Java Application Development (Day 5, 6)

• Database integration

1- Data Access Layes class:

this Java class, **DataAccessLayer**, serves as a data access layer for a JavaFX application interacting with an Oracle database. It encapsulates database connectivity and provides methods for performing various operations such as retrieving, updating, and adding data related to students, courses, departments, and grades.

Class Structure

Singleton Pattern

The class is implemented using the Singleton pattern, ensuring that only one instance of **DataAccessLayer** is created.

```
public class DataAccessLayer {
    private static volatile DataAccessLayer instance;
    private static Connection conn;
    private static ResultSet rs;

    private DataAccessLayer() throws SQLException {
        // Initialize the database connection
        DriverManager.registerDriver(new OracleDriver());
        conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE", "SCHOOL", "root");
    }

    public static DataAccessLayer getInstance() throws SQLException {
        // Singleton pattern to ensure only one instance of DataAccessLayer is created
        DataAccessLayer result = instance;
        if (result == null) {
            synchronized (DataAccessLayer.class) {
                result = instance;
                if (result == null) {
                    instance = result = new DataAccessLayer();
                }
            }
        }
        return result;
    }

    public Connection connect() {
        // Get the database connection
        return conn;
    }
```

Constructor

The private constructor initializes the database connection using Oracle JDBC.

Methods

1. **getInstance(): DataAccessLayer**
    - Returns an instance of the **DataAccessLayer** using the Singleton pattern.
2. **connect(): Connection**
    - Provides the database connection.

3. **getStudent(int id): Student**
    - Retrieves student details by ID.
4. **addStudent(int id, String name, String level, String email, int did)**
    - Adds a new student to the database.
5. **updateStudent(int id, String name, String level, String email, int did)**
    - Updates student details in the database.
6. **getStudentCourses(int id): String**
    - Retrieves the courses a student is enrolled in.
7. **getDepartmentCourses(String name): String**
    - Retrieves the courses of a department.
8. **updateGrade(int id, String course, String grade)**
    - Updates the grade of a student for a specific course.
9. **getAllCourses(): List<String>**
    - Retrieves a list of all courses from the database.
10. **enrollStudent(int id, String course)**
    - Enrolls a student in a specific course.
11. **getAllDepartments(): List<String>**
    - Retrieves a list of all departments from the database.
12. **fetchCoursesFromDatabase(String department): ObservableList<Course>**
    - Fetches courses for a specific department.
13. **fetchStudentsFromDatabase(String department): ObservableList<Student>**
    - Fetches students for a specific department.
14. **fetchStudentsByCourse(String courseName): ObservableList<Student>**
    - Fetches students enrolled in a specific course by course name.
15. **fetchStudentsByCourse(int course_id): ObservableList<Student>**
    - Overloaded method, fetches students enrolled in a specific course by course ID.
16. **getStudentGradePercentages(int studentId): Map<String, Double>**
    - Retrieves a map of grade percentages for a specific student.
17. **calculateGpa(int id): String**
    - Calculates the GPA for a specific student.
18. **fetchCoursesByStudent(int id): ObservableList<Course>**
    - Fetches courses with grades for a specific student.

2- Data transferable objects:

1) Course class objects:

The **Course** class in JavaFX represents a course entity with properties such as course ID (**cid**), course name (**courseName**), and course grade (**courseGrade**). This class provides getters and setters for these properties and uses JavaFX **SimpleIntegerProperty** and **SimpleStringProperty** for binding with UI components.

```
public class Course {
    private final SimpleIntegerProperty cid = new SimpleIntegerProperty();
    private final SimpleStringProperty courseGrade = new SimpleStringProperty();
    private final SimpleStringProperty courseName = new SimpleStringProperty();

    // Constructors
    public Course() {
        // Default constructor
    }

    public Course(int cid, String courseName) {
        this.cid.set(cid);
        this.courseName.set(courseName);
    }

    public Course(String courseName, String courseGrade) {
        this.courseName.set(courseName);
        this.courseGrade.set(courseGrade);
    }
```

2) Student Class objects

The Student class in JavaFX represents a student entity with properties such as student ID (id), name (name), academic level (level), email (email), and department ID (did). This class follows the JavaBeans convention and uses JavaFX SimpleIntegerProperty and SimpleStringProperty for property binding with UI components.

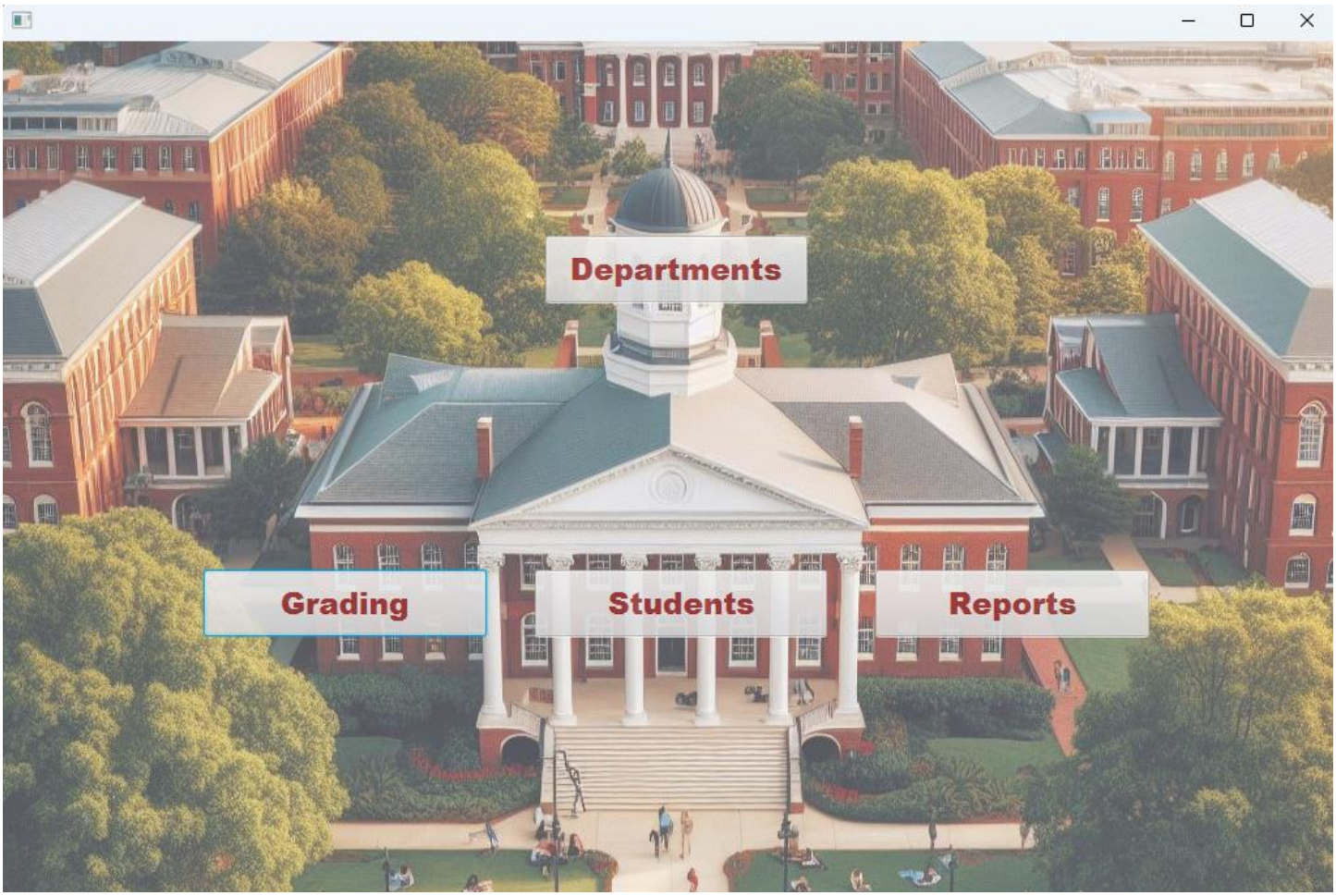
```
// Constructors
public Student() {
    // Default constructor
}

public Student(int id, String name){
    this.id.set(id);
    this.name.set(name);
}

public Student(int id, String name, String level, String email, int did) {
    this.id.set(id);
    this.name.set(name);
    this.level.set(level);
    this.email.set(email);
    this.did.set(did);
}

public Student(String course_name, String grade) {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
}
```

- **Layout:**



Scene 1: HomeScene:

Description: The LoginScene FXML file represents the initial screen of the application, providing a visually appealing and user-friendly login interface. It incorporates background imagery, buttons for navigating to different sections of the application, and a layout that balances aesthetics with functionality.

FXML Components:

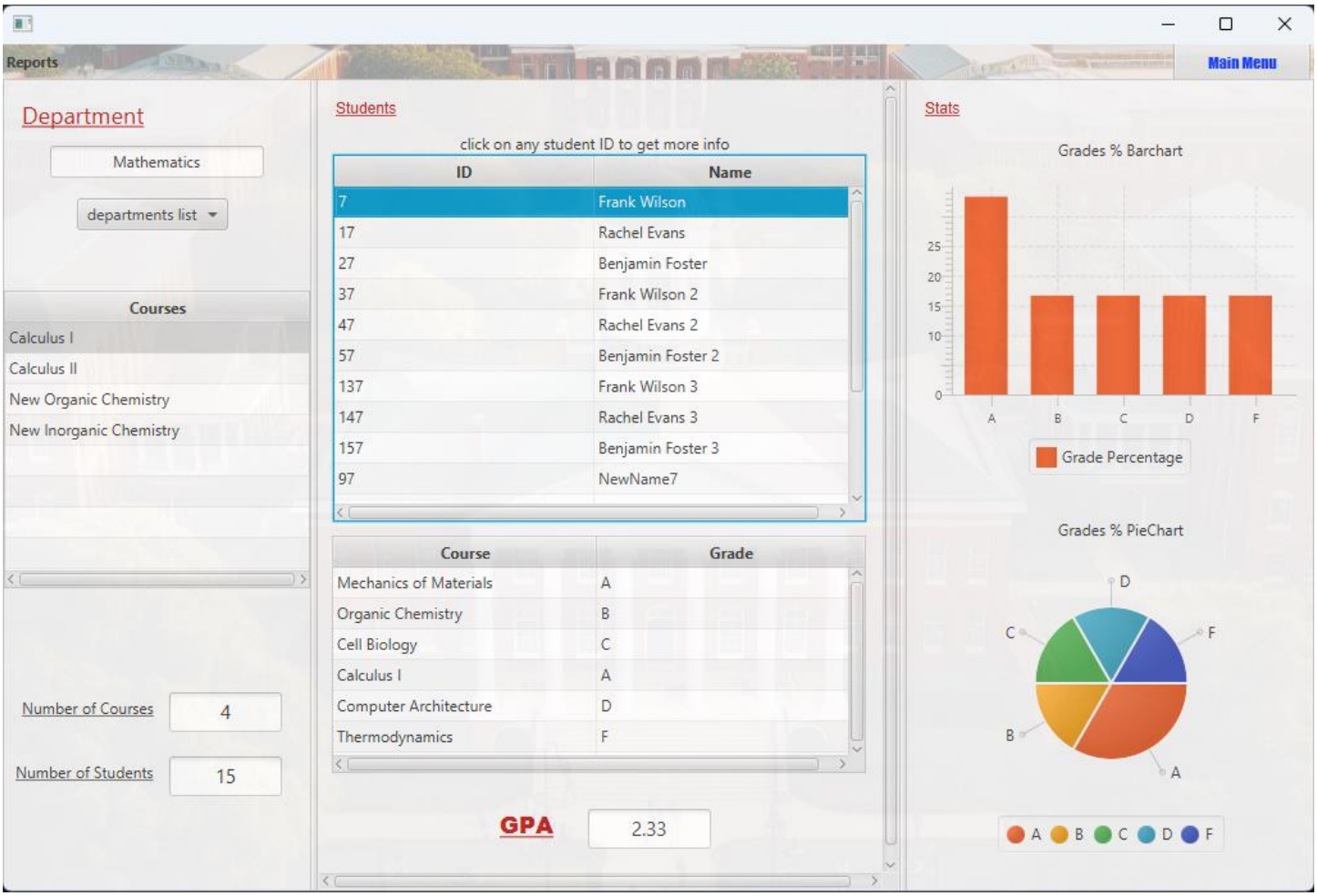
1. StackPane:
   - Root container for the scene, allowing overlaying of elements.
2. Pane:
   - Positioned in the center of the StackPane, contains an ImageView and BorderPane.
3. ImageView:
   - Background image with responsive sizing to enhance the visual appeal.
4. BorderPane:
   - Central layout container holding buttons for different sections.
   - Positioned at the center of the Pane.
5. Button Components:

- gradBtn (Grading):
    - Button for accessing the grading section.
    - Positioned on the left side of the BorderPane.
- reportBtn (Reports):
    - Button for accessing the reports section.
    - Positioned on the right side of the BorderPane.
- stuBtn (Students):
    - Button for accessing the students section.
    - Positioned at the center of the BorderPane.
- DepartmnetsBtn (Departments):
    - Button for accessing the departments section.
    - Positioned at the bottom of the BorderPane.

Usage:

1. Launch the application, and the LoginScene will be the initial view.
2. Background image enhances the visual experience.
3. Click on buttons (Grading, Reports, Students, Departments) to navigate to different sections of the application.

Scene 2: Reporting Scene:



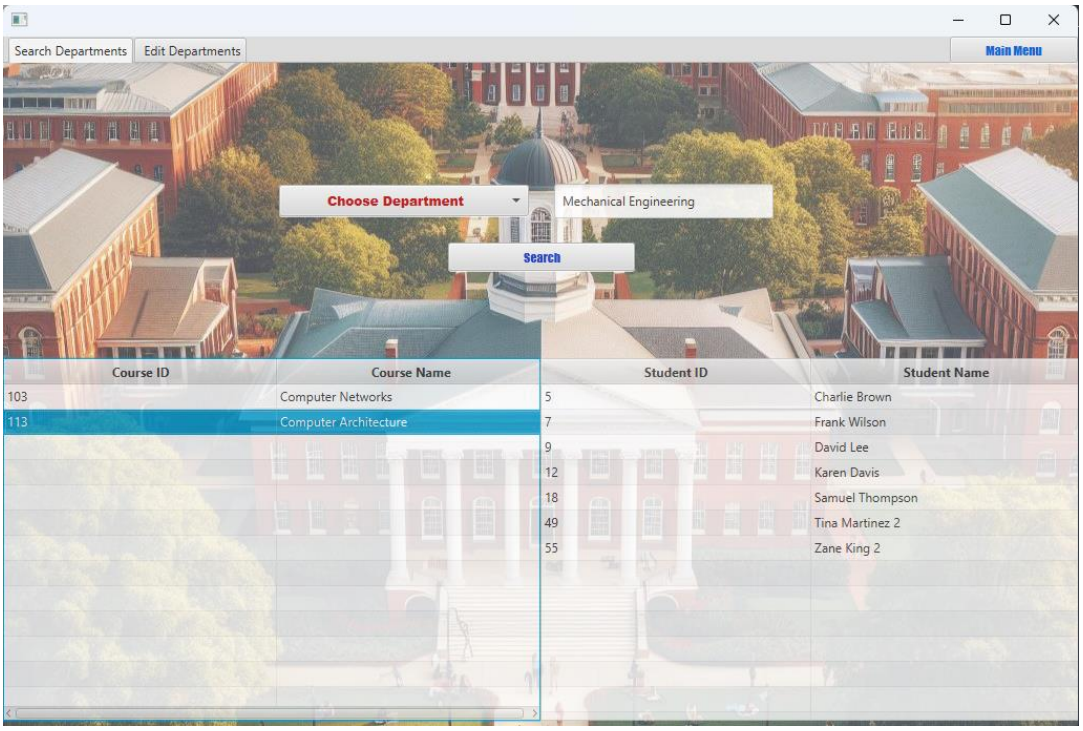
Features

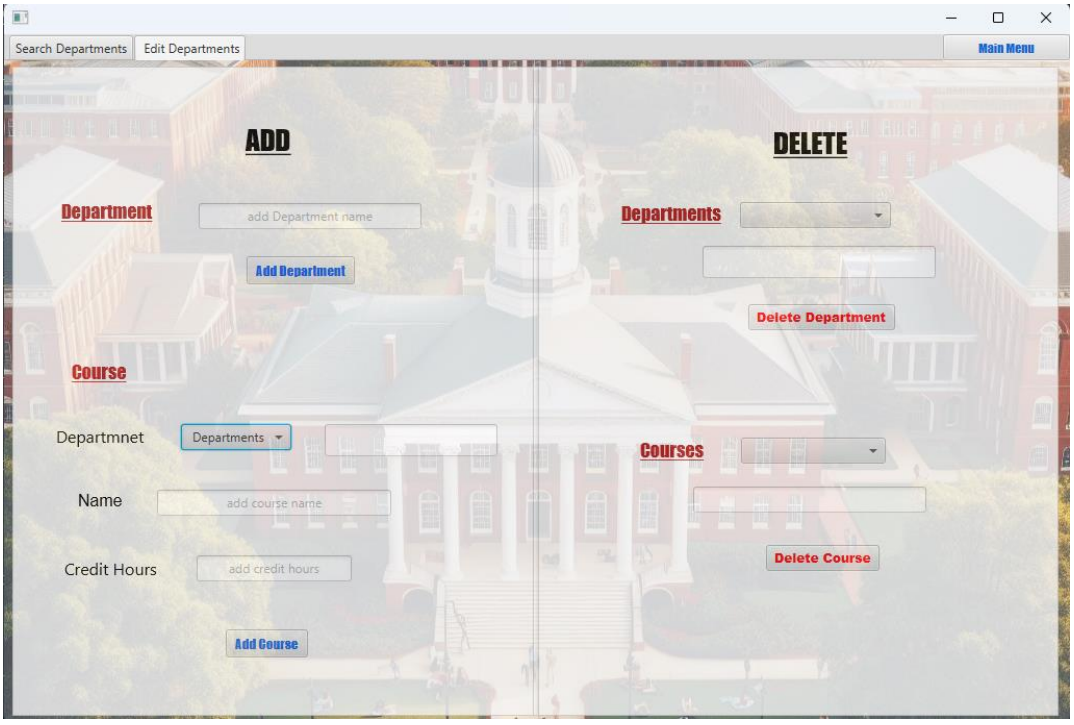
1. Select Department:
    - Users can choose a department from the available options.
    - Upon selection, the application displays relevant information about courses and students in that department.
2. View Department Courses:
    - Courses offered by the selected department are displayed in a TableView.
    - The table includes information about each course, such as the course name.
3. View Department Students:
    - Students enrolled in the selected department are displayed in a TableView.
    - The table includes information about each student, such as student ID and name.
4. Detailed Student Information:
    - Users can click on a student's ID to view more detailed information.
    - Information includes GPA, courses taken, grade distribution, a pie chart, and a bar chart.
5. Interactive Student Grade Distribution Charts:
    - The application dynamically generates a PieChart and a BarChart to represent the grade distribution of the selected student.
    - PieChart: Visual representation of the percentage distribution of grades.
    - BarChart: Displaying the percentage of each grade in a bar chart.

Usage

1. Select Department:
    - Choose a department from the dropdown menu.
2. View Courses and Students:
    - After selecting a department, the application displays information about courses and students in that department.
3. Detailed Student Information:
    - Click on a student's ID in the students' table to view detailed information, including GPA, courses, and grade distribution.
4. Interactive Charts:
    - The application generates interactive charts dynamically based on the selected student's data.

Scene 3: Departments Scene:

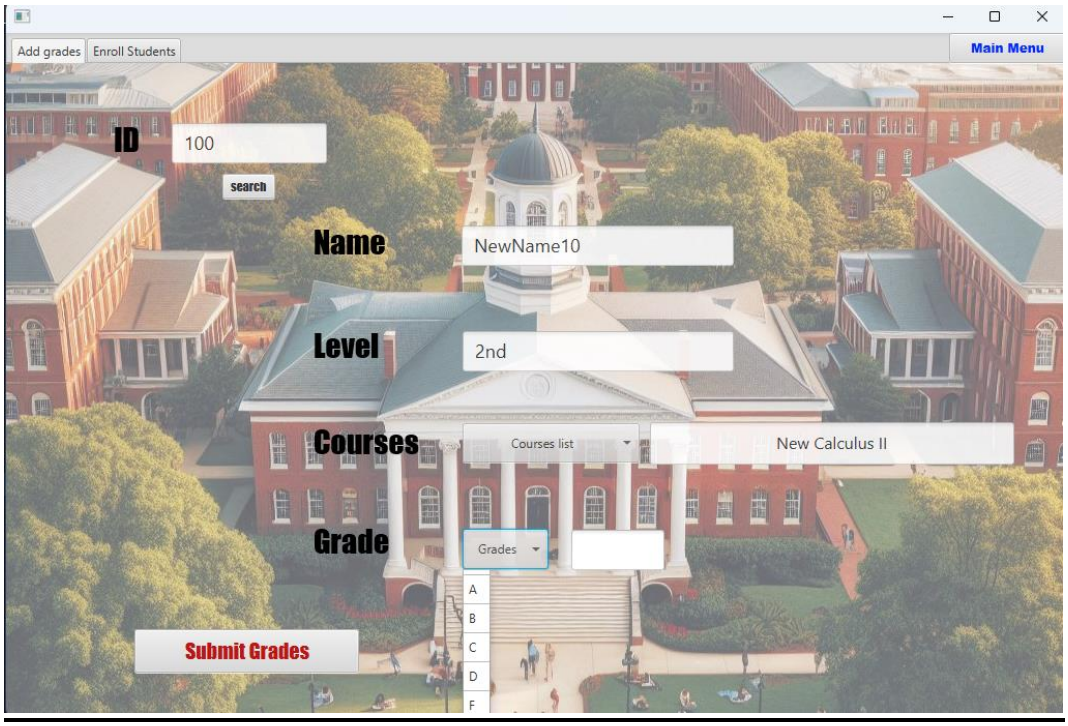


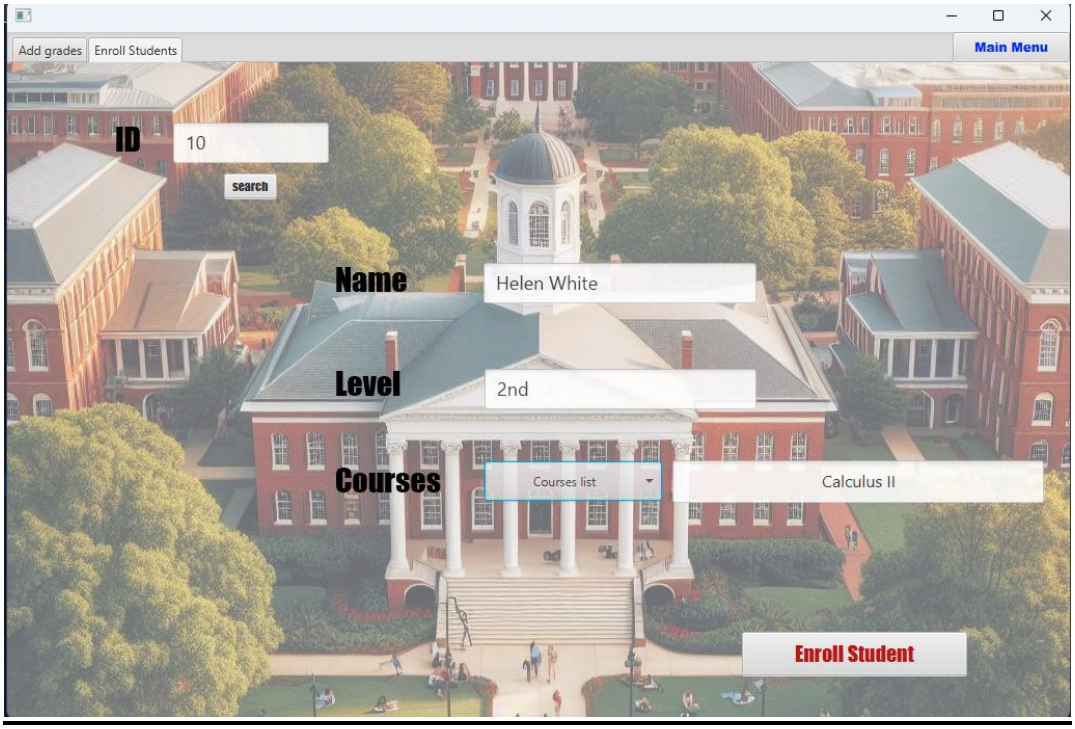


Key Functionalities:

1. Switching to Main Scene:
    - The mainMenuBtn event handler switches to the main scene, but the implementation is currently empty (switchToMainScene method). This can be customized based on your application's requirements.
2. Setting Departments List:
    - Retrieves and sets the list of departments dynamically.
    - Populates different menus (departmentSearchList, deleteDepartList, addCourseDepartmentList) with department names.
    - Provides event handlers for selecting departments, updating associated text fields (searchDepartmentTxt, deleteDepartmentNameTxt, addCourseDepartmentTxt).
3. Setting Courses List:
    - Retrieves and sets the list of courses dynamically.
    - Populates the deleteCourseList menu with course names.
    - Provides an event handler for selecting courses and updating the associated text field (deleteCourseNameTxt).
4. Event Handlers for Tabs:
    - Detects tab selection changes (setOnSelectionChanged).
    - Calls updateDepartmentsList() when "Search Departments" or "Edit Departments" tab is selected.
5. Search Button (searchSearchBtn):
    - Performs a search based on the selected department.
    - Retrieves and displays associated courses and students from the database.
    - Updates Courses_Table and Student_Table in the UI.
6. Add Department Button (addDepartmentBtn):
    - Adds a new department to the database.
    - Performs validation to avoid duplicates and empty entries.
    - Displays success or warning alerts accordingly.
7. Delete Department and Course Buttons (deleteDepartmentBtn, deleteCourseBtn):
    - Deletes the selected department or course from the database.
    - Confirms the action with a user prompt.
    - Updates the departments or courses list and displays success or warning alerts accordingly.
8. Add Course Button (addCourseBtn):
    - Adds a new course to the database.
    - Validates input fields and checks for duplicates.
    - Displays success or warning alerts accordingly.

Scene 4: Grading Scene:





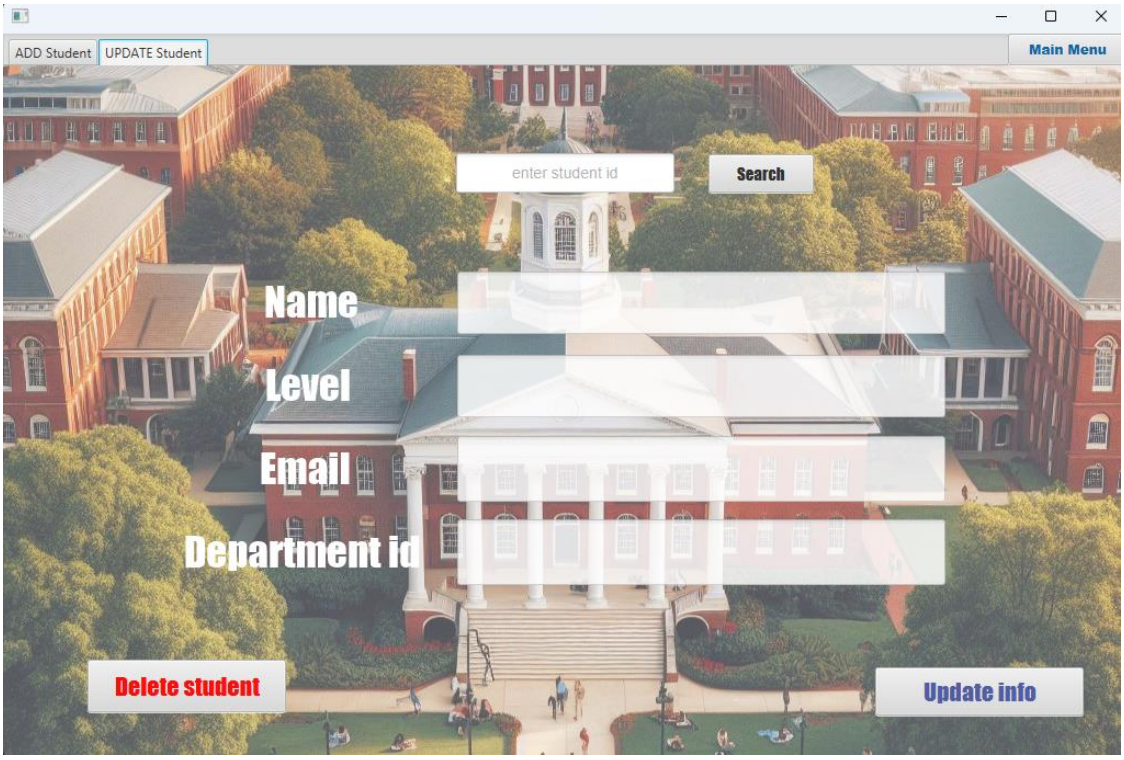
UI Components:

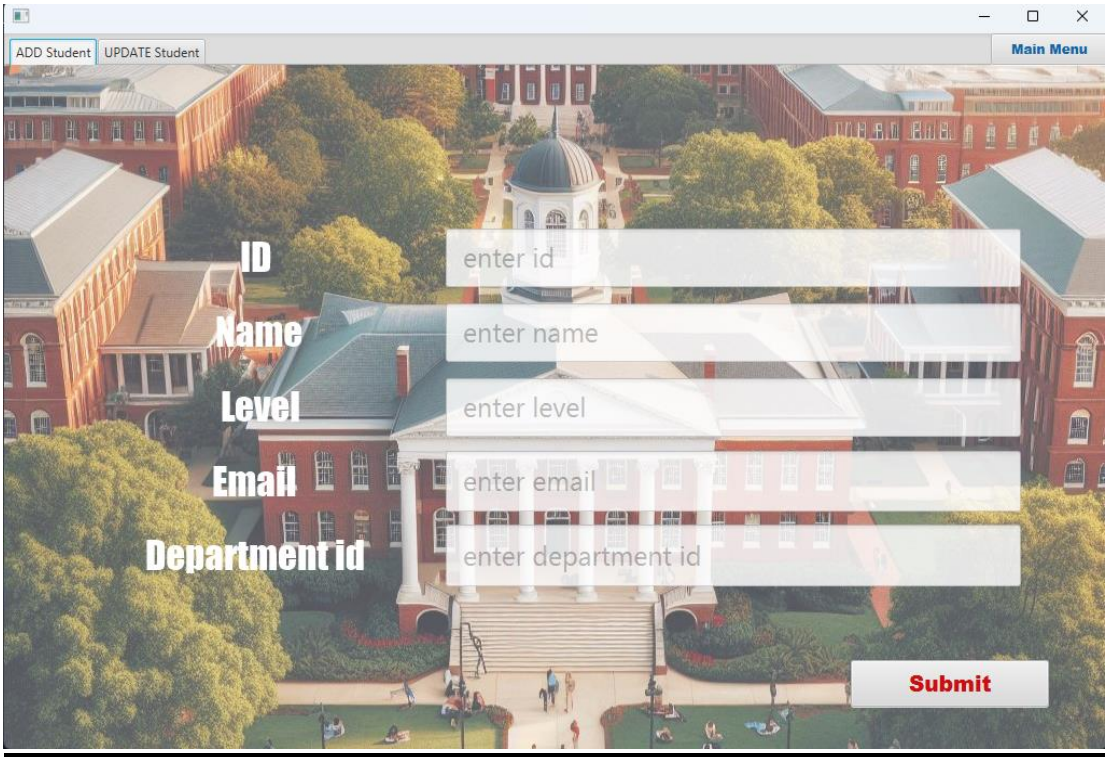
1. TabPane (tabPane):
    - Contains two tabs: "Add grades" (tab) and "Enroll Students" (tab0).
2. "Add grades" Tab (tab):
    - Includes input fields for searching, displaying, and updating student information.
    - Displays student details such as ID, Name, Level, Courses, and Grade.
    - Provides buttons for searching, updating grades, and navigating to the main menu.
3. "Enroll Students" Tab (tab0):
    - Similar structure to "Add grades" but focused on enrolling students in courses.
    - Allows searching, displaying, and enrolling students in available courses.
4. Buttons:
    - "Submit Grades" (addSubmitBtn): Initiates the process of updating a student's grade.
    - "Enroll Student" (enrollBtn): Initiates the process of enrolling a student in a course.
    - "Main Menu" (button): Navigates to the main menu.

Event Handlers:

1. switchToMainScene:
    - Linked to the "Main Menu" button, intended to switch to the main scene.
2. addSearchBtn:
    - Handles the search button in the "Add grades" tab.
    - Retrieves and displays student information based on the provided ID.
3. addSubmitBtn:
    - Handles the submit button in the "Add grades" tab.
    - Updates the student's grade for a specific course.
4. enrollSearchBtn:
    - Handles the search button in the "Enroll Students" tab.
    - Retrieves and displays student information based on the provided ID.
5. enrollBtn:
    - Handles the enroll button in the "Enroll Students" tab.
    - Enrolls a student in a specific course.

Scene 5: Students Scene:





Features:

1. Add Student:
    - Navigate to the "ADD Student" tab.
    - Enter student details such as ID, Name, Level, Email, and Department ID.
    - Click the "Submit" button to add the student to the database.
    - Input validation ensures that essential fields are filled, and valid integers are provided for ID and Department ID.
2. Update Student:
    - Navigate to the "UPDATE Student" tab.
    - Enter the student ID you want to update in the search field and click "Search."
    - If the student exists, their details are displayed.
    - Modify the student's information (Name, Level, Email, Department ID).
    - Click the "Update info" button to save the changes to the database.
3. Delete Student:
    - On the "UPDATE Student" tab, enter the student ID you want to delete in the search field and click "Search."
    - Confirm the deletion by clicking "Delete student" after the student details are displayed.
    - The student is permanently removed from the database.
4. Switch to Main Menu:
    - Click the "Main Menu" button to switch to the main menu or another scene.