Sistem de gestiune a notelor pentru universitate

Proiect Laborator Medii și Instrumente de Programare

Realizat de: Pirău Tudor-Ioan

Grupa 10LF332

Universitatea Transilvania din Brașov
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

Cuprins:

Structura codului:

Interfețe:

IPerson:

```
package Interfaces;

public interface IPerson {
    int getId();

    void setId(int id);

    String getFirstName();

    void setFirstName(String firstName);

    String getLastName();

    void setLastName(String lastName);

    void displayInfo();
}
```

IStudent:

```
package Interfaces;

import Models.ClassInfo;
import java.util.List;

public interface IStudent {
    int getId();
    void setId(int id);
    String getFirst_name();
    void setFirst_name(String first_name);
    String getLast_name();
    void setLast_name(String last_name);
    int getAge();
    void setAge(int age);
    String getGender();
    void setGender(String gender);
    String getEmail();
    void setEmail(String email);
    String getPhone();
    void setPhone(String phone);
    String getAddress();
    void setAddress(String address);
    String getFaculty();
    void setFaculty(String faculty);
    String getMajor();
    void setMajor(String major);
    boolean isPsycho_pedagogical_module();
    void setPsycho_pedagogical_module(boolean psycho_pedagogical_module);
    List<ClassInfo> getClasses();
    void addClass(ClassInfo classInfo);
    ClassInfo getClassInfo(String className);
}
```

IProfessor:

```
package Interfaces;

public interface IProfessor {
    int getId();

    void setId(int id);

    String getFirstName();

    void setFirstName(String firstName);

    String getLastName();

    void setLastName(String lastName);

    String getFaculty();

    void setFaculty(String faculty);

    String getClassName();

    void setClassName(String className);
}
```

IClassInfo:

```
package Interfaces;

import java.util.List;

public interface IClassInfo {
    String getClassName();

    void setClassName(String className);

    List<Float> getGrades();

    void addGrade(float grade);

    float calculateAverageGrade();
}
```

IStudentState:

```
package Interfaces;

import Models.ClassInfo;

import java.util.List;

public interface IStudentState {

    List<ClassInfo> getClasses();
```

```
    void addClass(ClassInfo classInfo);

    ClassInfo getClassInfo(String className);
}
```

IProfessorDAO:

```
package Interfaces;

import Models.Professor;

import java.sql.SQLException;
import java.util.List;

public interface IProfessorDAO {
    List<Professor> getAllProfessors() throws SQLException;
}
```

IStudentDAO:

```
package Interfaces;

import Models.ClassInfo;
import Models.Student;

import java.sql.SQLException;
import java.util.List;

public interface IStudentDAO {
    void addStudent(Student student) throws SQLException;

    Student getStudent(int id) throws SQLException;

    List<Student> getAllStudents() throws SQLException;

    void updateStudent(Student student) throws SQLException;

    void deleteStudent(int id) throws SQLException;

    List<Double> getStudentGrades(int studentId, String className) throws
SQLException;

    void addGrade(int studentId, String className, float grade) throws
SQLException;

    List<ClassInfo> getAllGrades(int studentId) throws SQLException;

    void deleteGrade(int studentId, String className, float grade) throws
SQLException;

    List<String> getAllClasses() throws SQLException; // Add this method
}
```

Clase:

Person.java:

```java
package Models;

import Interfaces.IPerson;

public abstract class Person implements IPerson {
    private int id;
    private String firstName;
    private String lastName;

    public Person(int id, String firstName, String lastName) {
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
    }

    @Override
    public int getId() {
        return id;
    }

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

    @Override
    public String getFirstName() {
        return firstName;
    }

    @Override
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    @Override
    public String getLastName() {
        return lastName;
    }

    @Override
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    @Override
    public abstract void displayInfo();
}
```

Student.java:

```java
package Models;

import Interfaces.IStudent;

import java.util.ArrayList;
import java.util.List;

public class Student extends Person implements IStudent {
    private int age;
    private String gender;
    private String email;
    private String phone;
    private String address;
    private String faculty;
    private String major;
    private boolean psycho_pedagogical_module;
    private List<ClassInfo> classes;

    public Student() {
        super(0, "", "");
        this.age = 0;
        this.gender = "";
        this.email = "";
        this.phone = "";
        this.address = "";
        this.faculty = "";
        this.major = "";
        this.psycho_pedagogical_module = false;
        this.classes = new ArrayList<>();
    }

    public Student(int id, String firstName, String lastName, int age,
String gender, String email,
                   String phone, String address, String faculty, String
major, boolean psycho_pedagogical_module) {
        super(id, firstName, lastName);
        this.age = age;
        this.gender = gender;
        this.email = email;
        this.phone = phone;
        this.address = address;
        this.faculty = faculty;
        this.major = major;
        this.psycho_pedagogical_module = psycho_pedagogical_module;
        this.classes = new ArrayList<>();
    }

    @Override
    public void displayInfo() {
        System.out.println("Student ID: " + getId());
        System.out.println("Name: " + getFirstName() + " " +
getLastName());
        System.out.println("Age: " + age);
        System.out.println("Gender: " + gender);
        System.out.println("Email: " + email);
        System.out.println("Phone: " + phone);
        System.out.println("Address: " + address);
        System.out.println("Faculty: " + faculty);
        System.out.println("Major: " + major);
        System.out.println("Psycho-Pedagogical Module: " +
psycho_pedagogical_module);
```

```
    }

    @Override
    public int getId() {
        return super.getId();
    }

    @Override
    public void setId(int id) {
        super.setId(id);
    }

    @Override
    public String getFirst_name() {
        return super.getFirstName();
    }

    @Override
    public void setFirst_name(String firstName) {
        super.setFirstName(firstName);
    }

    @Override
    public String getLast_name() {
        return super.getLastName();
    }

    @Override
    public void setLast_name(String lastName) {
        super.setLastName(lastName);
    }

    @Override
    public int getAge() {
        return age;
    }

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

    @Override
    public String getGender() {
        return gender;
    }

    @Override
    public void setGender(String gender) {
        this.gender = gender;
    }

    @Override
    public String getEmail() {
        return email;
    }

    @Override
    public void setEmail(String email) {
        this.email = email;
    }
```

```
    @Override
    public String getPhone() {
        return phone;
    }

    @Override
    public void setPhone(String phone) {
        this.phone = phone;
    }

    @Override
    public String getAddress() {
        return address;
    }

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

    @Override
    public String getFaculty() {
        return faculty;
    }

    @Override
    public void setFaculty(String faculty) {
        this.faculty = faculty;
    }

    @Override
    public String getMajor() {
        return major;
    }

    @Override
    public void setMajor(String major) {
        this.major = major;
    }

    @Override
    public boolean isPsycho_pedagogical_module() {
        return psycho_pedagogical_module;
    }

    @Override
    public void setPsycho_pedagogical_module(boolean
psycho_pedagogical_module) {
        this.psycho_pedagogical_module = psycho_pedagogical_module;
    }

    @Override
    public List<ClassInfo> getClasses() {
        return classes;
    }

    @Override
    public void addClass(ClassInfo classInfo) {
        classes.add(classInfo);
    }
```

```
    @Override
    public ClassInfo getClassInfo(String className) {
        for (ClassInfo classInfo : classes) {
            if (classInfo.getClassName().equals(className)) {
                return classInfo;
            }
        }
        return null;
    }
}
```

Professor.java:

```
package Models;

import Interfaces.IProfessor;

public class Professor extends Person implements IProfessor {
    private String faculty;
    private String className;

    public Professor() {
        super(0, "", "");
        this.faculty = "";
        this.className = "";
    }

    public Professor(int id, String firstName, String lastName, String
faculty, String className) {
        super(id, firstName, lastName);
        this.faculty = faculty;
        this.className = className;
    }

    @Override
    public void displayInfo() {
        System.out.println("Professor ID: " + getId());
        System.out.println("Name: " + getFirstName() + " " +
getLastName());
        System.out.println("Faculty: " + faculty);
        System.out.println("Class Name: " + className);
    }

    @Override
    public int getId() {
        return super.getId();
    }

    @Override
    public void setId(int id) {
        super.setId(id);
    }

    @Override
    public String getFirstName() {
        return super.getFirstName();
```

```
    }

    @Override
    public void setFirstName(String firstName) {
        super.setFirstName(firstName);
    }

    @Override
    public String getLastName() {
        return super.getLastName();
    }

    @Override
    public void setLastName(String lastName) {
        super.setLastName(lastName);
    }

    @Override
    public String getFaculty() {
        return faculty;
    }

    @Override
    public void setFaculty(String faculty) {
        this.faculty = faculty;
    }

    @Override
    public String getClassName() {
        return className;
    }

    @Override
    public void setClassName(String className) {
        this.className = className;
    }
}
```

ClassInfo.java:

```
package Models;

import Interfaces.IClassInfo;
import java.util.ArrayList;
import java.util.List;

public class ClassInfo implements IClassInfo {
    private String className;
    private List<Float> grades;

    public ClassInfo(String className) {
        this.className = className;
        this.grades = new ArrayList<>();
    }

    @Override
    public String getClassName() {
        return className;
    }
```

```
    @Override
    public void setClassName(String className) {
        this.className = className;
    }

    @Override
    public List<Float> getGrades() {
        return grades;
    }

    @Override
    public void addGrade(float grade) {
        grades.add(grade);
    }

    @Override
    public float calculateAverageGrade() {
        float sum = 0;
        for (float grade : grades) {
            sum += grade;
        }
        return grades.size() > 0 ? sum / grades.size() : 0;
    }
}
```

StudentState.java:

```
package Models;

import Interfaces.IStudentState;
import java.util.ArrayList;
import java.util.List;

public class StudentState extends Student implements IStudentState {
    private List<ClassInfo> classes;

    public StudentState(int id, String first_name, String last_name, int
age, String gender, String email, String phone, String address, String
faculty, String major, boolean psycho_pedagogical_module) {
        super(id, first_name, last_name, age, gender, email, phone,
address, faculty, major, psycho_pedagogical_module);
        this.classes = new ArrayList<>();
    }

    @Override
    public List<ClassInfo> getClasses() {
        return classes;
    }

    @Override
    public void addClass(ClassInfo classInfo) {
        classes.add(classInfo);
    }

    @Override
    public ClassInfo getClassInfo(String className) {
        for (ClassInfo classInfo : classes) {
            if (classInfo.getClassName().equals(className)) {
```

```
                return classInfo;
            }
        }
        return null;
    }
}
```

ProfessorDAO.java:

```
package Models;

import Interfaces.IProfessorDAO;
import org.example.DatabaseUtil;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class ProfessorDAO implements IProfessorDAO {
    private Connection connection;

    public ProfessorDAO() {
        this.connection = DatabaseUtil.getConnection();
    }

    @Override
    public List<Professor> getAllProfessors() throws SQLException {
        List<Professor> professors = new ArrayList<>();
        String query = "SELECT * FROM professor";
        try (PreparedStatement stmt = connection.prepareStatement(query);
             ResultSet rs = stmt.executeQuery()) {
            while (rs.next()) {
                professors.add(new Professor(
                        rs.getInt("id"),
                        rs.getString("first_name"),
                        rs.getString("last_name"),
                        rs.getString("faculty"),
                        rs.getString("class_name")
                ));
            }
        }
        return professors;
    }
}
```

StudentDAO.java:

```
package Models;

import Interfaces.IStudentDAO;
import org.example.DatabaseUtil;

import java.sql.Connection;
import java.sql.PreparedStatement;
```

```
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class StudentDAO implements IStudentDAO {
    private Connection connection;

    public StudentDAO() {
        this.connection = DatabaseUtil.getConnection();
    }

    @Override
    public void addStudent(Student student) throws SQLException {
        String query = "INSERT INTO students (id, first_name, last_name,
age, gender, email, phone, address, faculty, major,
psycho_pedagogical_module) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
        try (PreparedStatement stmt = connection.prepareStatement(query)) {
            stmt.setInt(1, student.getId());
            stmt.setString(2, student.getFirst_name());
            stmt.setString(3, student.getLast_name());
            stmt.setInt(4, student.getAge());
            stmt.setString(5, student.getGender());
            stmt.setString(6, student.getEmail());
            stmt.setString(7, student.getPhone());
            stmt.setString(8, student.getAddress());
            stmt.setString(9, student.getFaculty());
            stmt.setString(10, student.getMajor());
            stmt.setBoolean(11, student.isPsycho_pedagogical_module());
            stmt.executeUpdate();
        }
    }

    @Override
    public Student getStudent(int id) throws SQLException {
        String query = "SELECT * FROM students WHERE id = ?";
        try (PreparedStatement stmt = connection.prepareStatement(query)) {
            stmt.setInt(1, id);
            ResultSet rs = stmt.executeQuery();
            if (rs.next()) {
                return new Student(
                        rs.getInt("id"),
                        rs.getString("first_name"),
                        rs.getString("last_name"),
                        rs.getInt("age"),
                        rs.getString("gender"),
                        rs.getString("email"),
                        rs.getString("phone"),
                        rs.getString("address"),
                        rs.getString("faculty"),
                        rs.getString("major"),
                        rs.getBoolean("psycho_pedagogical_module")
                );
            }
        }
        return null;
    }

    @Override
    public List<Student> getAllStudents() throws SQLException {
        List<Student> students = new ArrayList<>();
```

```
        String query = "SELECT * FROM students";
        try (PreparedStatement stmt = connection.prepareStatement(query);
             ResultSet rs = stmt.executeQuery()) {
            while (rs.next()) {
                students.add(new Student(
                        rs.getInt("id"),
                        rs.getString("first_name"),
                        rs.getString("last_name"),
                        rs.getInt("age"),
                        rs.getString("gender"),
                        rs.getString("email"),
                        rs.getString("phone"),
                        rs.getString("address"),
                        rs.getString("faculty"),
                        rs.getString("major"),
                        rs.getBoolean("psycho_pedagogical_module")
                ));
            }
        }
        return students;
    }

    @Override
    public void updateStudent(Student student) throws SQLException {
        String query = "UPDATE students SET first_name = ?, last_name = ?,
age = ?, gender = ?, email = ?, phone = ?, address = ?, faculty = ?, major
= ?, psycho_pedagogical_module = ? WHERE id = ?";
        try (PreparedStatement stmt = connection.prepareStatement(query)) {
            stmt.setString(1, student.getFirst_name());
            stmt.setString(2, student.getLast_name());
            stmt.setInt(3, student.getAge());
            stmt.setString(4, student.getGender());
            stmt.setString(5, student.getEmail());
            stmt.setString(6, student.getPhone());
            stmt.setString(7, student.getAddress());
            stmt.setString(8, student.getFaculty());
            stmt.setString(9, student.getMajor());
            stmt.setBoolean(10, student.isPsycho_pedagogical_module());
            stmt.setInt(11, student.getId());
            stmt.executeUpdate();
        }
    }

    @Override
    public void deleteStudent(int id) throws SQLException {
        String query = "DELETE FROM students WHERE id = ?";
        try (PreparedStatement stmt = connection.prepareStatement(query)) {
            stmt.setInt(1, id);
            stmt.executeUpdate();
            System.out.println("Deleted student with ID: " + id);
        }
    }

    @Override
    public List<Double> getStudentGrades(int studentId, String className)
throws SQLException {
        List<Double> grades = new ArrayList<>();
        String query = "SELECT grade FROM classes WHERE student_id = ? AND
class_name = ?";
        try (PreparedStatement stmt = connection.prepareStatement(query)) {
            stmt.setInt(1, studentId);
```

```
            stmt.setString(2, className);
            ResultSet rs = stmt.executeQuery();
            while (rs.next()) {
                grades.add(Double.parseDouble(rs.getString("grade")));
            }
        }
        return grades;
    }

    @Override
    public void addGrade(int studentId, String className, float grade)
throws SQLException {
        String query = "INSERT INTO classes (student_id, class_name, grade)
VALUES (?, ?, ?)";
        try (PreparedStatement stmt = connection.prepareStatement(query)) {
            stmt.setInt(1, studentId);
            stmt.setString(2, className);
            stmt.setFloat(3, grade);
            stmt.executeUpdate();
        }
    }

    @Override
    public List<ClassInfo> getAllGrades(int studentId) throws SQLException
{
        List<ClassInfo> classes = new ArrayList<>();
        String query = "SELECT class_name, grade FROM classes WHERE
student_id = ?";
        try (PreparedStatement stmt = connection.prepareStatement(query)) {
            stmt.setInt(1, studentId);
            ResultSet rs = stmt.executeQuery();
            while (rs.next()) {
                String className = rs.getString("class_name");
                float grade = rs.getFloat("grade");
                ClassInfo classInfo = findClassInfo(classes, className);
                if (classInfo == null) {
                    classInfo = new ClassInfo(className);
                    classes.add(classInfo);
                }
                classInfo.addGrade(grade);
            }
        }
        return classes;
    }

    private ClassInfo findClassInfo(List<ClassInfo> classes, String
className) {
        for (ClassInfo classInfo : classes) {
            if (classInfo.getClassName().equals(className)) {
                return classInfo;
            }
        }
        return null;
    }

    public List<String> getAllClasses() throws SQLException {
        List<String> classes = new ArrayList<>();
        String query = "SELECT DISTINCT class_name FROM classes";
        try (PreparedStatement stmt = connection.prepareStatement(query);
             ResultSet rs = stmt.executeQuery()) {
            while (rs.next()) {
```

```
                classes.add(rs.getString("class_name"));
            }
        }
        return classes;
    }

    @Override
    public void deleteGrade(int studentId, String className, float grade)
throws SQLException {
        String query = "DELETE FROM classes WHERE student_id = ? AND
class_name = ? AND grade = ?";
        try (PreparedStatement stmt = connection.prepareStatement(query)) {
            stmt.setInt(1, studentId);
            stmt.setString(2, className);
            stmt.setFloat(3, grade);
            stmt.executeUpdate();
        }
    }
}
```

DatabaseUtil.java:

```
package org.example;

import java.io.InputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.util.Properties;

public class DatabaseUtil {
    private static Connection connection;

    static {
        try (InputStream input =
DatabaseUtil.class.getClassLoader().getResourceAsStream("db.properties")) {
            Properties prop = new Properties();
            if (input == null) {
                throw new RuntimeException("Sorry, unable to find
db.properties");
            }
            prop.load(input);
            System.out.println("Properties file loaded successfully.");

            String url = prop.getProperty("db.url");
            String user = prop.getProperty("db.username");
            String password = prop.getProperty("db.password");

            Class.forName("org.postgresql.Driver");
            System.out.println("PostgreSQL driver loaded successfully.");

            connection = DriverManager.getConnection(url, user, password);
            System.out.println("Database connection established
successfully.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static Connection getConnection() {
```

```
        return connection;
    }
}
```

DataExporter.java:

```
package org.example;

import Models.Student;
import Models.StudentDAO;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.SerializationFeature;

import java.io.File;
import java.io.IOException;
import java.sql.SQLException;
import java.util.List;

public class DataExporter {

    public static void exportDataToJson(String filePath) throws
SQLException, IOException {
        StudentDAO studentDAO = new StudentDAO();
        List<Student> students = studentDAO.getAllStudents();

        ObjectMapper objectMapper = new ObjectMapper();
        objectMapper.enable(SerializationFeature.INDENT_OUTPUT);

        objectMapper.writeValue(new File(filePath), students);
    }

    public static void main(String[] args) {
        try {
            exportDataToJson("students.json");
            System.out.println("Data exported to students.json");
        } catch (SQLException | IOException e) {
            e.printStackTrace();
        }
    }
}
```

Main.java:

```
package org.example;

import Interfaces.IProfessorDAO;
import Interfaces.IStudentDAO;
import Models.*;

import java.sql.SQLException;
import java.util.List;
import java.util.Scanner;
```

```
public class Main {
    private static final String STUDENTS_FILE =
"C:/Users/Tudor/Desktop/MIP/ProiectPirauTudorIoanMaven/students.json";

    public static void main(String[] args) throws SQLException {
        try (Scanner scanner = new Scanner(System.in)) {

            IStudentDAO studentDAO = new StudentDAO();
            IProfessorDAO professorDAO = new ProfessorDAO();

            int option = 0;
            while (option != 10) {

System.out.println("====================================");
                System.out.println("Select an option:");
                System.out.println("1. Display all students");
                System.out.println("2. Add a new student");
                System.out.println("3. Delete a student by ID");
                System.out.println("4. Display all available classes");
                System.out.println("5. Display grades for a student in a
class");
                System.out.println("6. Add a grade for a student");
                System.out.println("7. Display all grades of a student in
all classes");
                System.out.println("8. Delete a grade for a student");
                System.out.println("9. Display all professors");
                System.out.println("10. Exit");

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

                option = scanner.nextInt();
                scanner.nextLine();

                switch (option) {
                    case 1:
                        List<Student> students =
studentDAO.getAllStudents();
                        System.out.println("All students in the
database:");
                        for (Student student : students) {
                            System.out.println(student.getId() + ": " +
student.getFirst_name() + " " + student.getLast_name() + ", " +
student.getAge() + " years old, " + student.getEmail());
                        }
                        break;
                    case 2:
                        System.out.print("Enter student ID: ");
                        int newStudentId = scanner.nextInt();
                        scanner.nextLine();

                        System.out.print("Enter first name: ");
                        String firstName = scanner.nextLine();

                        System.out.print("Enter last name: ");
                        String lastName = scanner.nextLine();

                        System.out.print("Enter age: ");
                        int age = scanner.nextInt();
                        scanner.nextLine();
```

```
                        System.out.print("Enter gender: ");
                        String gender = scanner.nextLine();

                        System.out.print("Enter email: ");
                        String email = scanner.nextLine();

                        System.out.print("Enter phone number: ");
                        String phoneNumber = scanner.nextLine();

                        System.out.print("Enter address: ");
                        String address = scanner.nextLine();

                        System.out.print("Enter faculty: ");
                        String faculty = scanner.nextLine();

                        System.out.print("Enter major: ");
                        String major = scanner.nextLine();

                        System.out.print("Is the student active
(true/false): ");
                        boolean isActive = scanner.nextBoolean();
                        scanner.nextLine();

                        Student newStudent = new Student(newStudentId,
firstName, lastName, age, gender, email, phoneNumber, address, faculty,
major, isActive);
                        studentDAO.addStudent(newStudent);
                        System.out.println("Added new student: " +
newStudent.getFirst_name() + " " + newStudent.getLast_name());
                        DataExporter.exportDataToJson(STUDENTS_FILE);
                        break;
                    case 3:
                        System.out.print("Enter student ID to delete: ");
                        int studentIdToDelete = scanner.nextInt();
                        studentDAO.deleteStudent(studentIdToDelete);
                        DataExporter.exportDataToJson(STUDENTS_FILE);
                        break;
                    case 4:
                        List<String> classesList =
studentDAO.getAllClasses();
                        System.out.println("All available classes:");
                        for (String className : classesList) {
                            System.out.println(className);
                        }
                        break;
                    case 5:
                        System.out.print("Enter student ID: ");
                        int studentId = scanner.nextInt();
                        scanner.nextLine();
                        System.out.print("Enter class name: ");
                        String className = scanner.nextLine();
                        List<Double> grades =
studentDAO.getStudentGrades(studentId, className);
                        System.out.println("Grades for student ID " +
studentId + " in class " + className + ":");
                        for (double grade : grades) {
                            System.out.println(grade);
                        }
                        break;
                    case 6:
                        System.out.print("Enter student ID: ");
```

```
                        int studentIdForGrade = scanner.nextInt();
                        scanner.nextLine();
                        System.out.print("Enter class name: ");
                        String classNameForGrade = scanner.nextLine();
                        System.out.print("Enter grade: ");
                        float grade = scanner.nextFloat();
                        scanner.nextLine();
                        studentDAO.addGrade(studentIdForGrade,
classNameForGrade, grade);
                        System.out.println("Added grade " + grade + " for
class " + classNameForGrade + " to student ID " + studentIdForGrade);
                        break;
                    case 7:
                        System.out.print("Enter student ID: ");
                        int studentIdForAllGrades = scanner.nextInt();
                        scanner.nextLine();
                        List<ClassInfo> classes =
studentDAO.getAllGrades(studentIdForAllGrades);
                        System.out.println("Grades for student ID " +
studentIdForAllGrades + " in all classes:");
                        for (ClassInfo classInfo : classes) {
                            System.out.println("Class: " +
classInfo.getClassName());
                            for (float grade2 : classInfo.getGrades()) {
                                System.out.println("Grade: " + grade2);
                            }
                        }
                        break;
                    case 8:
                        System.out.print("Enter student ID: ");
                        int studentIdForDeleteGrade = scanner.nextInt();
                        scanner.nextLine();
                        System.out.print("Enter class name: ");
                        String classNameForDeleteGrade =
scanner.nextLine();
                        System.out.print("Enter grade: ");
                        float gradeToDelete = scanner.nextFloat();
                        scanner.nextLine();
                        studentDAO.deleteGrade(studentIdForDeleteGrade,
classNameForDeleteGrade, gradeToDelete);
                        System.out.println("Deleted grade " + gradeToDelete
+ " for class " + classNameForDeleteGrade + " from student ID " +
studentIdForDeleteGrade);
                        break;
                    case 9:
                        List<Professor> professors =
professorDAO.getAllProfessors();
                        System.out.println("All professors in the
database:");
                        for (Professor professor : professors) {
                            System.out.println("Professor ID: " +
professor.getId());
                            System.out.println("Name: " +
professor.getFirstName() + " " + professor.getLastName());
                            System.out.println("Faculty: " +
professor.getFaculty());
                            System.out.println("Class Name: " +
professor.getClassName());
                            System.out.println();
                        }
                        break;
```

```
                    case 10:
                        System.out.println("Exiting...");
                        break;
                    default:
                        System.out.println("Invalid option");
                        break;
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

MainApp.java:

```
package org.example;

import Interfaces.IProfessorDAO;
import Interfaces.IStudentDAO;
import Models.*;
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.CheckBox;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

import java.io.IOException;
import java.sql.SQLException;
import java.util.List;

public class MainApp extends Application {

    private IStudentDAO studentDAO = new StudentDAO();
    private Stage primaryStage;
    private static final String STUDENTS_FILE =
"C:/Users/Tudor/Desktop/MIP/ProiectPirauTudorIoanMaven/students.json";

    public static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage primaryStage) {
        this.primaryStage = primaryStage;
        primaryStage.setTitle("Student Management System");

        primaryStage.setFullScreenExitHint("");

        try {
            DataExporter.exportDataToJson(STUDENTS_FILE);
        } catch (SQLException | IOException e) {
```

```
            e.printStackTrace();
        }

        VBox mainLayout = new VBox(20);
        mainLayout.setPadding(new Insets(20));
        mainLayout.setAlignment(Pos.CENTER);

        Label titleLabel = new Label("Student Management System");
        titleLabel.getStyleClass().add("title-label");

        VBox buttonLayout = new VBox(10);
        buttonLayout.setAlignment(Pos.CENTER);

        Button retrieveButton = new Button("Display all students");
        retrieveButton.setOnAction(e -> retrieveAllStudents());

        Button addButton = new Button("Add a new student");
        addButton.setOnAction(e -> addStudent());

        Button deleteButton = new Button("Delete a student by ID");
        deleteButton.setOnAction(e -> deleteStudent());

        Button displayClassesButton = new Button("Display all available
classes");
        displayClassesButton.setOnAction(e -> displayAllClasses());

        Button retrieveGradesButton = new Button("Display grades for a
student in a class");
        retrieveGradesButton.setOnAction(e -> retrieveStudentGrades());

        Button addGradeButton = new Button("Add a grade for a student");
        addGradeButton.setOnAction(e -> addGrade());

        Button displayGradesButton = new Button("Display all grades of a
student in all classes");
        displayGradesButton.setOnAction(e -> displayAllGrades());

        Button deleteGradeButton = new Button("Delete a grade for a
student");
        deleteGradeButton.setOnAction(e -> deleteGrade());

        Button retrieveProfessorsButton = new Button("Display all
professors");
        retrieveProfessorsButton.setOnAction(e -> retrieveAllProfessors());

        Button exitButton = new Button("Exit");
        exitButton.setOnAction(e -> primaryStage.close());

        buttonLayout.getChildren().addAll(retrieveButton, addButton,
deleteButton, displayClassesButton, retrieveGradesButton, addGradeButton,
displayGradesButton, deleteGradeButton, retrieveProfessorsButton,
exitButton);

        mainLayout.getChildren().addAll(titleLabel, buttonLayout);

        Scene scene = new Scene(mainLayout, 800, 600);
        scene.getStylesheets().add("/style.css");

        primaryStage.setScene(scene);
        primaryStage.setFullScreen(true);
        primaryStage.show();
```

```
    }

    private void addStudent() {
        Stage addStudentStage = new Stage();
        addStudentStage.setTitle("Add a New Student");

        addStudentStage.setFullScreenExitHint("");

        GridPane grid = new GridPane();
        grid.setPadding(new Insets(10, 10, 10, 10));
        grid.setVgap(8);
        grid.setHgap(10);
        grid.setAlignment(Pos.CENTER);

        Label idLabel = new Label("Student ID:");
        GridPane.setConstraints(idLabel, 0, 0);
        TextField idInput = new TextField();
        GridPane.setConstraints(idInput, 1, 0);

        Label firstNameLabel = new Label("First Name:");
        GridPane.setConstraints(firstNameLabel, 0, 1);
        TextField firstNameInput = new TextField();
        GridPane.setConstraints(firstNameInput, 1, 1);

        Label lastNameLabel = new Label("Last Name:");
        GridPane.setConstraints(lastNameLabel, 0, 2);
        TextField lastNameInput = new TextField();
        GridPane.setConstraints(lastNameInput, 1, 2);

        Label ageLabel = new Label("Age:");
        GridPane.setConstraints(ageLabel, 0, 3);
        TextField ageInput = new TextField();
        GridPane.setConstraints(ageInput, 1, 3);

        Label genderLabel = new Label("Gender:");
        GridPane.setConstraints(genderLabel, 0, 4);
        TextField genderInput = new TextField();
        GridPane.setConstraints(genderInput, 1, 4);

        Label emailLabel = new Label("Email:");
        GridPane.setConstraints(emailLabel, 0, 5);
        TextField emailInput = new TextField();
        GridPane.setConstraints(emailInput, 1, 5);

        Label phoneLabel = new Label("Phone:");
        GridPane.setConstraints(phoneLabel, 0, 6);
        TextField phoneInput = new TextField();
        GridPane.setConstraints(phoneInput, 1, 6);

        Label addressLabel = new Label("Address:");
        GridPane.setConstraints(addressLabel, 0, 7);
        TextField addressInput = new TextField();
        GridPane.setConstraints(addressInput, 1, 7);

        Label facultyLabel = new Label("Faculty:");
        GridPane.setConstraints(facultyLabel, 0, 8);
        TextField facultyInput = new TextField();
        GridPane.setConstraints(facultyInput, 1, 8);

        Label majorLabel = new Label("Major:");
        GridPane.setConstraints(majorLabel, 0, 9);
```

```
        TextField majorInput = new TextField();
        GridPane.setConstraints(majorInput, 1, 9);

        Label moduleLabel = new Label("Psycho-pedagogical Module:");
        GridPane.setConstraints(moduleLabel, 0, 10);
        CheckBox moduleInput = new CheckBox();
        GridPane.setConstraints(moduleInput, 1, 10);

        Button submitButton = new Button("Submit");
        GridPane.setConstraints(submitButton, 1, 11);
        submitButton.setOnAction(e -> {
            try {
                int studentId = Integer.parseInt(idInput.getText());
                Student newStudent = new Student(
                        studentId,
                        firstNameInput.getText(),
                        lastNameInput.getText(),
                        Integer.parseInt(ageInput.getText()),
                        genderInput.getText(),
                        emailInput.getText(),
                        phoneInput.getText(),
                        addressInput.getText(),
                        facultyInput.getText(),
                        majorInput.getText(),
                        moduleInput.isSelected()
                );
                studentDAO.addStudent(newStudent);

                DataExporter.exportDataToJson(STUDENTS_FILE);

                addStudentStage.close();
                primaryStage.setFullScreen(true);
                primaryStage.show();
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        });

        Button backButton = new Button("Back");
        GridPane.setConstraints(backButton, 0, 11);
        backButton.setOnAction(e -> {
            addStudentStage.close();
            primaryStage.setFullScreen(true);
            primaryStage.show();
        });

        grid.getChildren().addAll(idLabel, idInput, firstNameLabel,
firstNameInput, lastNameLabel, lastNameInput, ageLabel, ageInput,
genderLabel, genderInput, emailLabel, emailInput, phoneLabel, phoneInput,
addressLabel, addressInput, facultyLabel, facultyInput, majorLabel,
majorInput, moduleLabel, moduleInput, submitButton, backButton);

        Scene scene = new Scene(grid, 300, 400);
        scene.getStylesheets().add("/style.css");
        addStudentStage.setScene(scene);
        addStudentStage.setFullScreen(true);
        addStudentStage.show();
    }

    private void deleteStudent() {
        Stage deleteStudentStage = new Stage();
```

```
        deleteStudentStage.setTitle("Delete a Student");

        deleteStudentStage.setFullScreenExitHint("");

        GridPane grid = new GridPane();
        grid.setPadding(new Insets(10, 10, 10, 10));
        grid.setVgap(8);
        grid.setHgap(10);
        grid.setAlignment(Pos.CENTER);

        Label idLabel = new Label("Student ID:");
        GridPane.setConstraints(idLabel, 0, 0);
        TextField idInput = new TextField();
        GridPane.setConstraints(idInput, 1, 0);

        Button submitButton = new Button("Submit");
        GridPane.setConstraints(submitButton, 1, 1);
        submitButton.setOnAction(e -> {
            try {
                int studentId = Integer.parseInt(idInput.getText());
                studentDAO.deleteStudent(studentId);
                DataExporter.exportDataToJson(STUDENTS_FILE);
                deleteStudentStage.close();
                primaryStage.setFullScreen(true);
                primaryStage.show();
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        });

        Button backButton = new Button("Back");
        GridPane.setConstraints(backButton, 0, 1);
        backButton.setOnAction(e -> {
            deleteStudentStage.close();
            primaryStage.setFullScreen(true);
            primaryStage.show();
        });

        grid.getChildren().addAll(idLabel, idInput, submitButton,
backButton);

        Scene scene = new Scene(grid, 300, 200);
        scene.getStylesheets().add("/style.css");
        deleteStudentStage.setScene(scene);
        deleteStudentStage.setFullScreen(true);
        deleteStudentStage.show();
    }

    private void retrieveAllStudents() {
        Stage retrieveStudentsStage = new Stage();
        retrieveStudentsStage.setTitle("All Students");

        retrieveStudentsStage.setFullScreenExitHint("");

        GridPane grid = new GridPane();
        grid.setPadding(new Insets(10, 10, 10, 10));
        grid.setVgap(8);
        grid.setHgap(10);
        grid.setAlignment(Pos.CENTER);

        try {
```

```
            List<Student> students = studentDAO.getAllStudents();
            int row = 0;
            for (Student student : students) {
                Label studentLabel = new Label(student.getId() + ": " +
student.getFirst_name() + " " + student.getLast_name() + ", " +
student.getAge() + " years old, " + student.getEmail());
                GridPane.setConstraints(studentLabel, 0, row++);
                grid.getChildren().add(studentLabel);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        Button backButton = new Button("Back");
        backButton.setOnAction(e -> {
            retrieveStudentsStage.close();
            primaryStage.setFullScreen(true);
            primaryStage.show();
        });

        VBox vbox = new VBox(10);
        vbox.setPadding(new Insets(10));
        vbox.setAlignment(Pos.CENTER);
        vbox.getChildren().addAll(grid, backButton);

        Scene scene = new Scene(vbox, 400, 400);
        scene.getStylesheets().add("/style.css");
        retrieveStudentsStage.setScene(scene);
        retrieveStudentsStage.setFullScreen(true);
        retrieveStudentsStage.show();
    }

    private void displayAllClasses() {
        Stage displayClassesStage = new Stage();
        displayClassesStage.setTitle("Display All Classes");

        displayClassesStage.setFullScreenExitHint("");

        GridPane grid = new GridPane();
        grid.setPadding(new Insets(10, 10, 10, 10));
        grid.setVgap(8);
        grid.setHgap(10);
        grid.setAlignment(Pos.CENTER);

        try {
            List<String> classes = studentDAO.getAllClasses();
            int row = 0;
            for (String className : classes) {
                Label classLabel = new Label("Class: " + className);
                GridPane.setConstraints(classLabel, 0, row++);
                grid.getChildren().add(classLabel);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        Button backButton = new Button("Back");
        backButton.setOnAction(e -> {
            displayClassesStage.close();
            primaryStage.setFullScreen(true);
            primaryStage.show();
```

```
        });

        VBox vbox = new VBox(10);
        vbox.setPadding(new Insets(10));
        vbox.setAlignment(Pos.CENTER);
        vbox.getChildren().addAll(grid, backButton);

        Scene scene = new Scene(vbox, 400, 400);
        scene.getStylesheets().add("/style.css");
        displayClassesStage.setScene(scene);
        displayClassesStage.setFullScreen(true);
        displayClassesStage.show();
    }

    private void retrieveStudentGrades() {
        Stage retrieveGradesStage = new Stage();
        retrieveGradesStage.setTitle("Retrieve Student Grades");

        retrieveGradesStage.setFullScreenExitHint("");

        GridPane grid = new GridPane();
        grid.setPadding(new Insets(10, 10, 10, 10));
        grid.setVgap(8);
        grid.setHgap(10);
        grid.setAlignment(Pos.CENTER);

        Label idLabel = new Label("Student ID:");
        GridPane.setConstraints(idLabel, 0, 0);
        TextField idInput = new TextField();
        GridPane.setConstraints(idInput, 1, 0);

        Label classLabel = new Label("Class Name:");
        GridPane.setConstraints(classLabel, 0, 1);
        TextField classInput = new TextField();
        GridPane.setConstraints(classInput, 1, 1);

        Button submitButton = new Button("Submit");
        GridPane.setConstraints(submitButton, 1, 2);
        submitButton.setOnAction(e -> {
            try {
                int studentId = Integer.parseInt(idInput.getText());
                String className = classInput.getText();
                List<Double> grades =
studentDAO.getStudentGrades(studentId, className);

                Stage gradesStage = new Stage();
                gradesStage.setTitle("Grades for Student ID " + studentId +
" in Class " + className);

                gradesStage.setFullScreenExitHint("");

                GridPane gradesGrid = new GridPane();
                gradesGrid.setPadding(new Insets(10, 10, 10, 10));
                gradesGrid.setVgap(8);
                gradesGrid.setHgap(10);
                gradesGrid.setAlignment(Pos.CENTER);

                int row = 0;
                for (Double grade : grades) {
                    Label gradeLabel = new Label("Grade: " + grade);
                    GridPane.setConstraints(gradeLabel, 0, row++);
```

```
                    gradesGrid.getChildren().add(gradeLabel);
                }

                Button backButton = new Button("Back");
                GridPane.setConstraints(backButton, 0, row);
                backButton.setOnAction(event -> {
                    gradesStage.close();
                    primaryStage.setFullScreen(true);
                    primaryStage.show();
                });
                gradesGrid.getChildren().add(backButton);

                Scene gradesScene = new Scene(gradesGrid, 300, 200);
                gradesScene.getStylesheets().add("/style.css");
                gradesStage.setScene(gradesScene);
                gradesStage.setFullScreen(true);
                gradesStage.show();

                retrieveGradesStage.close();
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        });

        Button backButton = new Button("Back");
        GridPane.setConstraints(backButton, 0, 2);
        backButton.setOnAction(e -> {
            retrieveGradesStage.close();
            primaryStage.setFullScreen(true);
            primaryStage.show();
        });

        grid.getChildren().addAll(idLabel, idInput, classLabel, classInput,
submitButton, backButton);

        Scene scene = new Scene(grid, 300, 200);
        scene.getStylesheets().add("/style.css");
        retrieveGradesStage.setScene(scene);
        retrieveGradesStage.setFullScreen(true);
        retrieveGradesStage.show();
    }

    private void addGrade() {
        Stage addGradeStage = new Stage();
        addGradeStage.setTitle("Add Grade");

        addGradeStage.setFullScreenExitHint("");

        GridPane grid = new GridPane();
        grid.setPadding(new Insets(10, 10, 10, 10));
        grid.setVgap(8);
        grid.setHgap(10);
        grid.setAlignment(Pos.CENTER);

        Label idLabel = new Label("Student ID:");
        GridPane.setConstraints(idLabel, 0, 0);
        TextField idInput = new TextField();
        GridPane.setConstraints(idInput, 1, 0);

        Label classLabel = new Label("Class Name:");
        GridPane.setConstraints(classLabel, 0, 1);
```

```
        TextField classInput = new TextField();
        GridPane.setConstraints(classInput, 1, 1);

        Label gradeLabel = new Label("Grade:");
        GridPane.setConstraints(gradeLabel, 0, 2);
        TextField gradeInput = new TextField();
        GridPane.setConstraints(gradeInput, 1, 2);

        Button submitButton = new Button("Submit");
        GridPane.setConstraints(submitButton, 1, 3);
        submitButton.setOnAction(e -> {
            try {
                int studentId = Integer.parseInt(idInput.getText());
                String className = classInput.getText();
                float grade = Float.parseFloat(gradeInput.getText());
                studentDAO.addGrade(studentId, className, grade);
                addGradeStage.close();
                primaryStage.setFullScreen(true);
                primaryStage.show();
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        });

        Button backButton = new Button("Back");
        GridPane.setConstraints(backButton, 0, 3);
        backButton.setOnAction(e -> {
            addGradeStage.close();
            primaryStage.setFullScreen(true);
            primaryStage.show();
        });

        grid.getChildren().addAll(idLabel, idInput, classLabel, classInput,
gradeLabel, gradeInput, submitButton, backButton);

        Scene scene = new Scene(grid, 300, 200);
        scene.getStylesheets().add("/style.css");
        addGradeStage.setScene(scene);
        addGradeStage.setFullScreen(true);
        addGradeStage.show();
    }

    private void displayAllGrades() {
        Stage displayGradesStage = new Stage();
        displayGradesStage.setTitle("Display All Grades");

        displayGradesStage.setFullScreenExitHint("");

        GridPane grid = new GridPane();
        grid.setPadding(new Insets(10, 10, 10, 10));
        grid.setVgap(8);
        grid.setHgap(10);
        grid.setAlignment(Pos.CENTER);

        Label idLabel = new Label("Student ID:");
        GridPane.setConstraints(idLabel, 0, 0);
        TextField idInput = new TextField();
        GridPane.setConstraints(idInput, 1, 0);

        Button submitButton = new Button("Submit");
        GridPane.setConstraints(submitButton, 1, 1);
```

```
        submitButton.setOnAction(e -> {
            try {
                int studentId = Integer.parseInt(idInput.getText());
                List<ClassInfo> classes =
studentDAO.getAllGrades(studentId);

                Stage gradesStage = new Stage();
                gradesStage.setTitle("Grades for Student ID " + studentId);

                gradesStage.setFullScreenExitHint("");

                GridPane gradesGrid = new GridPane();
                gradesGrid.setPadding(new Insets(10, 10, 10, 10));
                gradesGrid.setVgap(8);
                gradesGrid.setHgap(10);
                gradesGrid.setAlignment(Pos.CENTER);

                int row = 0;
                for (ClassInfo classInfo : classes) {
                    Label classLabel = new Label("Class: " +
classInfo.getClassName());
                    GridPane.setConstraints(classLabel, 0, row++);
                    gradesGrid.getChildren().add(classLabel);
                    for (float grade : classInfo.getGrades()) {
                        Label gradeLabel = new Label("Grade: " + grade);
                        GridPane.setConstraints(gradeLabel, 0, row++);
                        gradesGrid.getChildren().add(gradeLabel);
                    }
                }

                Button backButton = new Button("Back");
                GridPane.setConstraints(backButton, 0, row);
                backButton.setOnAction(event -> {
                    gradesStage.close();
                    primaryStage.setFullScreen(true);
                    primaryStage.show();
                });
                gradesGrid.getChildren().add(backButton);

                Scene gradesScene = new Scene(gradesGrid, 400, 400);
                gradesScene.getStylesheets().add("/style.css");
                gradesStage.setScene(gradesScene);
                gradesStage.setFullScreen(true);
                gradesStage.show();

                displayGradesStage.close();
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        });

        Button backButton = new Button("Back");
        GridPane.setConstraints(backButton, 0, 1);
        backButton.setOnAction(e -> {
            displayGradesStage.close();
            primaryStage.setFullScreen(true);
            primaryStage.show();
        });

        HBox buttonBox = new HBox(10);
        buttonBox.setAlignment(Pos.CENTER);
```

```
        buttonBox.getChildren().addAll(backButton, submitButton);
        GridPane.setConstraints(buttonBox, 1, 1);

        grid.getChildren().addAll(idLabel, idInput, buttonBox);

        Scene scene = new Scene(grid, 300, 200);
        scene.getStylesheets().add("/style.css");
        displayGradesStage.setScene(scene);
        displayGradesStage.setFullScreen(true);
        displayGradesStage.show();
    }

    private void deleteGrade() {
        Stage deleteGradeStage = new Stage();
        deleteGradeStage.setTitle("Delete Grade");

        deleteGradeStage.setFullScreenExitHint("");

        GridPane grid = new GridPane();
        grid.setPadding(new Insets(10, 10, 10, 10));
        grid.setVgap(8);
        grid.setHgap(10);
        grid.setAlignment(Pos.CENTER);

        Label idLabel = new Label("Student ID:");
        GridPane.setConstraints(idLabel, 0, 0);
        TextField idInput = new TextField();
        GridPane.setConstraints(idInput, 1, 0);

        Label classLabel = new Label("Class Name:");
        GridPane.setConstraints(classLabel, 0, 1);
        TextField classInput = new TextField();
        GridPane.setConstraints(classInput, 1, 1);

        Label gradeLabel = new Label("Grade:");
        GridPane.setConstraints(gradeLabel, 0, 2);
        TextField gradeInput = new TextField();
        GridPane.setConstraints(gradeInput, 1, 2);

        Button submitButton = new Button("Submit");
        GridPane.setConstraints(submitButton, 1, 3);
        submitButton.setOnAction(e -> {
            try {
                int studentId = Integer.parseInt(idInput.getText());
                String className = classInput.getText();
                float grade = Float.parseFloat(gradeInput.getText());
                studentDAO.deleteGrade(studentId, className, grade);
                deleteGradeStage.close();
                primaryStage.setFullScreen(true);
                primaryStage.show();
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        });

        Button backButton = new Button("Back");
        GridPane.setConstraints(backButton, 0, 3);
        backButton.setOnAction(e -> {
            deleteGradeStage.close();
            primaryStage.setFullScreen(true);
            primaryStage.show();
```

```
        });

        grid.getChildren().addAll(idLabel, idInput, classLabel, classInput,
gradeLabel, gradeInput, submitButton, backButton);

        Scene scene = new Scene(grid, 300, 200);
        scene.getStylesheets().add("/style.css");
        deleteGradeStage.setScene(scene);
        deleteGradeStage.setFullScreen(true);
        deleteGradeStage.show();
    }

    private void retrieveAllProfessors() {
        Stage retrieveProfessorsStage = new Stage();
        retrieveProfessorsStage.setTitle("All Professors");

        retrieveProfessorsStage.setFullScreenExitHint("");

        GridPane grid = new GridPane();
        grid.setPadding(new Insets(10, 10, 10, 10));
        grid.setVgap(8);
        grid.setHgap(10);
        grid.setAlignment(Pos.CENTER);

        try {
            IProfessorDAO professorDAO = new ProfessorDAO();
            List<Professor> professors = professorDAO.getAllProfessors();
            int row = 0;
            for (Professor professor : professors) {
                Label professorLabel = new Label("Professor ID: " +
professor.getId() + ", Name: " + professor.getFirstName() + " " +
professor.getLastName() + ", Faculty: " + professor.getFaculty() + ", Class
Name: " + professor.getClassName());
                GridPane.setConstraints(professorLabel, 0, row++);
                grid.getChildren().add(professorLabel);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        Button backButton = new Button("Back");
        backButton.setOnAction(e -> {
            retrieveProfessorsStage.close();
            primaryStage.setFullScreen(true);
            primaryStage.show();
        });

        VBox vbox = new VBox(10);
        vbox.setPadding(new Insets(10));
        vbox.setAlignment(Pos.CENTER);
        vbox.getChildren().addAll(grid, backButton);

        Scene scene = new Scene(vbox, 400, 400);
        scene.getStylesheets().add("/style.css");
        retrieveProfessorsStage.setScene(scene);
        retrieveProfessorsStage.setFullScreen(true);
        retrieveProfessorsStage.show();
    }
}
```

Teste:

PersonTest:

```
package Models;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertNotEquals;

class PersonTest {

    @Test
    public void getIdFromPersonTest() {
        Person person = new Person(1, "Ion", "Popescu") {
            @Override
            public void displayInfo() {
            }
        };
        assertEquals(1, person.getId());
    }

    @Test
    public void checkBugIfPersonIdIsIncorrectTest() {
        Person person = new Person(1, "Ion", "Popescu") {
            @Override
            public void displayInfo() {
            }
        };
        assertNotEquals(2, person.getId());
    }

    @Test
    public void getFirstNameFromPersonTest() {
        Person person = new Person(1, "Ion", "Popescu") {
            @Override
            public void displayInfo() {
            }
        };
        assertEquals("Ion", person.getFirstName());
    }

    @Test
    public void checkBugIfPersonFirstNameIsIncorrectTest() {
        Person person = new Person(1, "Ion", "Popescu") {
            @Override
            public void displayInfo() {
            }
        };
        assertNotEquals("Jane", person.getFirstName());
    }

    @Test
    public void getLastNameFromPersonTest() {
        Person person = new Person(1, "Ion", "Popescu") {
            @Override
            public void displayInfo() {
            }
        };
```

```
        assertEquals("Popescu", person.getLastName());
    }

    @Test
    public void checkBugIfPersonLastNameIsIncorrectTest() {
        Person person = new Person(1, "Ion", "Popescu") {
            @Override
            public void displayInfo() {
            }
        };
        assertNotEquals("Smith", person.getLastName());
    }

    @Test
    public void setIdForPersonTest() {
        Person person = new Person(1, "Ion", "Popescu") {
            @Override
            public void displayInfo() {
            }
        };
        person.setId(2);
        assertEquals(2, person.getId());
    }

    @Test
    public void setFirstNameForPersonTest() {
        Person person = new Person(1, "Ion", "Popescu") {
            @Override
            public void displayInfo() {
            }
        };
        person.setFirstName("Jane");
        assertEquals("Jane", person.getFirstName());
    }

    @Test
    public void setLastNameForPersonTest() {
        Person person = new Person(1, "Ion", "Popescu") {
            @Override
            public void displayInfo() {
            }
        };
        person.setLastName("Smith");
        assertEquals("Smith", person.getLastName());
    }
}
```

StudentTest:

```
package Models;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertNotEquals;

class StudentTest {

    @Test
```

```
    public void getFirstNameFromStudentTest() {
        Student student = new Student(1, "Mihai", "Dorin", 20, "Masculin",
"Mihai.Dorin@gmail.com", "1234567890", "Lunga 123", "Matematica",
"Informatica", true);
        assertEquals("Mihai", student.getFirst_name());
    }

    @Test
    public void checkBugIfStudentFirstNameIsIncorrectTest() {
        Student student = new Student(1, "Mihai", "Dorin", 20, "Masculin",
"Mihai.Dorin@gmail.com", "1234567890", "Lunga 123", "Matematica",
"Informatica", true);
        assertNotEquals("Jane", student.getFirst_name());
    }

    @Test
    public void getLastNameFromStudentTest() {
        Student student = new Student(1, "Mihai", "Dorin", 20, "Masculin",
"Mihai.Dorin@gmail.com", "1234567890", "Lunga 123", "Matematica",
"Informatica", true);
        assertEquals("Dorin", student.getLast_name());
    }

    @Test
    public void checkBugIfStudentLastNameIsIncorrectTest() {
        Student student = new Student(1, "Mihai", "Dorin", 20, "Masculin",
"Mihai.Dorin@gmail.com", "1234567890", "Lunga 123", "Matematica",
"Informatica", true);
        assertNotEquals("Smith", student.getLast_name());
    }

    @Test
    public void getAgeFromStudentTest() {
        Student student = new Student(1, "Mihai", "Dorin", 20, "Masculin",
"Mihai.Dorin@gmail.com", "1234567890", "Lunga 123", "Matematica",
"Informatica", true);
        assertEquals(20, student.getAge());
    }

    @Test
    public void checkBugIfStudentAgeIsIncorrectTest() {
        Student student = new Student(1, "Mihai", "Dorin", 20, "Masculin",
"Mihai.Dorin@gmail.com", "1234567890", "Lunga 123", "Matematica",
"Informatica", true);
        assertNotEquals(25, student.getAge());
    }

    @Test
    public void getEmailFromStudentTest() {
        Student student = new Student(1, "Mihai", "Dorin", 20, "Masculin",
"Mihai.Dorin@gmail.com", "1234567890", "Lunga 123", "Matematica",
"Informatica", true);
        assertEquals("Mihai.Dorin@gmail.com", student.getEmail());
    }

    @Test
    public void checkBugIfStudentEmailIsIncorrectTest() {
        Student student = new Student(1, "Mihai", "Dorin", 20, "Masculin",
"Mihai.Dorin@gmail.com", "1234567890", "Lunga 123", "Matematica",
"Informatica", true);
        assertNotEquals("jane.Dorin@gmail.com", student.getEmail());
```

```
    }
}
```

ProfessorTest:

```
package Models;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertNotEquals;

class ProfessorTest {

    @Test
    public void getIdFromProfessorTest() {
        Professor professor = new Professor(1, "Ion", "Popescu",
"Informatica", "Programare") {
            @Override
            public void displayInfo() {
            }
        };
        assertEquals(1, professor.getId());
    }

    @Test
    public void checkBugIfProfessorIdIsIncorrectTest() {
        Professor professor = new Professor(1, "Ion", "Popescu",
"Informatica", "Programare") {
            @Override
            public void displayInfo() {
            }
        };
        assertNotEquals(2, professor.getId());
    }

    @Test
    public void getFirstNameFromProfessorTest() {
        Professor professor = new Professor(1, "Ion", "Popescu",
"Informatica", "Programare") {
            @Override
            public void displayInfo() {
            }
        };
        assertEquals("Ion", professor.getFirstName());
    }

    @Test
    public void checkBugIfProfessorFirstNameIsIncorrectTest() {
        Professor professor = new Professor(1, "Ion", "Popescu",
"Informatica", "Programare") {
            @Override
            public void displayInfo() {
            }
        };
        assertNotEquals("Jane", professor.getFirstName());
    }

    @Test
```

```
    public void getLastNameFromProfessorTest() {
        Professor professor = new Professor(1, "Ion", "Popescu",
"Informatica", "Programare") {
            @Override
            public void displayInfo() {
            }
        };
        assertEquals("Popescu", professor.getLastName());
    }

    @Test
    public void checkBugIfProfessorLastNameIsIncorrectTest() {
        Professor professor = new Professor(1, "Ion", "Popescu",
"Informatica", "Programare") {
            @Override
            public void displayInfo() {
            }
        };
        assertNotEquals("Smith", professor.getLastName());
    }

    @Test
    public void setIdForProfessorTest() {
        Professor professor = new Professor(1, "Ion", "Popescu",
"Informatica", "Programare") {
            @Override
            public void displayInfo() {
            }
        };
        professor.setId(2);
        assertEquals(2, professor.getId());
    }

    @Test
    public void setFirstNameForProfessorTest() {
        Professor professor = new Professor(1, "Ion", "Popescu",
"Informatica", "Programare") {
            @Override
            public void displayInfo() {
            }
        };
        professor.setFirstName("Jane");
        assertEquals("Jane", professor.getFirstName());
    }

    @Test
    public void setLastNameForProfessorTest() {
        Professor professor = new Professor(1, "Ion", "Popescu",
"Informatica", "Programare") {
            @Override
            public void displayInfo() {
            }
        };
        professor.setLastName("Smith");
        assertEquals("Smith", professor.getLastName());
    }
}
```

ClassInfoTest:

```
package Models;

import org.junit.jupiter.api.Test;

import java.util.List;

import static org.junit.jupiter.api.Assertions.*;

class ClassInfoTest {

    @Test
    public void testGetClassName() {
        ClassInfo classInfo = new ClassInfo("Matematica");
        assertEquals("Matematica", classInfo.getClassName());
    }

    @Test
    public void testSetClassName() {
        ClassInfo classInfo = new ClassInfo("Matematica");
        classInfo.setClassName("Science");
        assertEquals("Science", classInfo.getClassName());
    }

    @Test
    public void testGetGrades() {
        ClassInfo classInfo = new ClassInfo("Matematica");
        List<Float> grades = classInfo.getGrades();
        assertNotNull(grades);
    }

    @Test
    public void testAddGrade() {
        ClassInfo classInfo = new ClassInfo("Matematica");
        classInfo.addGrade(95.0f);
        List<Float> grades = classInfo.getGrades();
        assertTrue(grades.contains(95.0f));
    }

    @Test
    public void testCalculateAverageGrade() {
        ClassInfo classInfo = new ClassInfo("Matematica");
        classInfo.addGrade(90.0f);
        classInfo.addGrade(80.0f);
        assertEquals(85.0f, classInfo.calculateAverageGrade());
    }
}
```

StudentStateTest:

```
package Models;

import org.junit.jupiter.api.Test;

import java.util.List;

import static org.junit.jupiter.api.Assertions.*;

class StudentStateTest {
```

```
    @Test
    public void testGetClasses() {
        StudentState studentState = new StudentState(1, "Mihai", "Dorin",
20, "Masculin", "Mihai.Dorin@example.com", "1234567890", "Lunga 123",
"Matematica", "Informatica", true);
        List<ClassInfo> classes = studentState.getClasses();
        assertNotNull(classes);
    }

    @Test
    public void testAddClass() {
        StudentState studentState = new StudentState(1, "Mihai", "Dorin",
20, "Masculin", "Mihai.Dorin@example.com", "1234567890", "Lunga 123",
"Matematica", "Informatica", true);
        ClassInfo classInfo = new ClassInfo("Math");
        studentState.addClass(classInfo);
        List<ClassInfo> classes = studentState.getClasses();
        assertTrue(classes.contains(classInfo));
    }

    @Test
    public void testGetClassInfo() {
        StudentState studentState = new StudentState(1, "Mihai", "Dorin",
20, "Masculin", "Mihai.Dorin@example.com", "1234567890", "Lunga 123",
"Matematica", "Informatica", true);
        ClassInfo classInfo = new ClassInfo("Chimie");
        studentState.addClass(classInfo);
        ClassInfo retrievedClassInfo = studentState.getClassInfo("Chimie");
        assertNotNull(retrievedClassInfo);
        assertEquals("Chimie", retrievedClassInfo.getClassName());
    }
}
```

ProfessorDAOTest:

```
package Models;

import org.example.DatabaseUtil;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import java.sql.Connection;
import java.sql.SQLException;
import java.util.List;

import static org.junit.jupiter.api.Assertions.assertEquals;

class ProfessorDAOTest {

    private static Connection connection;
    private ProfessorDAO professorDAO;

    @BeforeAll
    static void setupDatabase() {
        connection = DatabaseUtil.getConnection();
    }
```

```
    @BeforeEach
    void setup() {
        professorDAO = new ProfessorDAO();
    }

    @AfterEach
    void tearDown() throws SQLException {
        connection.createStatement().execute("DELETE FROM professor");
    }

    @Test
    void testGetAllProfessors() throws SQLException {
        Professor professor1 = new Professor(1, "Ion", "Popescu",
"Informatica", "Programare");
        Professor professor2 = new Professor(2, "Maria", "Ionescu",
"Matematica", "Algebra");

        connection.createStatement().execute("INSERT INTO professor (id,
first_name, last_name, faculty, class_name) VALUES (1, 'Ion', 'Popescu',
'Informatica', 'Programare')");
        connection.createStatement().execute("INSERT INTO professor (id,
first_name, last_name, faculty, class_name) VALUES (2, 'Maria', 'Ionescu',
'Matematica', 'Algebra')");

        List<Professor> professors = professorDAO.getAllProfessors();
        assertEquals(2, professors.size());
    }
}
```

StudentDAOTest:

```
package Models;

import org.example.DatabaseUtil;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import java.sql.Connection;
import java.sql.SQLException;
import java.util.List;

import static org.junit.jupiter.api.Assertions.*;

class StudentDAOTest {

    private static Connection connection;
    private StudentDAO studentDAO;

    @BeforeAll
    static void setupDatabase() {
        connection = DatabaseUtil.getConnection();
    }

    @BeforeEach
    void setup() {
        studentDAO = new StudentDAO();
    }
```

```
    @AfterEach
    void tearDown() throws SQLException {
        connection.createStatement().execute("DELETE FROM students");
        connection.createStatement().execute("DELETE FROM classes");
    }

    @Test
    void testAddStudent() throws SQLException {
        Student student = new Student(1, "Mihai", "Dorin", 20, "Masculin",
"Mihai.Dorin@gmail.com", "0722222222", "Lunga 123", "Matematica",
"Informatica", true);
        studentDAO.addStudent(student);
        Student retrievedStudent = studentDAO.getStudent(1);
        assertNotNull(retrievedStudent);
        assertEquals("Mihai", retrievedStudent.getFirst_name());
    }

    @Test
    void testGetStudent() throws SQLException {
        Student student = new Student(1, "Mihai", "Dorin", 20, "Masculin",
"Mihai.Dorin@gmail.com", "0722222222", "Lunga 123", "Matematica",
"Informatica", true);
        studentDAO.addStudent(student);
        Student retrievedStudent = studentDAO.getStudent(1);
        assertNotNull(retrievedStudent);
        assertEquals("Mihai", retrievedStudent.getFirst_name());
    }

    @Test
    void testGetAllStudents() throws SQLException {
        Student student1 = new Student(1, "Mihai", "Dorin", 20, "Masculin",
"Mihai.Dorin@gmail.com", "0722222222", "Lunga 123", "Matematica",
"Informatica", true);
        Student student2 = new Student(2, "Jane", "Smith", 22,
"FeMasculin", "jane.smith@gmail.com", "0987654321", "456 Main St",
"Science", "Biology", false);
        studentDAO.addStudent(student1);
        studentDAO.addStudent(student2);
        List<Student> students = studentDAO.getAllStudents();
        assertEquals(2, students.size());
    }

    @Test
    void testUpdateStudent() throws SQLException {
        Student student = new Student(1, "Mihai", "Dorin", 20, "Masculin",
"Mihai.Dorin@gmail.com", "0722222222", "Lunga 123", "Matematica",
"Informatica", true);
        studentDAO.addStudent(student);
        student.setFirst_name("Mihainy");
        studentDAO.updateStudent(student);
        Student updatedStudent = studentDAO.getStudent(1);
        assertEquals("Mihainy", updatedStudent.getFirst_name());
    }

    @Test
    void testDeleteStudent() throws SQLException {
        Student student = new Student(1, "Mihai", "Dorin", 20, "Masculin",
"Mihai.Dorin@gmail.com", "0722222222", "Lunga 123", "Matematica",
"Informatica", true);
        studentDAO.addStudent(student);
```

```
        studentDAO.deleteStudent(1);
        Student deletedStudent = studentDAO.getStudent(1);
        assertNull(deletedStudent);
    }

    @Test
    void testGetStudentGrades() throws SQLException {
        Student student = new Student(1, "Mihai", "Dorin", 20, "Masculin",
"Mihai.Dorin@gmail.com", "0722222222", "Lunga 123", "Matematica",
"Informatica", true);
        studentDAO.addStudent(student);
        studentDAO.addGrade(1, "Math", 95.0f);
        List<Double> grades = studentDAO.getStudentGrades(1, "Math");
        assertEquals(1, grades.size());
        assertEquals(95.0, grades.get(0));
    }

    @Test
    void testAddGrade() throws SQLException {
        Student student = new Student(1, "Mihai", "Dorin", 20, "Masculin",
"Mihai.Dorin@gmail.com", "0722222222", "Lunga 123", "Matematica",
"Informatica", true);
        studentDAO.addStudent(student);
        studentDAO.addGrade(1, "Math", 95.0f);
        List<Double> grades = studentDAO.getStudentGrades(1, "Math");
        assertTrue(grades.contains(95.0));
    }

    @Test
    void testGetAllGrades() throws SQLException {
        Student student = new Student(1, "Mihai", "Dorin", 20, "Masculin",
"Mihai.Dorin@gmail.com", "0722222222", "Lunga 123", "Matematica",
"Informatica", true);
        studentDAO.addStudent(student);
        studentDAO.addGrade(1, "Math", 95.0f);
        studentDAO.addGrade(1, "Science", 85.0f);
        List<ClassInfo> classes = studentDAO.getAllGrades(1);
        assertEquals(2, classes.size());
    }

    @Test
    void testDeleteGrade() throws SQLException {
        Student student = new Student(1, "Mihai", "Dorin", 20, "Masculin",
"Mihai.Dorin@gmail.com", "0722222222", "Lunga 123", "Matematica",
"Informatica", true);
        studentDAO.addStudent(student);
        studentDAO.addGrade(1, "Math", 95.0f);
        studentDAO.deleteGrade(1, "Math", 95.0f);
        List<Double> grades = studentDAO.getStudentGrades(1, "Math");
        assertFalse(grades.contains(95.0));
    }
}
```

Crearea bazei de date în pgAdmin4:

CREATE TABLE students (

```
    id SERIAL PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    age INT,
    gender VARCHAR(10),
    email VARCHAR(100),
    phone VARCHAR(20),
    address VARCHAR(255),
    faculty VARCHAR(100),
    major VARCHAR(100),
    psycho_pedagogical_module BOOLEAN
);

CREATE TABLE classes (
    id SERIAL PRIMARY KEY,
    student_id INT REFERENCES students(id),
    class_name VARCHAR(100),
    grade FLOAT
);

CREATE TABLE professor (
    id SERIAL PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    faculty VARCHAR(100) NOT NULL,
    class_name VARCHAR(100) NOT NULL
);
```

INSERT INTO students (first_name, last_name, age, gender, email, phone, address, faculty, major, psycho_pedagogical_module) VALUES

('Ion', 'Popescu', 20, 'Masculin', 'ion.popescu@gmail.ro', '0712-345-678', 'Strada Principală 123', 'Științe', 'Biologie', true),

('Maria', 'Ionescu', 22, 'Feminin', 'maria.ionescu@gmail.ro', '0723-456-789', 'Strada Ulmului 456', 'Arte', 'Istorie', false),

('Ana', 'Georgescu', 21, 'Feminin', 'ana.georgescu@gmail.ro', '0734-567-890', 'Strada Stejarului 789', 'Inginerie', 'Informatică', true),

('Mihai', 'Dumitrescu', 23, 'Masculin', 'mihai.dumitrescu@gmail.ro', '0745-678-901', 'Strada Pinului 101', 'Afaceri', 'Marketing', false),

('George', 'Marinescu', 20, 'Masculin', 'george.marinescu@gmail.ro', '0756-789-012', 'Strada Arțarului 202', 'Științe', 'Chimie', true),

('Elena', 'Stanescu', 22, 'Feminin', 'elena.stanescu@gmail.ro', '0767-890-123', 'Strada Mesteacănului 303', 'Arte', 'Literatură', false),

('Ioana', 'Petrescu', 21, 'Feminin', 'ioana.petrescu@gmail.ro', '0778-901-234', 'Strada Cedrului 404', 'Inginerie', 'Inginerie Mecanică', true),

('Vasile', 'Radu', 23, 'Masculin', 'vasile.radu@gmail.ro', '0789-012-345', 'Strada Molidului 505', 'Afaceri', 'Finanțe', false),

('Gabriela', 'Constantinescu', 20, 'Feminin', 'gabriela.constantinescu@gmail.ro', '0790-123-456', 'Strada Salciei 606', 'Științe', 'Fizică', true),

('Florin', 'Andrei', 22, 'Masculin', 'florin.andrei@gmail.ro', '0701-234-567', 'Strada Frasinului 707', 'Arte', 'Filosofie', false);

INSERT INTO classes (student_id, class_name, grade) VALUES

(1, 'Biologie', 9.5),

(1, 'Chimie', 8.7),

(1, 'Fizică', 9.0),

(2, 'Istorie', 7.5),

(2, 'Literatură', 8.0),

(2, 'Filosofie', 7.8),

(3, 'Informatică', 9.2),

(3, 'Matematică', 8.9),

(3, 'Fizică', 9.1),

(4, 'Marketing', 7.4),

(4, 'Finanțe', 8.3),

(4, 'Management', 7.9),

(5, 'Chimie', 8.5),

(5, 'Biologie', 9.3),

(5, 'Fizică', 8.8),

(6, 'Literatură', 7.6),

(6, 'Istorie', 8.1),

(6, 'Filosofie', 7.7),

(7, 'Inginerie Mecanică', 9.4),

(7, 'Termodinamică', 8.6),

(7, 'Matematică', 9.0),

(8, 'Finanțe', 7.8),

(8, 'Marketing', 8.2),

(8, 'Management', 7.5),

(9, 'Fizică', 9.1),

(9, 'Chimie', 8.9),

(9, 'Biologie', 9.2),

(10, 'Filosofie', 7.3),

(10, 'Istorie', 8.4),

(10, 'Literatură', 7.9);

INSERT INTO professor (first_name, last_name, faculty, class_name) VALUES

('Ion', 'Popescu', 'Matematică', 'Algebră'),

('Maria', 'Ionescu', 'Fizică', 'Mecanică Cuantică'),

('George', 'Georgescu', 'Informatică', 'Structuri de Date'),

('Elena', 'Marinescu', 'Chimie', 'Chimie Organică'),

('Mihai', 'Dumitrescu', 'Biologie', 'Genetică'),

('Ana', 'Stanescu', 'Istorie', 'Istorie Modernă'),

('Vasile', 'Radu', 'Filosofie', 'Etică'),

('Ioana', 'Nistor', 'Literatură', 'Literatură Română'),

('Florin', 'Petrescu', 'Inginerie', 'Termodinamică'),

('Gabriela', 'Constantinescu', 'Economie', 'Microeconomie');

Laborator 1:

Output:

getStudent(1):

```
Student student = studentDAO.getStudent(1);
System.out.println(student.getFirst_name());
```

getAllStudents():

```
List<Student> students = studentDAO.getAllStudents();
for (Student student : students) {
    System.out.println(student.getFirst_name());
}
```

// Exemplu de output: Mihai, Iustin

getStudentGrades(1, "Matematica"):

```
List<Double> grades = studentDAO.getStudentGrades(1, "Matematica");
for (Double grade : grades) {
    System.out.println(grade);
}
```

// Exemplu de output: 9.5

getAllGrades(1):

```
List<ClassInfo> classes = studentDAO.getAllGrades(1);
for (ClassInfo classInfo : classes) {
    System.out.println(classInfo.getClassName());
    for (float grade : classInfo.getGrades()) {
        System.out.println(grade);
    }
}
```

// Exemplu de output: Matematica 9, Chimie 8.5

getAllProfessors():

```
List<Professor> professors = professorDAO.getAllProfessors();
for (Professor professor : professors) {
   System.out.println(professor.getFirstName() + " " + professor.getLastName());
}
```

// Exemplu de output: Ion Mihai

Tipuri de valori folosite:

int: folosit pentru ID-uri și alte valori numerice.

float: folosit pentru note.

boolean: folosit pentru a indica starea (de exemplu, dacă un student este activ).

String: folosit pentru nume, email-uri, adrese etc.

List<Student>: folosit pentru a stoca liste de obiecte Student.

List<Double>: folosit pentru a stoca liste de note.

List<ClassInfo>: folosit pentru a stoca informații despre clase și notele aferente.

Exemple de Utilizare:

```
int studentId = 1;
float grade = 9.5f;
boolean isActive = true;

String firstName = "Mihai";
String email = "Mihai.Dorin@gmail.com";
List<Student> students = studentDAO.getAllStudents();
List<Double> grades = studentDAO.getStudentGrades(1, "Math");
List<ClassInfo> classes = studentDAO.getAllGrades(1);
```

Funcții folosite în proiect:

- addStudent(Student student)
- getStudent(int studentId)
- getAllStudents()
- updateStudent(Student student)
- deleteStudent(int studentId)
- addGrade(int studentId, String className, float grade)
- getStudentGrades(int studentId, String className)
- getAllGrades(int studentId)
- deleteGrade(int studentId, String className, float grade)
- getAllProfessors()

Laborator 2:

Input:

```
System.out.print("Enter first name: ");
String firstName = scanner.nextLine();

System.out.print("Enter last name: ");
String lastName = scanner.nextLine();
```

For:

```
for (double grade : grades) {
    System.out.println(grade);
}
```

While:

```
while (option != 10) {
    System.out.println("====================================");
    System.out.println("Select an option:");
    System.out.println("1. Display all students");
    System.out.println("2. Add a new student");
    System.out.println("3. Delete a student by ID");
    System.out.println("4. Display all available classes");
    System.out.println("5. Display grades for a student in a class");
    System.out.println("6. Add a grade for a student");
    System.out.println("7. Display all grades of a student in all
classes");
    System.out.println("8. Delete a grade for a student");
    System.out.println("9. Display all professors");
    System.out.println("10. Exit");
    System.out.println("====================================");
.....
}
```

Switch:

```
switch (option) {
    case 1:
        List<Student> students = studentDAO.getAllStudents();
        System.out.println("All students in the database:");
        for (Student student : students) {
            System.out.println(student.getId() + ": " +
student.getFirst_name() + " " + student.getLast_name() + ", " +
student.getAge() + " years old, " + student.getEmail());
        }
        break;
    case 2:
        System.out.print("Enter student ID: ");
        int newStudentId = scanner.nextInt();
        scanner.nextLine();

        ……

    case 10:
        System.out.println("Exiting...");
        break;
    default:
        System.out.println("Invalid option");
        break;
}
```

If:

```
if (input == null) {
    throw new RuntimeException("Sorry, unable to find db.properties");
}
```

Laborator 3:

List:

```
List<Student> students = studentDAO.getAllStudents();

private List<Float> grades;

List<String> classes = new ArrayList<>();
```

Laborator 4:

Exemple de clase din proiect:

- **Main**
- **DatabaseUtil**

- **ProfessorDAO**
- **Student**
- **StudentState**
- **StudentDAO**
- **ClassInfo**
- **Person**

Laborator 5:

Clasă abstractă:

Clasa Person este o clasa abstractă.

```
public abstract class Person implements IPerson {
    private int id;
    private String firstName;
    private String lastName;
……….
}
```

Moștenire de clase:

```
public class Professor extends Person implements IProfessor {
```

clasa Profesor moșenește de la clasa Person folosind cuvântul “extends”;

```
public class Student extends Person implements IStudent {
```

clasa Student moșenește de la clasa Person folosind cuvântul “extends”;

```
public class StudentState extends Student implements IStudentState {
```

clasa StudentState moșenește de la clasa Student folosind cuvântul “extends”;

Laborator 6:

Interfețele din proiectul meu:

- IClassInfo
- IPerson
- IProfessor
- IProfessorDAO
- IStudent
- IStudentDAO
- IStudentState

Laborator 7:

Teste implementate in proiectul meu:

ClassInfoTest:

- testGetClassName()
- testSetClassName()
- testGetGrades()
- testAddGrade()
- testCalculateAverageGrade()

PersonTest:

- getIdFromPersonTest()
- checkBugIfPersonIdIsIncorrectTest()
- getFirstNameFromPersonTest()
- checkBugIfPersonFirstNameIsIncorrectTest()
- getLastNameFromPersonTest()
- checkBugIfPersonLastNameIsIncorrectTest()
- setIdForPersonTest()
- setFirstNameForPersonTest()
- setLastNameForPersonTest()

StudentStateTest:

- testGetClasses()
- testAddClass()
- testGetClassInfo()

ProfessorTest:

- getIdFromProfessorTest()
- checkBugIfProfessorIdIsIncorrectTest()
- getFirstNameFromProfessorTest()
- checkBugIfProfessorFirstNameIsIncorrectTest()
- getLastNameFromProfessorTest()
- checkBugIfProfessorLastNameIsIncorrectTest()
- setIdForProfessorTest()
- setFirstNameForProfessorTest()
- setLastNameForProfessorTest()

ProfessorDAOTest:

- testGetAllProfessors()

StudentDAOTest:

- testAddStudent()
- testGetStudent()
- testGetAllStudents()
- testUpdateStudent()
- testDeleteStudent()
- testGetStudentGrades()
- testAddGrade()
- testGetAllGrades()
- testDeleteGrade()

StudentTest:

- getFirstNameFromStudentTest()
- checkBugIfStudentFirstNameIsIncorrectTest()
- getLastNameFromStudentTest()
- checkBugIfStudentLastNameIsIncorrectTest()
- getAgeFromStudentTest()
- checkBugIfStudentAgeIsIncorrectTest()
- getEmailFromStudentTest()
- checkBugIfStudentEmailIsIncorrectTest()

Exemplu de implementare:

```
@Test
public void getLastNameFromStudentTest() {
    Student student = new Student(1, "Mihai", "Dorin", 20, "Masculin",
"Mihai.Dorin@gmail.com", "1234567890", "Lunga 123", "Matematica",
"Informatica", true);
    assertEquals("Dorin", student.getLast_name());
}
```

Laborator 8:

Pentru salvarea datelor într-un fișier de tip .json se ocupă clasa DataExporter.

```
public static void exportDataToJson(String filePath) throws SQLException,
IOException {
    StudentDAO studentDAO = new StudentDAO();
    List<Student> students = studentDAO.getAllStudents();

    ObjectMapper objectMapper = new ObjectMapper();
    objectMapper.enable(SerializationFeature.INDENT_OUTPUT);

    objectMapper.writeValue(new File(filePath), students);
}

public static void main(String[] args) {
    try {
        exportDataToJson("students.json");
        System.out.println("Data exported to students.json");
    } catch (SQLException | IOException e) {
        e.printStackTrace();
    }
}
```

Fișierul students.json salvează datele sub următoarea formă:

```
[ {
  "id" : 1,
  "firstName" : "Ion",
  "lastName" : "Popescu",
  "age" : 20,
  "gender" : "Masculin",
  "email" : "ion.popescu@gmail.ro",
  "phone" : "0712-345-678",
  "address" : "Strada Principală 123",
  "faculty" : "Științe",
  "major" : "Biologie",
  "psycho_pedagogical_module" : true,
  "classes" : [ ],
  "first_name" : "Ion",
  "last_name" : "Popescu"
}, {
  "id" : 2,
  "firstName" : "Maria",
  "lastName" : "Ionescu",
  "age" : 22,
  "gender" : "Feminin",
  "email" : "maria.ionescu@gmail.ro",
  "phone" : "0723-456-789",
  "address" : "Strada Ulmului 456",
  "faculty" : "Arte",
  "major" : "Istorie",
  "psycho_pedagogical_module" : false,
  "classes" : [ ],
  "first_name" : "Maria",
  "last_name" : "Ionescu"
}, {
```

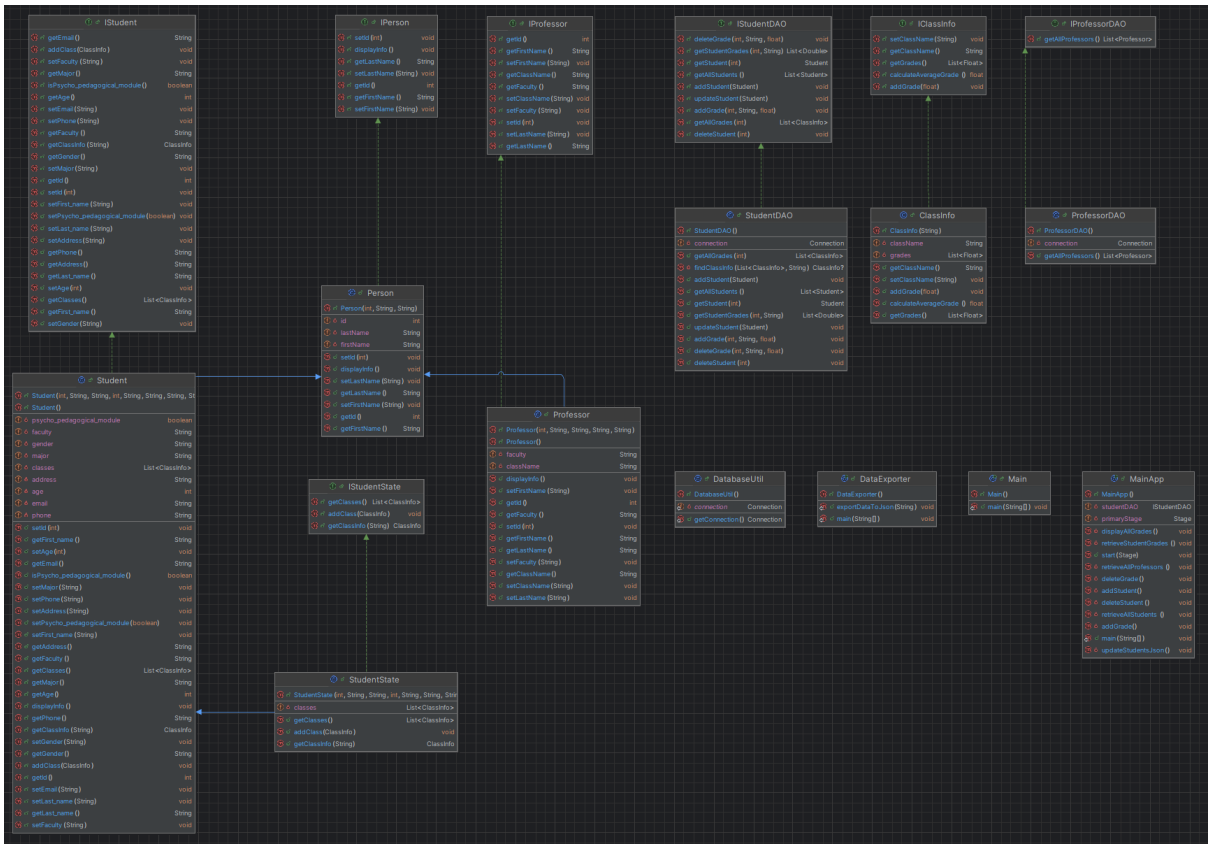
```
  "id" : 3,
  "firstName" : "Ana",
  "lastName" : "Georgescu",
  "age" : 21,
  "gender" : "Feminin",
  "email" : "ana.georgescu@gmail.ro",
  "phone" : "0734-567-890",
  "address" : "Strada Stejarului 789",
  "faculty" : "Inginerie",
  "major" : "Informatică",
  "psycho_pedagogical_module" : true,
  "classes" : [ ],
  "first_name" : "Ana",
  "last_name" : "Georgescu"
}, {
  "id" : 4,
  "firstName" : "Mihai",
  "lastName" : "Dumitrescu",
  "age" : 23,
  "gender" : "Masculin",
  "email" : "mihai.dumitrescu@gmail.ro",
  "phone" : "0745-678-901",
  "address" : "Strada Pinului 101",
  "faculty" : "Afaceri",
  "major" : "Marketing",
  "psycho_pedagogical_module" : false,
  "classes" : [ ],
  "first_name" : "Mihai",
  "last_name" : "Dumitrescu"
}, {
  "id" : 5,
  "firstName" : "George",
  "lastName" : "Marinescu",
  "age" : 20,
  "gender" : "Masculin",
  "email" : "george.marinescu@gmail.ro",
  "phone" : "0756-789-012",
  "address" : "Strada Arțarului 202",
  "faculty" : "Științe",
  "major" : "Chimie",
  "psycho_pedagogical_module" : true,
  "classes" : [ ],
  "first_name" : "George",
  "last_name" : "Marinescu"
}, {
  "id" : 6,
  "firstName" : "Elena",
  "lastName" : "Stanescu",
  "age" : 22,
  "gender" : "Feminin",
  "email" : "elena.stanescu@gmail.ro",
  "phone" : "0767-890-123",
  "address" : "Strada Mesteacănului 303",
  "faculty" : "Arte",
  "major" : "Literatură",
  "psycho_pedagogical_module" : false,
  "classes" : [ ],
  "first_name" : "Elena",
  "last_name" : "Stanescu"
}, {
  "id" : 7,
```

```
  "firstName" : "Ioana",
  "lastName" : "Petrescu",
  "age" : 21,
  "gender" : "Feminin",
  "email" : "ioana.petrescu@gmail.ro",
  "phone" : "0778-901-234",
  "address" : "Strada Cedrului 404",
  "faculty" : "Inginerie",
  "major" : "Inginerie Mecanică",
  "psycho_pedagogical_module" : true,
  "classes" : [ ],
  "first_name" : "Ioana",
  "last_name" : "Petrescu"
}, {
  "id" : 8,
  "firstName" : "Vasile",
  "lastName" : "Radu",
  "age" : 23,
  "gender" : "Masculin",
  "email" : "vasile.radu@gmail.ro",
  "phone" : "0789-012-345",
  "address" : "Strada Molidului 505",
  "faculty" : "Afaceri",
  "major" : "Finanțe",
  "psycho_pedagogical_module" : false,
  "classes" : [ ],
  "first_name" : "Vasile",
  "last_name" : "Radu"
}, {
  "id" : 9,
  "firstName" : "Gabriela",
  "lastName" : "Constantinescu",
  "age" : 20,
  "gender" : "Feminin",
  "email" : "gabriela.constantinescu@gmail.ro",
  "phone" : "0790-123-456",
  "address" : "Strada Salciei 606",
  "faculty" : "Științe",
  "major" : "Fizică",
  "psycho_pedagogical_module" : true,
  "classes" : [ ],
  "first_name" : "Gabriela",
  "last_name" : "Constantinescu"
}, {
  "id" : 10,
  "firstName" : "Florin",
  "lastName" : "Andrei",
  "age" : 22,
  "gender" : "Masculin",
  "email" : "florin.andrei@gmail.ro",
  "phone" : "0701-234-567",
  "address" : "Strada Frasinului 707",
  "faculty" : "Arte",
  "major" : "Filosofie",
  "psycho_pedagogical_module" : false,
  "classes" : [ ],
  "first_name" : "Florin",
  "last_name" : "Andrei"
} ]
```

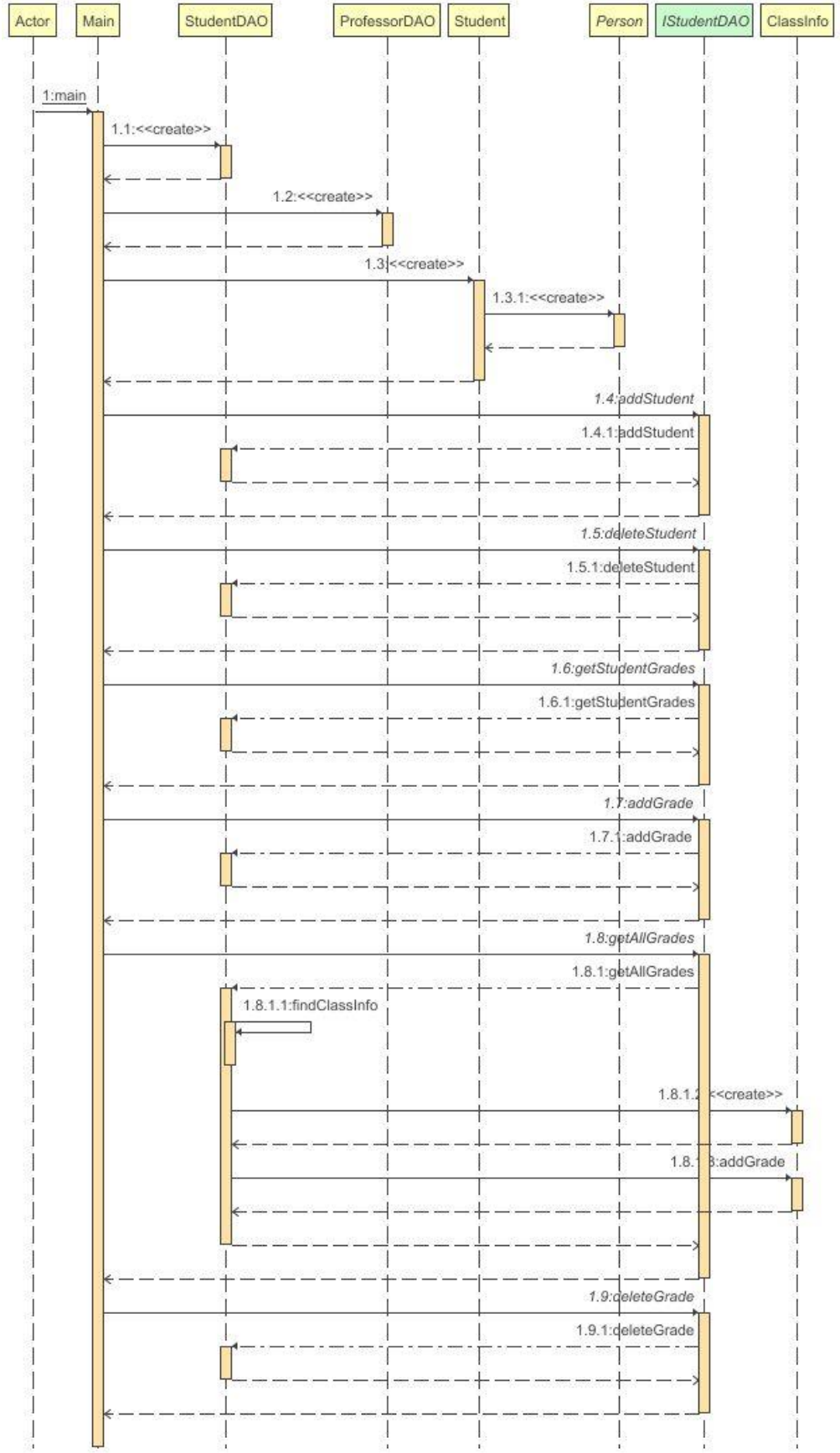
Laborator 9:

Diagrame UML:

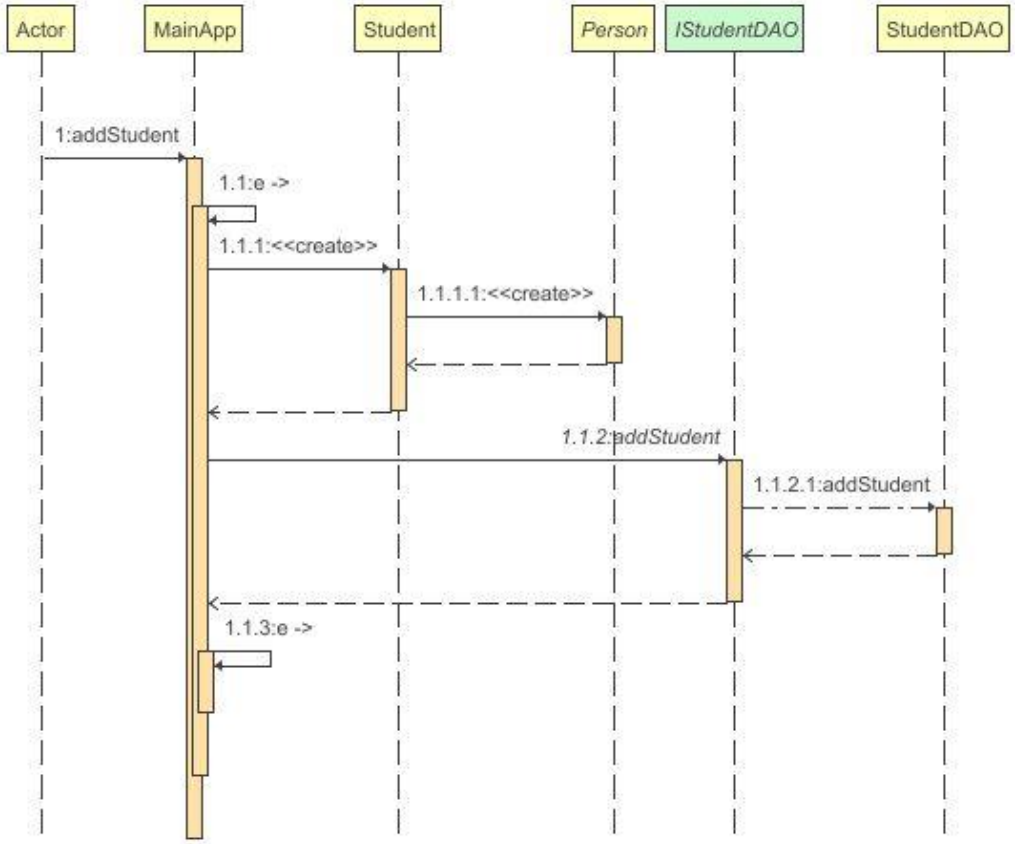
Class Diagram:



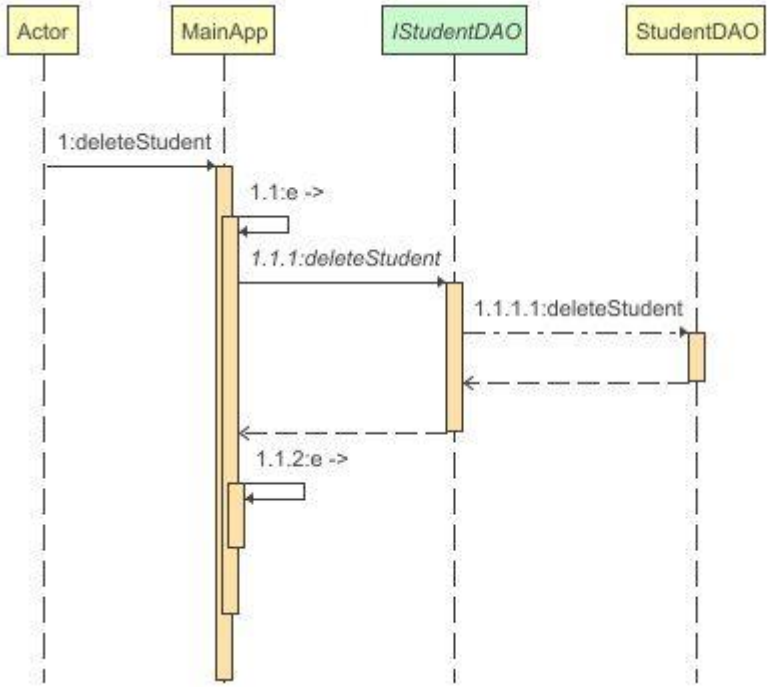
Sequence Diagram pentru Main.java:



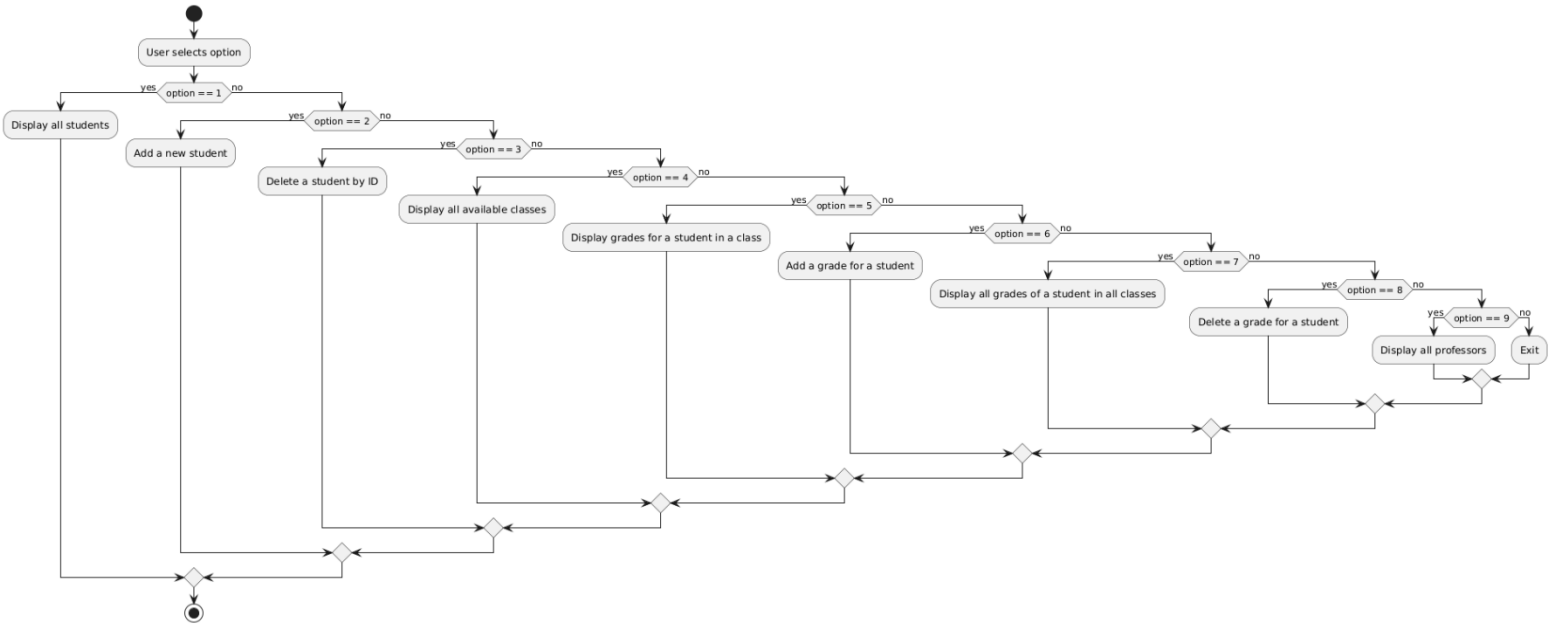
Sequence Diagram pentru AddStudent:



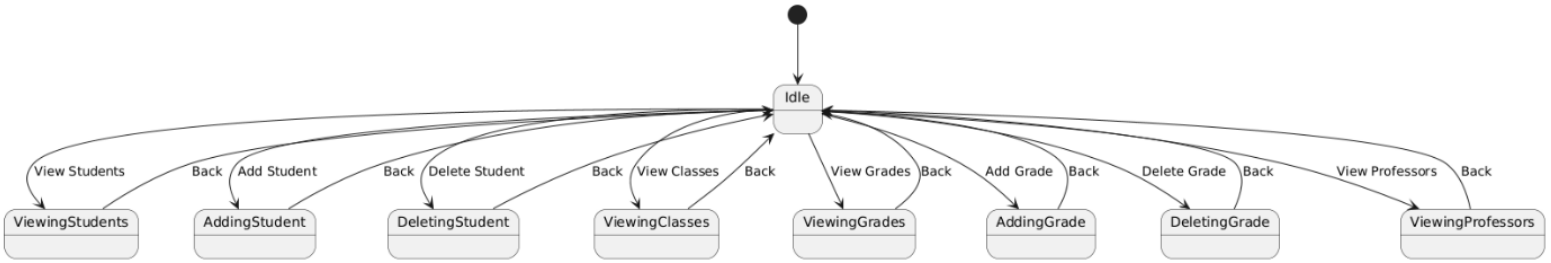
Sequence Diagram pentru DeleteStudent:



Activity Diagram pentru Main.java:



State Diagram pentru Main.java:



UseCase Diagram pentru Main.java: