**Explanation: Student Management System (Swing GUI)**

This Java program creates a **Student Management System** using **Swing**, which is a toolkit for making desktop apps with graphical interfaces (buttons, fields, etc.).

**1. Student Class – Storing Student Info**

class Student {

String name;

String id;

ArrayList<String> courses;

ArrayList<Integer> grades;

}

* Think of this class as a **student profile**.
* It keeps track of:
  + Name
  + Student ID
  + List of courses
  + List of grades
* ArrayList lets us add many courses and grades easily.

**2. Main Program – The GUI Layout**

public class StudentManagementSystem extends JFrame {

// GUI components and student list declared here

}

* This is the **main window** of the app.
* It includes:
  + Text fields to enter student info.
  + A table to show student records.
  + Dropdowns (ComboBoxes) for picking students and courses.
  + Buttons to add, update, enroll, and assign grades.

**3. GUI Layout Setup**

public StudentManagementSystem() {

setLayout(new BorderLayout());

add(createInputPanel(), BorderLayout.NORTH);

add(new JScrollPane(studentTable), BorderLayout.CENTER);

add(createEnrollmentPanel(), BorderLayout.SOUTH);

}

* The screen is split into three parts:
  + **Top (North)**: Input area for adding/updating students.
  + **Middle (Center)**: Table that shows all students.
  + **Bottom (South)**: Panel to enroll students and assign grades.

**4. What the Buttons Do (Event Handling)**

**Add Student**

new Student(name, id);

students.add(student);

studentTableModel.addRow(...);

* When you click **"Add Student"**, it:
  1. Creates a new student.
  2. Adds them to the list.
  3. Shows them in the table and dropdown menu.

**Update Student**

students.get(selectedRow).name = newName;

* When you click **"Update Student"**, it changes the name in both:
  + The student list
  + The table and dropdown

**Enroll in Course**

student.courses.add(selectedCourse);

* When you click **"Enroll"**, it adds the chosen course to that student's record.

**Assign Grade**

student.grades.add(grade);

* When you click **"Assign Grade"**, it adds a grade to the selected student.

**5. Error Messages**

showError("Error message");

* If something goes wrong (e.g., missing name or invalid grade), a popup shows the error using JOptionPane.

**6. Limitations (Things That Could Be Better)**

1. **Courses and grades aren't shown** in the GUI, only stored internally.
2. It uses **names** to find students (not IDs), which can be confusing if names repeat.
3. It may become **slow** if the list of students gets very long (because of simple for loops).

**7. How to Use It**

1. Save the file as StudentManagementSystem.java.
2. Compile it using:

javac StudentManagementSystem.java

1. Run it using:

java StudentManagementSystem

1. A window will appear where you can manage students using buttons and fields.

**References (Code & Tools Used)**

* Java Swing Documentation: <https://docs.oracle.com/javase/tutorial/uiswing/>
* JFrame, JTextField, JComboBox, JTable, and JButton from **javax.swing**
* Java Collections: [ArrayList](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html)
* Event Handling in Java: <https://docs.oracle.com/javase/tutorial/uiswing/events/>