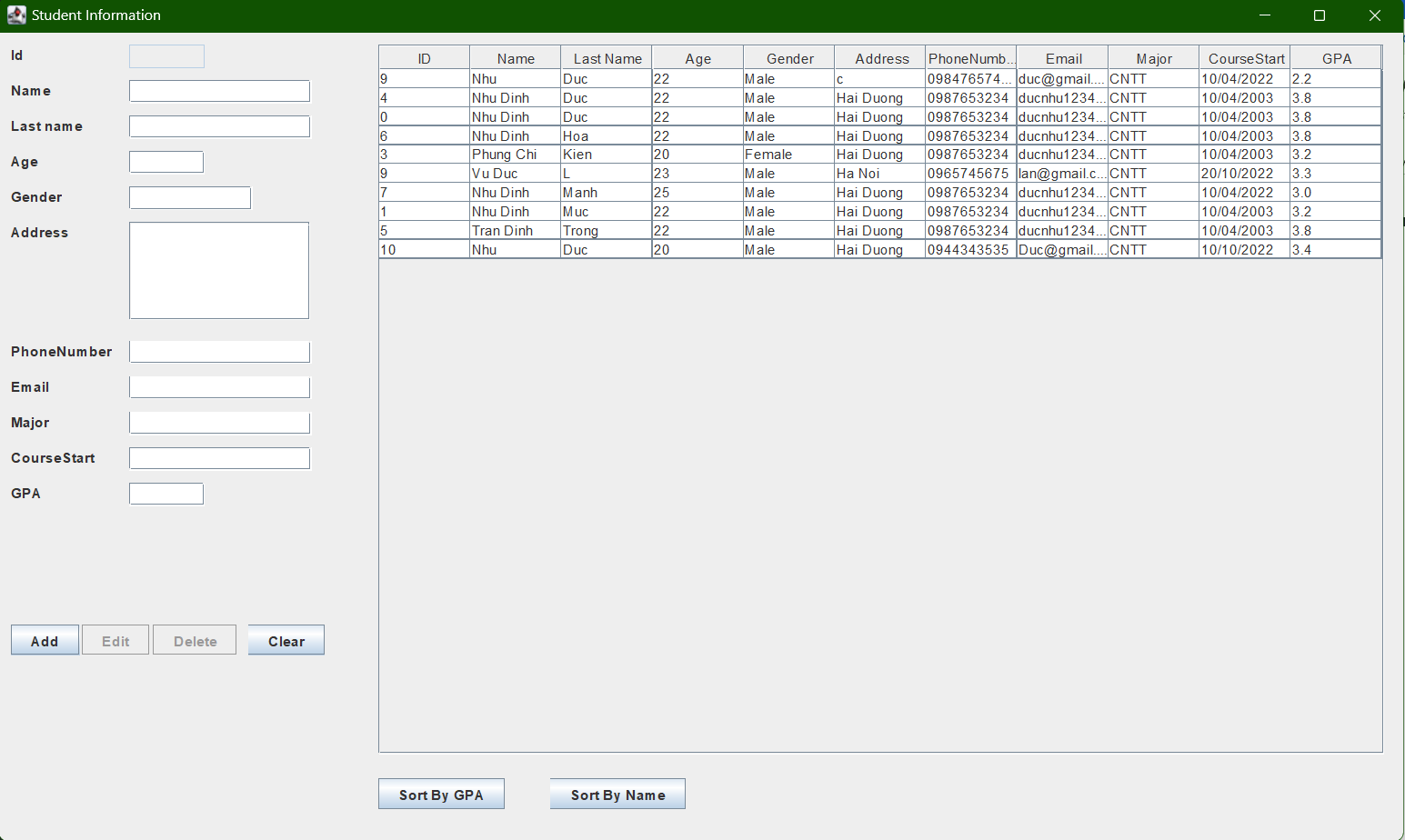
}

# 3. Giao diện

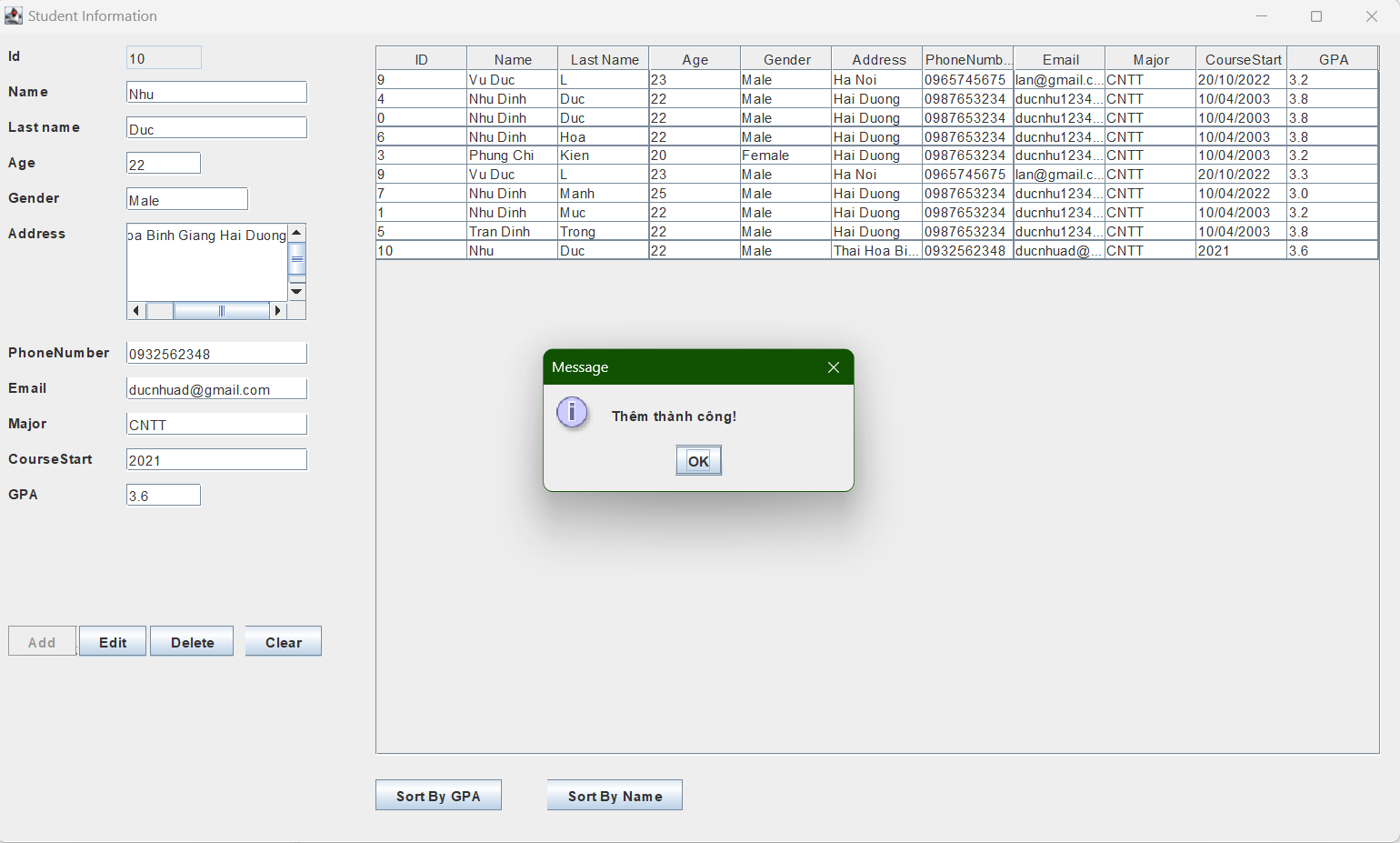
Màn hình đăng nhập :



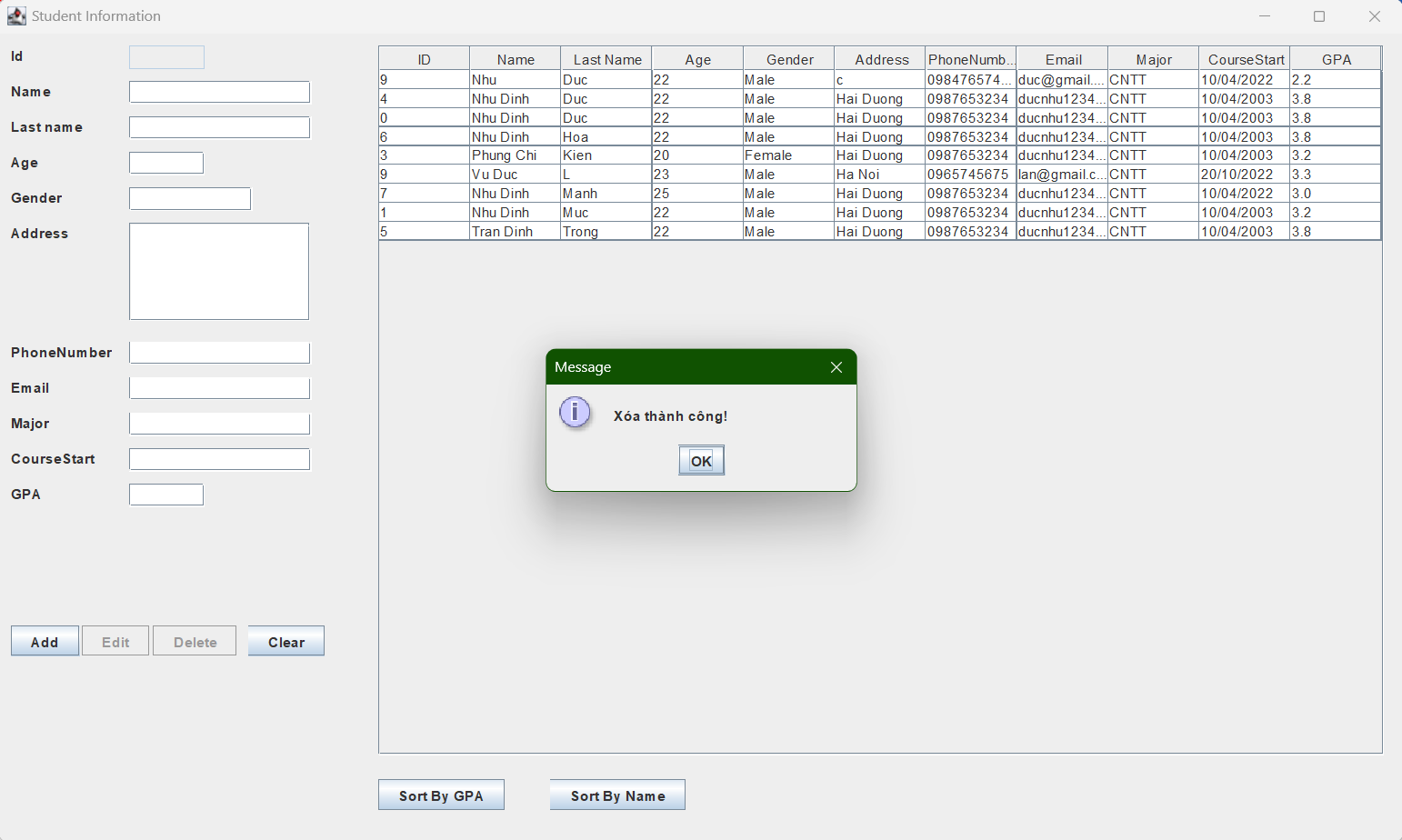
Màn hình Quản lý các chức năng:



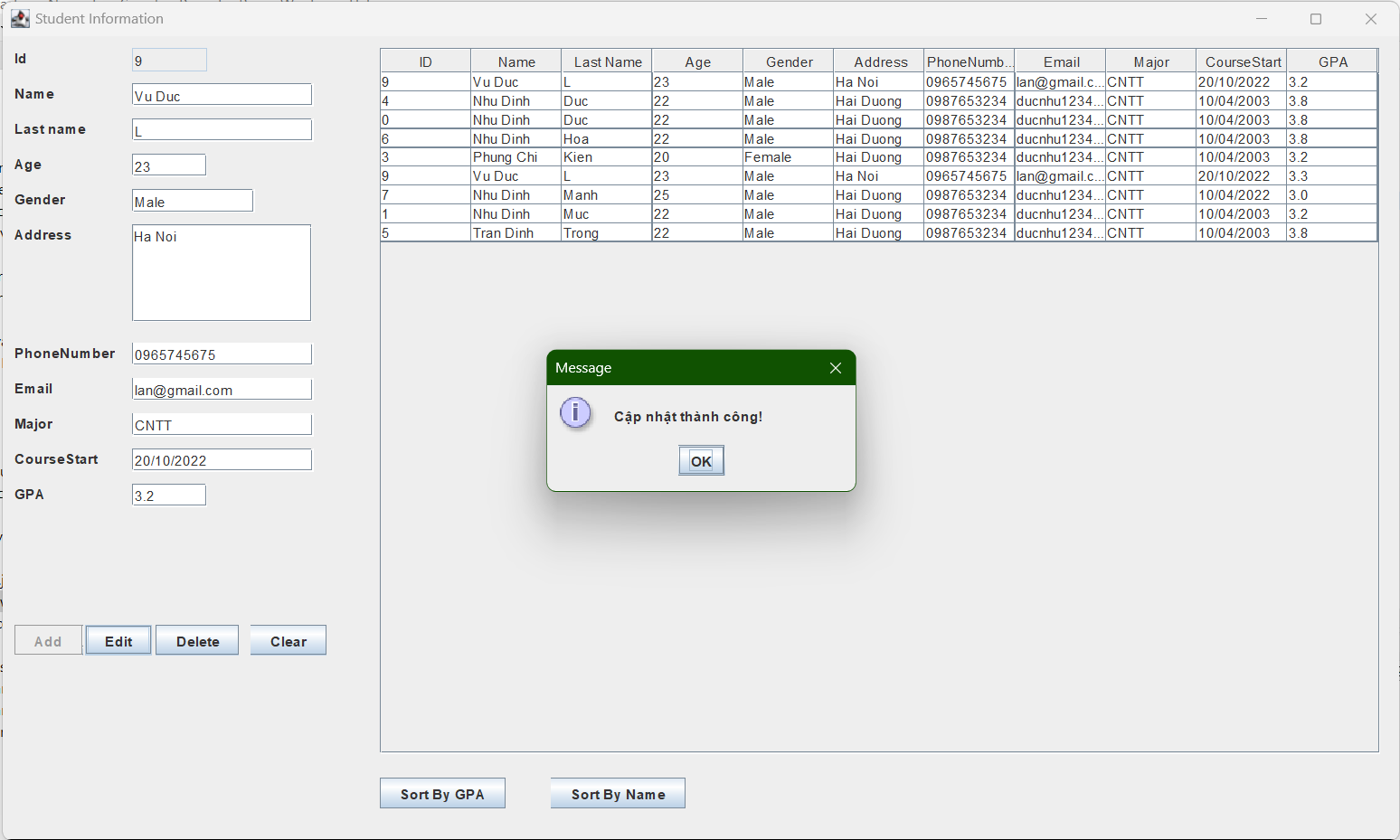
Màn hình chức năng Thêm sinh viên:



Màn hình chức năng Xóa sinh viên:



Màn hình chức năng Cập nhật sinh viên:



# 

# 

# TÀI LIỆU THAM KHẢO

**- https://viettuts.vn/**

**- https://freetuts.net/ - https://www.w3schools.com/-**

**- Delf Blog/**