

**STUDENT MANAGEMENT APPLICATION**

This project is a GUI for a student management system that enables users to manage student, courses and grades in an institution. The system allows for the creation of different student profiles, viewing and managing their details. The GUI consists of two sections – Student tab and Course tab – within the Administrator Interface. The system has three main backend classes – StudentManagement class, Student class and Course class.

**Administrator Interface**

The AdministratorInterface class is a graphical user interface (GUI) designed specifically for administrators in the student management system. It provides an intuitive and user-friendly interface for administrators to manage various aspects of the system, such as creating and managing student profiles, enrolling students in courses, assigning instructors to courses, and generating reports.

**Display:**

The AdministratorInterface features two primary tabs: the Students tab and the Courses tab. Both tabs utilize a table layout for an organized and intuitive presentation of information. Within each tab, every row showcases a distinct student or course, while the columns display relevant details such as ID, Student Name, Overall Grade, Course Code, and Course Name. This tabular format enables administrators to efficiently review and manage essential information pertaining to students and courses.

Administrator Interface

Students

Courses

ID	Student Name	Overall Grade	
----	--------------	---------------	--

Add New Student

View Details

Remove Student

Administrator Interface

Students

Courses

Code	Course Name	Students Count	Max Capacity	
------	-------------	----------------	--------------	--

Add New Course

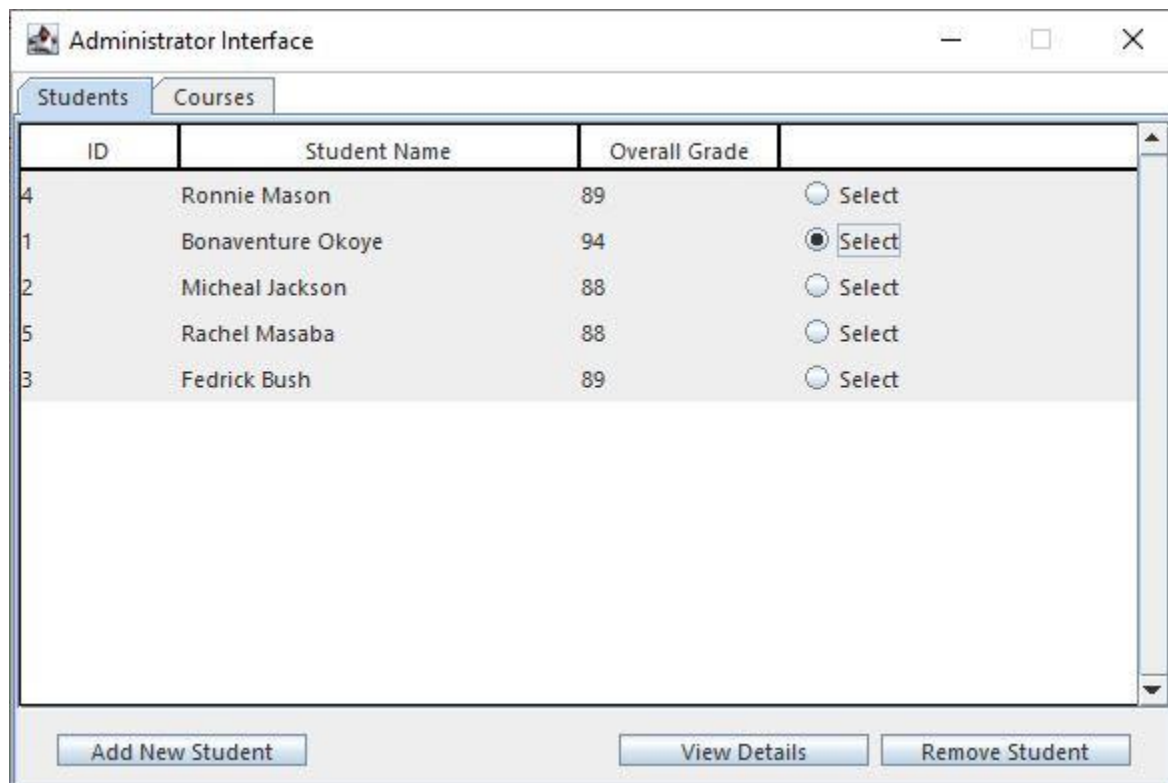
View Details

Delete Course

In addition to the two primary tabs, the AdministratorInterface also incorporates a footer panel, which houses a collection of buttons dedicated to managing students and courses. These buttons provide administrators with quick access to various management functions, such as adding, modifying, or removing students and courses. The inclusion of this footer panel enhances the overall user experience by offering a convenient and centralized location for executing essential administrative tasks.

### Students Tab:

The Students tab in the AdministratorInterface gives administrators the ability to efficiently navigate through an organized list of registered students. With a simple click, administrators can select a student to view or modify their details, ensuring that the information remains accurate and up-to-date. Furthermore, this tab enables the removal of a student from the system when necessary and provides an option to add new students, streamlining the management of student records.



## Code Showcase:

```
package StudentManagement;

import java.util.*;
import java.awt.*;
import java.awt.event.*;

import javax.swing.*;
import javax.swing.GroupLayout.Alignment;
import javax.swing.LayoutStyle.ComponentPlacement;
import javax.swing.border.LineBorder;

public class AdministratorInterface {

    // Singleton instance
    private static AdministratorInterface instance;

    // GUI components
    private JFrame frmAdministratorInterface;
    private static HashMap<JRadioButton, Student> studentMap = new HashMap<>();
    private final ButtonGroup studentButtonGroup = new ButtonGroup();

    // Entry point of the program
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    // Initialize and display the GUI
                    AdministratorInterface window = new
AdministratorInterface(100, 100);
                    window.frmAdministratorInterface.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    // Constructor
    public AdministratorInterface(int x, int y) {
        initialize(x, y);
    }

    // Singleton instance retrieval method
    public static AdministratorInterface getInstance() {
        return instance;
    }
}
```

```

// Method to refresh the interface
public void refresh() {
    int x = frmAdministratorInterface.getX();
    int y = frmAdministratorInterface.getY();

    try {
        // Create a new instance of the interface with the same position
        AdministratorInterface window = new AdministratorInterface(x, y);
        window.frmAdministratorInterface.setVisible(true);
    } catch (Exception e) {
        e.printStackTrace();
    }
    // Dispose the current interface
    this.frmAdministratorInterface.dispose();
}

// Initialize the GUI components
private void initialize(int x, int y) {
    instance = this;

    // Set up JFrame properties
    frmAdministratorInterface = new JFrame();
    frmAdministratorInterface.setResizable(false);
    frmAdministratorInterface.getContentPane().setBackground(new Color(255,
255, 255));
    frmAdministratorInterface.setTitle("Administrator Interface");
    frmAdministratorInterface.setBounds(x, y, 600, 400);

    frmAdministratorInterface.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    // Create tabbed pane for managing students and courses
    JTabbedPane Tab = new JTabbedPane(JTabbedPane.TOP);
    Tab.setBorder(new LineBorder(new Color(192, 192, 192)));
    Tab.setFont(new Font("Segoe UI", Font.PLAIN, 11));

    // Set up GroupLayout for main content pane
    GroupLayout groupLayout = new
GroupLayout(frmAdministratorInterface.getContentPane());
    groupLayout.setHorizontalGroup(
        groupLayout.createParallelGroup(Alignment.TRAILING)
            .addGroup(groupLayout.createSequentialGroup()
                .addComponent(Tab, GroupLayout.DEFAULT_SIZE, 434,
Short.MAX_VALUE)
            );
    groupLayout.setVerticalGroup(
        groupLayout.createParallelGroup(Alignment.LEADING)
            .addComponent(Tab, Alignment.TRAILING,
GroupLayout.DEFAULT_SIZE, 261, Short.MAX_VALUE)
    );
}

```

```

        // Add tabs and panels for managing students and courses
        setupStudentsPanel(Tab);
        setupCoursesPanel(Tab);

        frmAdministratorInterface.getContentPane().setLayout(groupLayout);
    }

    // Set up the panel for managing students
    private void setupStudentsPanel(JTabbedPane Tab) {

        // Set up students panel
        JPanel studentsPanel = new JPanel();
        studentsPanel.setBackground(Color.WHITE);

        // Set up main students panel and labels
        JPanel stMainPanel = new JPanel();
        stMainPanel.setBackground(Color.WHITE);

        JScrollPane stScrollPane = new
        JScrollPane(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS,
        ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
        stScrollPane.setViewportView(stMainPanel);

        JPanel headerID = new JPanel();
        headerID.setBorder(new LineBorder(new Color(0, 0, 0)));
        headerID.setBackground(Color.WHITE);
        JLabel lblID = new JLabel("ID");
        lblID.setFont(new Font("Segoe UI", Font.PLAIN, 11));
        headerID.add(lblID);

        JPanel headerStudentName = new JPanel();
        headerStudentName.setBorder(new LineBorder(new Color(0, 0, 0)));
        headerStudentName.setBackground(Color.WHITE);
        JLabel lblStudentname = new JLabel("Student Name");
        lblStudentname.setFont(new Font("Segoe UI", Font.PLAIN, 11));
        headerStudentName.add(lblStudentname);

        JPanel headerOverallGrade = new JPanel();
        headerOverallGrade.setBorder(new LineBorder(new Color(0, 0, 0)));
        headerOverallGrade.setBackground(Color.WHITE);
        JLabel lblOverallGrade = new JLabel("Overall Grade");
        lblOverallGrade.setFont(new Font("Segoe UI", Font.PLAIN, 11));
        headerOverallGrade.add(lblOverallGrade);

        JPanel stHeaderExtra = new JPanel();
        stHeaderExtra.setBorder(new LineBorder(new Color(0, 0, 0)));
        stHeaderExtra.setBackground(Color.WHITE);

        JPanel stViewportPanel = new JPanel();
        stViewportPanel.setBorder(new LineBorder(new Color(0, 0, 0)));
        stViewportPanel.setBackground(Color.WHITE);
    }

```

```

// Set up GroupLayout for main panel **OMMITTED**
GroupLayout gl_stMainPanel = new GroupLayout(stMainPanel);
stMainPanel.setLayout(gl_stMainPanel);

// Add footer panel and buttons for managing students
JPanel stFooterPanel = new JPanel();

JButton btnAddNewStudent = new JButton("Add New Student");
btnAddNewStudent.setFont(new Font("Segoe UI", Font.PLAIN, 11));

JButton btnViewStudentDetails = new JButton("View Details");
btnViewStudentDetails.setFont(new Font("Segoe UI", Font.PLAIN, 11));

JButton btnRemoveStudent = new JButton("Remove Student");
btnRemoveStudent.setFont(new Font("Segoe UI", Font.PLAIN, 11));

// Set up GroupLayout for footer panel **OMMITTED**
GroupLayout gl_stFooterPanel = new GroupLayout(stFooterPanel);
stFooterPanel.setLayout(gl_stFooterPanel);

// Set up GroupLayout for students panel **OMMITTED**
GroupLayout gl_studentsPanel = new GroupLayout(studentsPanel);
studentsPanel.setLayout(gl_studentsPanel);

// Set up 'AddNewStudent' button
btnAddNewStudent.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int x = frmAdministratorInterface.getX() + 100;
        int y = frmAdministratorInterface.getY() + 100;
        AddNewStudent newStudentFrame = new AddNewStudent(x, y);
        newStudentFrame.setVisible(true);
    }
});

// Set up 'ViewStudentDetails' button
btnViewStudentDetails.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        JRadioButton selectedButton =
getSelectedButtonText(studentButtonGroup);
        Student student = studentMap.get(selectedButton);
        if (student == null) {
            showErrorPopup("No Student Selected");
        } else {
            int x = frmAdministratorInterface.getX() + 100;
            int y = frmAdministratorInterface.getY() + 100;
            StudentDetails studentDetailsFrame = new
StudentDetails(student, x, y);
            studentDetailsFrame.setVisible(true);
        }
    }
});

```

```

        // Set up 'RemoveStudent' button
        btnRemoveStudent.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                JRadioButton selectedButton =
getSelectedButtonText(studentButtonGroup);
                Student student = studentMap.get(selectedButton);
                if (student == null) {
                    showErrorPopup("No Student Selected");
                } else {
                    int response = JOptionPane.showConfirmDialog(null, "Are
you sure you want to remove "
                        + student.getName()
                        + "?\nThis process cannot be reversed.",
"Confirm Removal", JOptionPane.YES_NO_OPTION);
                    if (response == JOptionPane.YES_OPTION) {
                        StudentManagement.getStudentList().remove(student);
                        student = null;
                        refresh();
                    }
                }
            }
        });

        // Retrieve student objects from StudentManagement class
        displayStudents(stViewportPanel, studentButtonGroup);

        // Add studentsPanel to the tabbed pane
        Tab.addTab("Students", null, studentsPanel, null);
    }

```

```

// Set up the panel for managing courses
private void setupCoursesPanel(JTabbedPane Tab) {

    // Set up courses panel
    JPanel coursesPanel = new JPanel();
    coursesPanel.setBackground(new Color(255, 255, 255));
    Tab.addTab("Courses", null, coursesPanel, null);

    // Set up main courses panel and labels
    JPanel csMainPanel = new JPanel();
    csMainPanel.setBackground(Color.WHITE);

    JScrollPane csScrollPane = new
JScrollPane(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS,
ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
    csScrollPane.setViewportView(csMainPanel);

    JPanel headerCode = new JPanel();
    headerCode.setBorder(new LineBorder(new Color(0, 0, 0)));
    headerCode.setBackground(Color.WHITE);
    JLabel lblCode = new JLabel("Code");
    lblCode.setFont(new Font("Segoe UI", Font.PLAIN, 11));
    headerCode.add(lblCode);

```



```

JPanel headerCourseName = new JPanel();
headerCourseName.setBorder(new LineBorder(new Color(0, 0, 0)));
headerCourseName.setBackground(Color.WHITE);
JLabel lblCourseName = new JLabel("Course Name");
lblCourseName.setFont(new Font("Segoe UI", Font.PLAIN, 11));
headerCourseName.add(lblCourseName);

JPanel headerStudentsCount = new JPanel();
headerStudentsCount.setBorder(new LineBorder(new Color(0, 0, 0)));
headerStudentsCount.setBackground(Color.WHITE);
JLabel lblNumberOfStudents = new JLabel("Students Count");
lblNumberOfStudents.setFont(new Font("Segoe UI", Font.PLAIN, 11));
headerStudentsCount.add(lblNumberOfStudents);

JPanel headerMaxCapacity = new JPanel();
headerMaxCapacity.setBorder(new LineBorder(new Color(0, 0, 0)));
headerMaxCapacity.setBackground(Color.WHITE);
JLabel lblMax = new JLabel("Max Capacity");
lblMax.setFont(new Font("Segoe UI", Font.PLAIN, 11));
headerMaxCapacity.add(lblMax);

JPanel csHeaderExtra = new JPanel();
csHeaderExtra.setBorder(new LineBorder(new Color(0, 0, 0)));
csHeaderExtra.setBackground(Color.WHITE);

JPanel csViewportPanel = new JPanel();
csViewportPanel.setBorder(new LineBorder(new Color(0, 0, 0)));
csViewportPanel.setBackground(Color.WHITE);

// Set up GroupLayout for main panel **OMITTED**
GroupLayout gl_csMainPanel = new GroupLayout(csMainPanel);
csMainPanel.setLayout(gl_csMainPanel);

// Add footer panel and buttons for managing courses
JPanel csFooterPanel = new JPanel();

JButton btnAddNewCourse = new JButton("Add New Course");
btnAddNewCourse.setFont(new Font("Segoe UI", Font.PLAIN, 11));

JButton btnViewCourseDetails = new JButton("View Details");
btnViewCourseDetails.setFont(new Font("Segoe UI", Font.PLAIN, 11));

JButton btnDeleteCourse = new JButton("Delete Course");
btnDeleteCourse.setFont(new Font("Segoe UI", Font.PLAIN, 11));

// Set up GroupLayout for footer panel **OMITTED**
GroupLayout gl_csFooterPanel = new GroupLayout(csFooterPanel);
csFooterPanel.setLayout(gl_csFooterPanel);

// Set up GroupLayout for courses panel **OMITTED**
GroupLayout gl_coursesPanel = new GroupLayout(coursesPanel);
coursesPanel.setLayout(gl_coursesPanel);

```

```

}

```

```

        // Method for retrieving student objects from StudentManagement class
        public static void displayStudents(JPanel stViewportPanel, ButtonGroup
buttonGroup) {
            // Create GroupLayout for managing the layout of the stViewportPanel
            GroupLayout gl_stViewportPanel = new GroupLayout(stViewportPanel);
            GroupLayout.ParallelGroup pg_stViewportPanel =
gl_stViewportPanel.createParallelGroup(Alignment.LEADING);
            GroupLayout.SequentialGroup sg_stViewportPanel =
gl_stViewportPanel.createSequentialGroup();

            // Loop through the list of students obtained from StudentManagement
class
            for (Student student : StudentManagement.getStudentList().keySet()) {

                // Create a panel to contain student information
                JPanel studentPanel = new JPanel();

                // Horizontal layout constraints for studentPanel
                gl_stViewportPanel.setHorizontalGroup(
                    pg_stViewportPanel.addComponent(studentPanel,
GroupLayout.DEFAULT_SIZE, 558, Short.MAX_VALUE)
                );
                // Vertical layout constraints for studentPanel
                gl_stViewportPanel.setVerticalGroup(
                    sg_stViewportPanel
                        .addComponent(studentPanel,
GroupLayout.PREFERRED_SIZE, 23, GroupLayout.PREFERRED_SIZE)
                );

                // Add labels for displaying student information
                JLabel lblStudentID = new
JLabel(String.valueOf(student.getID()));
                lblStudentID.setFont(new Font("Segoe UI", Font.PLAIN, 11));

                JLabel lblStudentName = new JLabel(student.getName());
                lblStudentName.setFont(new Font("Segoe UI", Font.PLAIN, 11));

                JLabel lblStudentOverallGrade = new
JLabel(String.valueOf(StudentManagement.getStudentOverallGrade(student)));
                lblStudentOverallGrade.setFont(new Font("Segoe UI", Font.PLAIN,
11));

                // Radio button for selecting the student
                JRadioButton rdbtnSelect = new JRadioButton("Select");
                buttonGroup.add(rdbtnSelect);
                rdbtnSelect.setFont(new Font("Segoe UI", Font.PLAIN, 11));

                // Map the radio button to the corresponding student
                studentMap.put(rdbtnSelect, student);

                // Set up GroupLayout for studentPanel **OMITTED**
                GroupLayout gl_studentPanel = new GroupLayout(studentPanel);
                studentPanel.setLayout(gl_studentPanel);
            }
}

```

```

stViewportPanel.setLayout(gl_stViewportPanel);
}

// Method to get selected radio button
public JRadioButton getSelectedButtonText(ButtonGroup buttonGroup) {
    for (Enumeration<AbstractButton> buttons = buttonGroup.getElements();
buttons.hasMoreElements();) {
        JRadioButton button = (JRadioButton) buttons.nextElement();

        if (button.isSelected()) {
            return button;
        }
    }
    return null;
}

// Method to display error popup
public void showErrorPopup(String message) {
    JLabel label = new JLabel(message);
    label.setFont(new Font("Segoe UI", Font.PLAIN, 11));

    JOptionPane.showMessageDialog(frmAdministratorInterface, label, "Error",
JOptionPane.ERROR_MESSAGE);
}
}

```

## AddNewStudent Frame

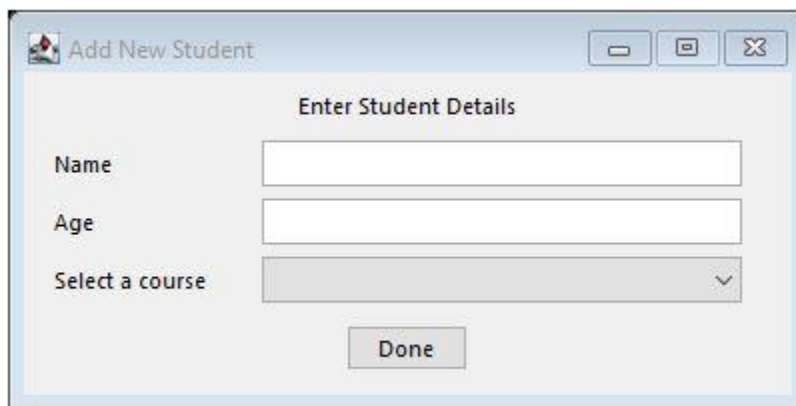
The AddNewStudentFrame is an integral graphical user interface (GUI) component within the AdministratorInterface that streamlines the process of adding a new student to the student management system. This frame is instantiated when the user activates the AddNewStudent button.

The frame comprises several input fields that capture essential details such as the student's name, age, and a selected starter course. The student's ID is automatically generated by the StudentManagement class, ensuring a unique identifier for each student. To prevent invalid entries and errors, the frame incorporates dedicated text fields with built-in validation functionalities for the student's name and age. Additionally, a Swing combobox component

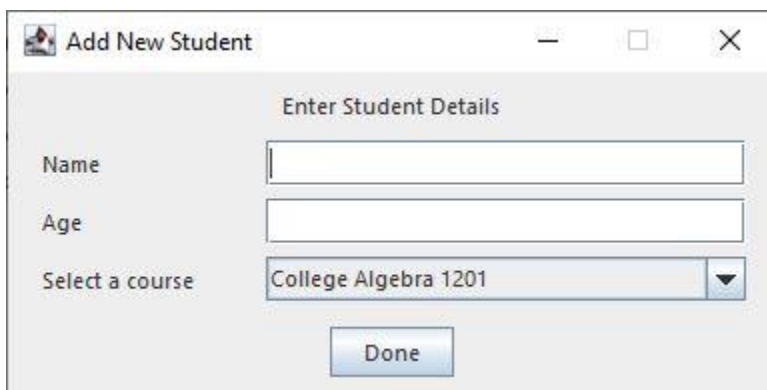
enables the user to select a starter course for the new student, offering a flexible and intuitive interface.

Upon completing the entry of the required information, the user proceeds by clicking the 'Done' button. The system then validates the input data, displaying an error popup if any discrepancies are detected. If the information is accurate, the StudentManagement class is updated with the new student's details, the AdministratorInterface is refreshed to reflect the changes, and the AddNewStudentFrame is disposed of, concluding the new student registration process.

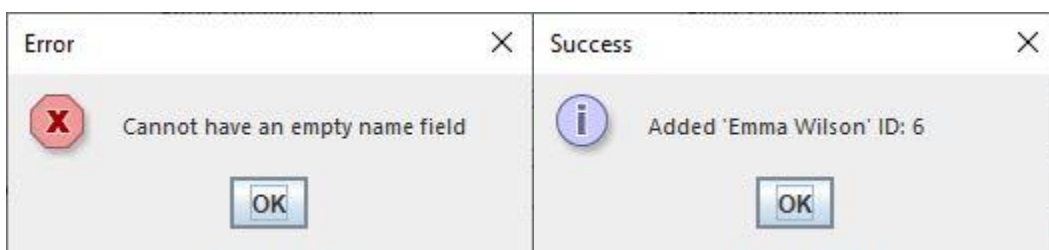
## Display



A screenshot of a Windows-style dialog box titled "Add New Student". It contains a section titled "Enter Student Details" with three input fields: "Name", "Age", and "Select a course". The "Name" and "Age" fields are empty text boxes, while "Select a course" is a dropdown menu. A "Done" button is located at the bottom right of the dialog.



A screenshot of the same "Add New Student" dialog box, but now with data entered. The "Name" field contains "Emma Wilson", the "Age" field contains "18", and the "Select a course" dropdown menu is set to "College Algebra 1201". The "Done" button is still present at the bottom right.



Two side-by-side screenshots of message boxes. The left one is titled "Error" and shows a red "X" icon with the text "Cannot have an empty name field" and an "OK" button. The right one is titled "Success" and shows a blue "i" icon with the text "Added 'Emma Wilson' ID: 6" and an "OK" button.

### Code Showcase:

```
package StudentManagement;

import java.awt.*;
import java.awt.event.*;

import javax.swing.*;
import javax.swing.border.EmptyBorder;
import javax.swing.GroupLayout.Alignment;
import javax.swing.LayoutStyle.ComponentPlacement;
import javax.swing.JOptionPane;

import java.text.NumberFormat;

public class AddNewStudent extends JFrame {

    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField textFieldName;

    // Constructor
    public AddNewStudent(int x, int y) {
        // Frame setup
        setResizable(false);
        setTitle("Add New Student");
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setBounds(x, y, 400, 200);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);

        // Labels
        JLabel lblEnterStudentDetails = new JLabel("Enter Student Details");
        lblEnterStudentDetails.setFont(new Font("Segoe UI", Font.PLAIN, 11));

        JLabel lblName = new JLabel("Name");
        lblName.setFont(new Font("Segoe UI", Font.PLAIN, 11));

        JLabel lblAge = new JLabel("Age");
        lblAge.setFont(new Font("Segoe UI", Font.PLAIN, 11));

        // Text fields
        textFieldName = new JTextField();
        textFieldName.setFont(new Font("Segoe UI", Font.PLAIN, 11));
        textFieldName.setColumns(10);

        NumberFormat integerFormat = NumberFormat.getIntegerInstance();
        integerFormat.setParseIntegerOnly(true);
        final JFormattedTextField textFieldAge = new
JFormattedTextField(integerFormat);
        textFieldAge.setFont(new Font("Segoe UI", Font.PLAIN, 11));
        textFieldAge.setColumns(10);
```

```

// Done Button
JButton btnDone = new JButton("Done");
btnDone.setFont(new Font("Segoe UI", Font.PLAIN, 11));

// Select course label and combo box
JLabel lblSelectCourse = new JLabel("Select a course");
lblSelectCourse.setFont(new Font("Segoe UI", Font.PLAIN, 11));

final JComboBox<Course> courseComboBox = new
JComboBox<>(StudentManagement.getCourseList());
courseComboBox.setFont(new Font("Segoe UI", Font.PLAIN, 11));
courseComboBox.setRenderer(new CourseListRenderer());

// Button action listener
btnDone.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String name = textFieldName.getText();
        String ageText = textFieldAge.getText();
        if (name.trim().isEmpty()) {
            showErrorPopup("Cannot have an empty name field");
        } else if (ageText.isEmpty()) {
            showErrorPopup("Cannot have an empty age field");
        } else {
            int age = Integer.parseInt(ageText);
            Course course =
courseComboBox.getItemAt(courseComboBox.getSelectedIndex());
            Student student = StudentManagement.enrollStudent(name,
age);

            StudentManagement.enrollStudentToCourse(student, course);
            AdministratorInterface.getInstance().refresh();
            showSuccessPopup("Added '" + student.getName() + "' ID: "
+ student.getID());

            dispose();
        }
    }
});

// Setup GroupLayout for contentPane **OMITTED**
GroupLayout gl_contentPane = new GroupLayout(contentPane);
contentPane.setLayout(gl_contentPane);
}

// Method to show error message popup
public void showErrorPopup(String message) {
    JLabel label = new JLabel(message);
    label.setFont(new Font("Segoe UI", Font.PLAIN, 11));

    JOptionPane.showMessageDialog(contentPane, label, "Error",
JOptionPane.ERROR_MESSAGE);
}

// Method to show success message popup
public void showSuccessPopup(String message) {
    JLabel label = new JLabel(message);

```

```

        label.setFont(new Font("Segoe UI", Font.PLAIN, 11));

        JOptionPane.showMessageDialog(contentPane, label, "Success",
JOptionPane.INFORMATION_MESSAGE);
    }
}

```

### **StudentDetails Frame:**

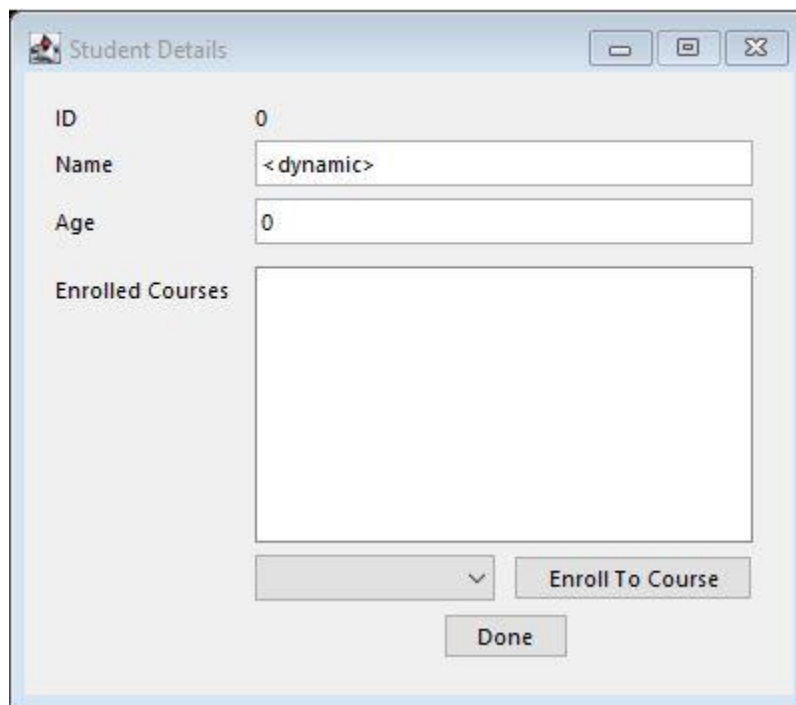
The StudentDetailsFrame is a graphical user interface (GUI) component within the AdministratorInterface specifically designed for viewing and modifying the information of a selected student. To access this frame, the user must first select a student's radio button and then click the 'view details' button; if no student is selected, an error popup is displayed, ensuring that only valid requests are processed.

Within the StudentDetailsFrame, there is an uneditable label that presents the selected student's ID, along with two editable text fields for displaying and modifying the student's name and age. Additionally, a table showcases the student's enrolled courses and current grades per course, where only the grades are editable by design.

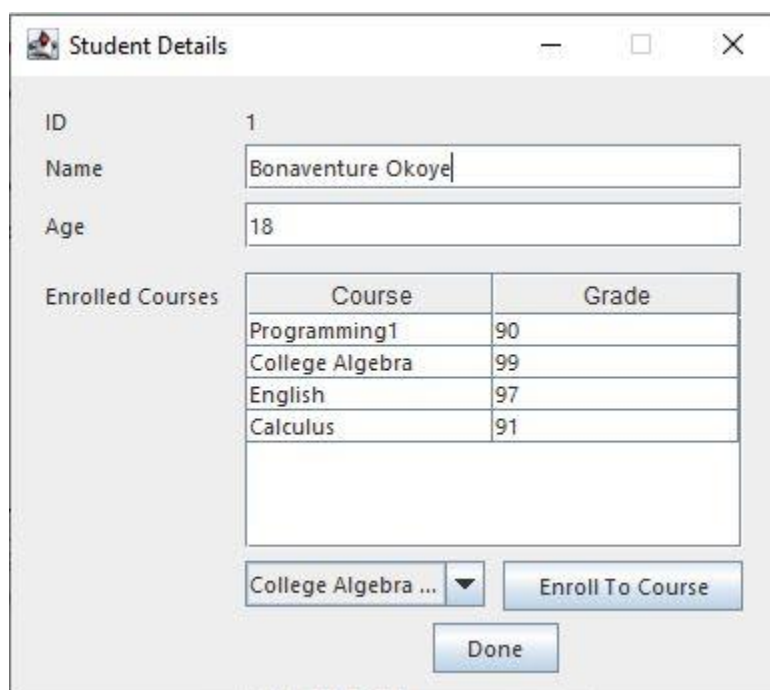
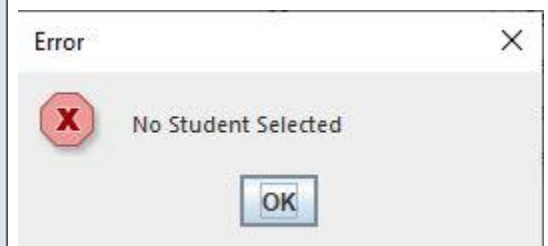
To enroll the student in a new course, the user can utilize the combobox located beneath the table to choose a course and then confirm their selection by clicking the 'Enroll to Course' button. This action refreshes the frame to include the newly added course in the table, ensuring an up-to-date view of the student's course enrollments.

Once the user has completed all desired changes, they can click the 'Done' button. At this point, the program compares the information in the frame with the data stored in the StudentManagement class to identify any differences. If changes have been made, the StudentManagement class is updated with the modified student details, ensuring that the edit operation is successfully applied. The system also validates the input data, displaying an error

popup if any discrepancies are detected. Subsequently, the AdministratorInterface is refreshed to reflect the changes, and the ViewStudentDetailsFrame is disposed of. Finally, a success popup is shown, signaling the completion of the student details modification process.



The 'Student Details' window shows a form for a student with ID 0. The Name field contains the placeholder text '<dynamic>', and the Age field contains '0'. The Enrolled Courses section is empty. At the bottom, there is a dropdown menu, an 'Enroll To Course' button, and a 'Done' button.



The 'Student Details' window shows a form for a student with ID 1. The Name field contains 'Bonaventure Okoye', and the Age field contains '18'. The Enrolled Courses section displays a table with the following data:

Course	Grade
Programming1	90
College Algebra	99
English	97
Calculus	91

Below the table, there is a dropdown menu showing 'College Algebra ...', an 'Enroll To Course' button, and a 'Done' button.



## Code Showcase:

```
package StudentManagement;

import java.awt.*;
import java.awt.event.*;

import javax.swing.*;
import javax.swing.border.EmptyBorder;
import javax.swing.GroupLayout.Alignment;
import javax.swing.LayoutStyle.ComponentPlacement;
import javax.swing.table.*;
import javax.swing.event.*;

import java.text.NumberFormat;
import java.text.Format;

public class StudentDetails extends JFrame {

    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField textFieldName;
    private JTable table;
    private Student student;
    private String studentName;
    private int studentAge;

    // Constructor
    public StudentDetails(Student student, int x, int y) {
        // Frame setup
        setResizable(false);
        setTitle("Student Details");
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setBounds(x, y, 400, 350);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);

        this.student = student;
        this.studentName = student.getName();
        this.studentAge = student.getAge();

        // Labels
        JLabel lblID = new JLabel("ID");
        lblID.setFont(new Font("Segoe UI", Font.PLAIN, 11));

        JLabel lblStudentID = new JLabel(String.valueOf(student.getID()));
        lblStudentID.setFont(new Font("Segoe UI", Font.PLAIN, 11));

        JLabel lblAge = new JLabel("Age");
        lblAge.setFont(new Font("Segoe UI", Font.PLAIN, 11));
```

```

JLabel lblManageCourses = new JLabel("Enrolled Courses");
lblManageCourses.setFont(new Font("Segoe UI", Font.PLAIN, 11));

JLabel lblName = new JLabel("Name");
lblName.setFont(new Font("Segoe UI", Font.PLAIN, 11));

// Text fields
textFieldName = new JTextField(student.getName());
textFieldName.setFont(new Font("Segoe UI", Font.PLAIN, 11));
textFieldName.setColumns(10);

NumberFormat integerFormat = NumberFormat.getIntegerInstance();
integerFormat.setParseIntegerOnly(true);

final JFormattedTextField textFieldAge = new
JFormattedTextField((Format) null);
textFieldAge.setFont(new Font("Segoe UI", Font.PLAIN, 11));
textFieldAge.setColumns(10);
textFieldAge.setValue(student.getAge());

// Buttons
JButton btnDone = new JButton("Done");
btnDone.setFont(new Font("Segoe UI", Font.PLAIN, 11));

JButton btnEnrollToCourse = new JButton("Enroll To Course");
btnEnrollToCourse.setFont(new Font("Segoe UI", Font.PLAIN, 11));

// Combo box
final JComboBox<Course> courseComboBox = new
JComboBox<>(StudentManagement.getCourseList());
courseComboBox.setFont(new Font("Segoe UI", Font.PLAIN, 11));
courseComboBox.setRenderer(new CourseListRenderer());

// Scroll pane
JScrollPane coursesScrollPane = new JScrollPane();

// Button action listeners
btnDone.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String newName = textFieldName.getText();
        String ageText = textFieldAge.getText();

        if (newName.trim().isEmpty()) {
            showErrorPopup("Cannot have an empty name field");
        } else if (ageText.isEmpty()) {
            showErrorPopup("Cannot have an empty age field");
        } else {
            int newAge = Integer.parseInt(ageText);

            if (!studentName.equals(newName)) {
                getStudent().setName(newName);
                showSuccessPopup("Changed Student Name to " +
newName);
            }
            if (studentAge != newAge) {

```

```

        getStudent().setAge(newAge);
        showSuccessPopup("Changed Student Age to " +
newAge);
    }

    AdministratorInterface.getInstance().refresh();
    dispose();
}
});

btnEnrollToCourse.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        Student student = getStudent();
        Course course =
courseComboBox.getItemAt(courseComboBox.getSelectedIndex());
        StudentManagement.enrollStudentToCourse(student, course);
        showSuccessPopup("Enrolled " + student.getName() + " to " +
course.getName());
        refresh();
    }
});

// Setup GroupLayout for contentPane **OMITTED**
GroupLayout gl_contentPane = new GroupLayout(contentPane);
contentPane.setLayout(gl_contentPane);

// Table setup
final DefaultTableModel model = new DefaultTableModel(){
    @Override
    public boolean isCellEditable(int row, int column) {
        return column != 0;
    }
};
model.addColumn("Course");
model.addColumn("Grade");
model.setColumnIdentifiers(new Object[]{"Course", "Grade"});

Course[] courses = student.getEnrolledCourses().toArray(new Course[0]);
for (int i = 0; i < courses.length; i++) {
    Course course = courses[i];
    int grade = student.getGradeForCourse(course);
    model.addRow(new Object[]{course.getName(), grade});
}

table = new JTable(model);
coursesScrollPane.setViewportView(table);
table.setFillViewportHeight(true);
table.setFont(new Font("Segoe UI", Font.PLAIN, 11));
table.setSelectionBackground(table.getBackground());
table.getColumnModel().setCellEditor(new DefaultCellEditor(new
JTextField()));

table.getModel().addTableModelListener(new TableModelListener() {
    public void tableChanged(TableModelEvent e) {

```

```

        if (e.getType() == TableModelEvent.UPDATE) {
            int row = e.getFirstRow();
            int column = e.getColumn();

            // If the grade column was updated
            if (column == 1) {
                String courseName = (String)
table.getModel().getValueAt(row, 0);

                Course selectedCourse = null;
                for (Course course:StudentManagement.getCourseList()) {
                    if (course.getName().equals(courseName)) {
                        selectedCourse = course;
                    }
                }
                int updatedGrade = Integer.parseInt( (String)
table.getModel().getValueAt(row, column));
                StudentManagement.assignGrade(getStudent(), selectedCourse,
updatedGrade);
            }
        }
    });
}

// Getter for student
public Student getStudent() {
    return student;
}

// Method to display error popup
public void showErrorPopup(String message) {
    JLabel label = new JLabel(message);
    label.setFont(new Font("Segoe UI", Font.PLAIN, 11));

    JOptionPane.showMessageDialog(contentPane, label, "Error",
JOptionPane.ERROR_MESSAGE);
}

// Method to display success popup
public void showSuccessPopup(String message) {
    JLabel label = new JLabel(message);
    label.setFont(new Font("Segoe UI", Font.PLAIN, 11));

    JOptionPane.showMessageDialog(contentPane, label, "Success",
JOptionPane.INFORMATION_MESSAGE);
}

// Method to refresh the frame
public void refresh() {
    int x = this.getX();
    int y = this.getY();

    StudentDetails studentDetailsFrame = new StudentDetails(student, x, y);
    studentDetailsFrame.setVisible(true);
    this.dispose();
}
}

```

## StudentManagement Class

The StudentManagement class serves as the backbone of the application and is responsible for managing the overall functionality of the system, including the coordination between the graphical user interface (GUI) and the backend data management.

```
package StudentManagement;
import java.util.ArrayList;
import java.util.HashMap;

public class StudentManagement {
    private static final ArrayList<Course> courses = new ArrayList<>();
    private static final HashMap<Student, HashMap<Course, Integer>> studentList = new
HashMap<>();
    private static final HashMap<Student, Integer> studentGrades = new HashMap<>();
    private static final HashMap<Student, Integer> studentIDs = new HashMap<>();

    public static void updateStudentName(Student student, String newName) {
        student.setName(newName);
    }

    public static void updateStudentAge(Student student, int newAge) {
        student.setAge(newAge);
    }

    public static Course addCourse(String name, int code, int maximumCapacity) {
        Course course = new Course(name, code, maximumCapacity);
        courses.add(course);
        return course;
    }

    public static Student enrollStudent(String name, int age) {
        int ID = 1;
        int maxID = 0;
        if (!studentIDs.isEmpty()) {
            for (int i : studentIDs.values()) {
                if (i > maxID) {
                    maxID = i;
                }
            }
            ID = maxID+1;
        }

        Student student = new Student(ID, name, age);
        studentIDs.put(student, ID);
        studentList.put(student, student.getCourses());
        return student;
    }

    public static HashMap<Student, HashMap<Course, Integer>> getStudentList() {
        return studentList;
    }
}
```

```

    }

    public static void enrollStudentToCourse(Student student, Course course) {
        student.enrollCourse(course);
    }

    public static void assignGrade(Student student, Course course, int grade) {
        student.assignGrade(course, grade);
        calculateOverallGrade(student);
    }

    public static void calculateOverallGrade(Student student) {
        if (student.courseGrades.isEmpty()) {
            studentGrades.put(student, 0);
        } else {
            int sum = 0;
            for (int grade : student.courseGrades.values()) {
                sum += grade;
            }
            studentGrades.put(student, sum / student.courseGrades.size());
        }
    }

    public static Course[] getCourseList() {
        return courses.toArray(new Course[0]);
    }

    public static int getStudentOverallGrade(Student student) {
        if (studentGrades.get(student) == null) {
            return 0;
        }
        return studentGrades.get(student);
    }
}

```

## Student Class

The Student class represents a student in the student management system. It contains fields for the student's name, age, ID, and a HashMap to store the student's grades for each enrolled course. The class provides methods to access and modify these fields, as well as to manage the student's course enrollments and grades.

```
package StudentManagement;
import java.util.ArrayList;
import java.util.HashMap;

public class Student {
    private String name;
    private int age;
    private int ID;
    public HashMap<Course, Integer> courseGrades = new HashMap<>();

    public Student(int ID, String name, int age) {
        this.name = name;
        this.ID = ID;
        this.age = age;
    }

    public int getID() {
        return this.ID;
    }

    public void setID(int newID) {
        this.ID = newID;
    }

    public String getName() {
        return this.name;
    }

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

    public int getAge() {
        return age;
    }

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

```

public ArrayList<Course> getEnrolledCourses() {
    ArrayList<Course> enrolledCourses = new ArrayList<Course>();
    for (Course course : this.courseGrades.keySet()) {
        enrolledCourses.add(course);
    }
    return enrolledCourses;
}

public void enrollCourse(Course newCourse) {
    if (!this.courseGrades.containsKey(newCourse)) {
        this.courseGrades.put(newCourse, 0);
        Course.enrolledStudentsCount++;
    }
}

public HashMap<Course, Integer> getCourses() {
    return this.courseGrades;
}

public void assignGrade(Course course, int grade) {
    this.courseGrades.put(course, grade);
}

public int getGradeForCourse(Course course) {
    return this.courseGrades.get(course);
}
}

```



## Course class

The Course class represents a course in the student management system. It contains fields for the course's name, course code, maximum capacity, a list of enrolled students, and a HashMap to store the grades for each enrolled student. The class provides methods to access and modify these fields, as well as to manage the course's enrolled students and their grades.

```
package StudentManagement;
import java.util.ArrayList;
import java.util.HashMap;

public class Course {
    private String name;
    private int courseCode;
    private int maximumCapacity;
    private ArrayList<Student> enrolledStudents = new ArrayList<>();
    private HashMap<Student, Integer> studentGrades = new HashMap<>();
    public static int enrolledStudentsCount;

    public Course(String name, int courseCode, int maximumCapacity) {
        this.name = name;
        this.courseCode = courseCode;
        this.maximumCapacity = maximumCapacity;
    }

    public int getCourseCode() {
        return this.courseCode;
    }

    public void setCourseCode(int newCode) {
        this.courseCode = newCode;
    }

    public int getMaximumCapacity() {
        return this.maximumCapacity;
    }

    public void setMaximumCapacity(int newCapacity) {
        this.maximumCapacity = newCapacity;
    }

    public String getName() {
        return this.name;
    }

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

```
public ArrayList<Student> getEnrolledStudents() {  
    return this.enrolledStudents;  
}  
  
public void enrollStudent(Student student) {  
    if (this.enrolledStudents.size() < this.maximumCapacity) {  
        this.enrolledStudents.add(student);  
        this.studentGrades.put(student, 0);  
    } else {  
        System.out.println("Error: Course is at maximum capacity");  
    }  
}  
}
```