# CS4001NI Programming

## 30% Individual Coursework

## 2023-24 Autumn

**Student Name: Evani Raut**

**London Met ID: 23047473**

**College ID: NP01NT4A230151**

**Group: N6**

**Assignment Due Date: Friday, May 10, 2024**

**Assignment Submission Date: Friday, May 10, 2024**

## Contents

## Table of Figures:

## Table of Tables:

# 1. Introduction:

## 1.1 About Project:

The development and implementation of a graphical user interface (GUI) to manage an array list of Teacher objects (including Lecturer and Tutor classes) forms the primary focus of this project. This project serves as a comprehensive exploration of object-oriented programming concepts and GUI design in Java. The goal is to create a user-friendly application that allows for the management of a list of teachers, including the ability to add, grade, set salaries, remove tutors, and display information about the teacher list.

The application will utilize an array list to store objects of Teacher types and will feature various text fields and buttons to allow for user interaction with the stored teacher data. This includes functionalities such as adding Lecturers and Tutors, grading assignments, setting Tutor salaries, and removing Tutors. By implementing these functionalities and providing user input validations, the program aims to offer a reliable and intuitive interface for managing teacher-related data.

**Assessment:**

The assessment involves the development of a comprehensive Java GUI application for managing teacher data. This includes creating an intuitive and functional interface for adding, modifying, and removing Lecturer and Tutor objects within an array list. Additionally, the assessment involves implementing user-friendly controls and error handling mechanisms to ensure smooth interaction with the application.

The application must be capable of handling various teacher attributes such as ID, name, address, working type, employment status, working hours, salary, specialization, and performance index. Testing and evaluation will be conducted to ensure the application meets the functional and usability requirements outlined in the project brief.

**Aim:**

The aim of this project is to design and implement a user-friendly Java GUI application that provides robust functionality for managing teachers. This includes creating, modifying, and removing Lecturer and Tutor objects within an array list. The application should offer a seamless experience for entering and manipulating teacher data, grading assignments, setting tutor salaries, and removing tutors. The interface should include clear feedback and error messages for invalid inputs, ensuring an efficient and reliable user experience.

**Summary:**

The project focuses on the creation of a Teacher management GUI application that handles Lecturer and Tutor objects. The application will provide a variety of buttons and text fields to allow users to interact with the system, including adding new teachers, grading assignments, setting salaries, and removing tutors from the system. Proper input validation and error handling will ensure the program's robustness and user satisfaction.

**Deliverables:**

The project will deliver:

- A functional Java GUI application for managing teachers.
- A user-friendly interface with buttons and text fields to interact with the teacher list.
- Clear input validation and error handling for user input.
- A well-documented codebase, including a class diagram and pseudocode.
- A report detailing the design and functionality of the application, as well as a discussion of testing, error handling, and reflection on the development process.

## 1.2 Tools used

### BlueJ

*Figure 1:BlueJ*

In my project, I selected BlueJ as the primary development environment for coding. It's simplicity enhances code readability and fosters a clear understanding of class structures, making it an ideal choice for building and organizing my project's codebase. BlueJ's interactive and visual approach to Java development is particularly beneficial for educational purposes and projects centered around Object-Oriented programming.

### Microsoft World

*Figure 2:MS Word*

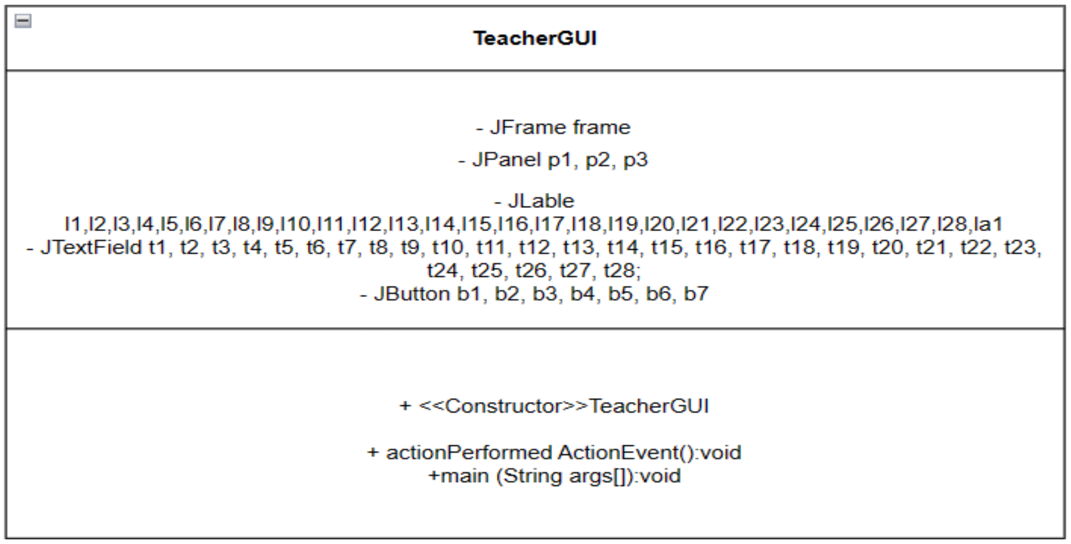
For documentation task, I choose Microsoft word as a reliable tool. MS word provides a robust platform for creating and formatting project documentation. Its rich set of features supports the creation of detailed documents, allowing me to articulate project requirements, design specifications, and user manuals with ease.

### Draw.io

*Figure 3:Draw.io*

To visually represent the architecture and relationships between classes in my project, I turned to draw.io. As a web-based diagramming tool, it offers an intuitive interface for creating various diagrams, including class diagrams. It’s flexibility and ease make it valuable asset in conveying the design aspects of my project.

## 2.Class Diagram



**TeacherGUI**

- JFrame frame
- JPanel p1, p2, p3
- JLable l1,l2,l3,l4,l5,l6,l7,l8,l9,l10,l11,l12,l13,l14,l15,l16,l17,l18,l19,l20,l21,l22,l23,l24,l25,l26,l27,l28,la1
- JTextField t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12, t13, t14, t15, t16, t17, t18, t19, t20, t21, t22, t23, t24, t25, t26, t27, t28;
- JButton b1, b2, b3, b4, b5, b6, b7

+ <<Constructor>>TeacherGUI
+ actionPerformed ActionEvent():void
+main (String args[]):void

*Figure 4:Class Diagram of TeacherGUI*

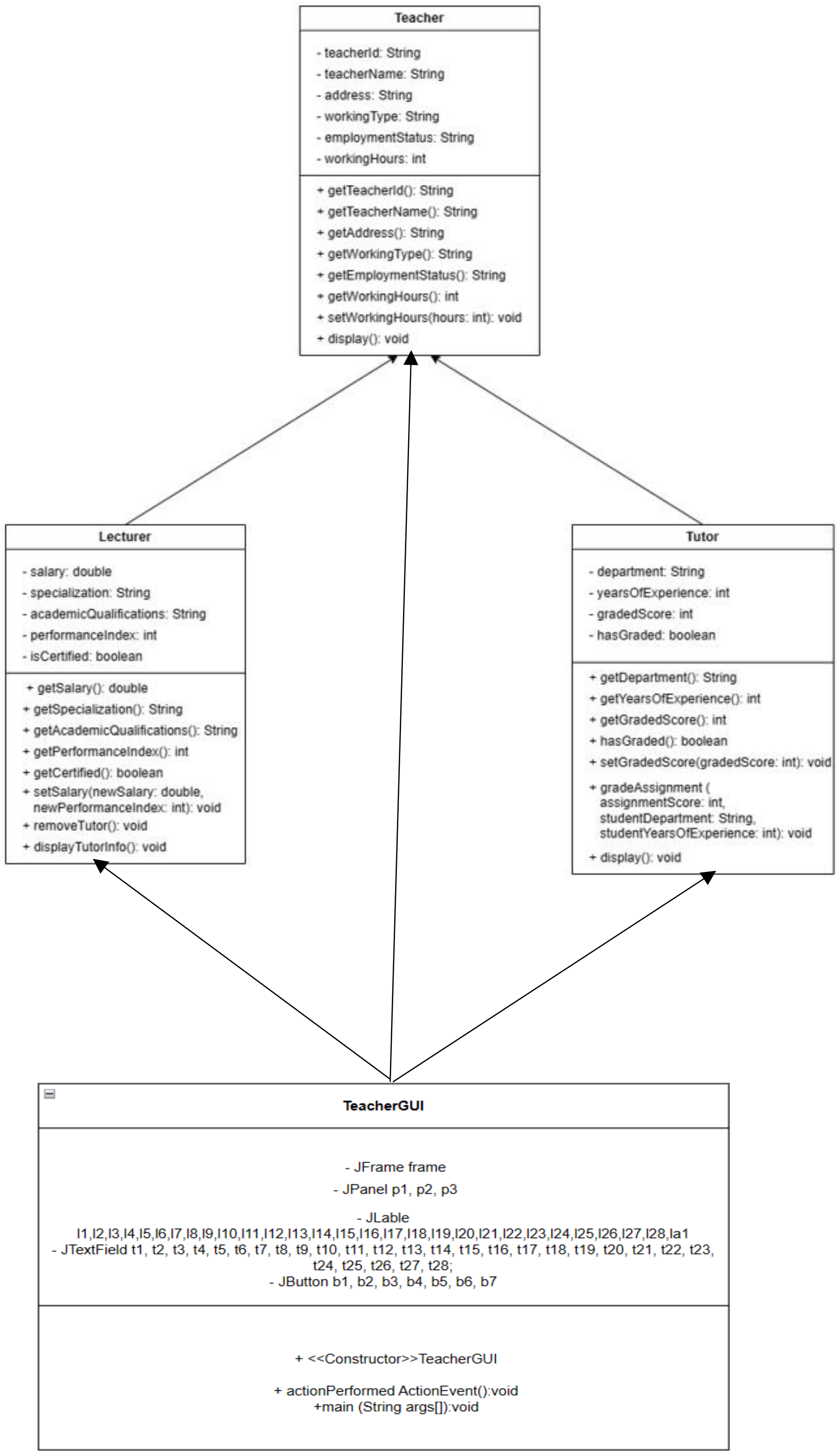

Teacher
- teacherId: String
- teacherName: String
- address: String
- workingType: String
- employmentStatus: String
- workingHours: int
+ getTeacherId(): String
+ getTeacherName(): String
+ getAddress(): String
+ getWorkingType(): String
+ getEmploymentStatus(): String
+ getWorkingHours(): int
+ setWorkingHours(hours: int): void
+ display(): void
Lecturer
- salary: double
- specialization: String
- academicQualifications: String
- performanceIndex: int
- isCertified: boolean
+ getSalary(): double
+ getSpecialization(): String
+ getAcademicQualifications(): String
+ getPerformanceIndex(): int
+ getCertified(): boolean
+ setSalary(newSalary: double, newPerformanceIndex: int): void
+ removeTutor(): void
+ displayTutorInfo(): void
Tutor
- department: String
- yearsOfExperience: int
- gradedScore: int
- hasGraded: boolean
+ getDepartment(): String
+ getYearsOfExperience(): int
+ getGradedScore(): int
+ hasGraded(): boolean
+ setGradedScore(gradedScore: int): void
+ gradeAssignment ( assignmentScore: int, studentDepartment: String, studentYearsOfExperience: int): void
+ display(): void
TeacherGUI
- JFrame frame
- JPanel p1, p2, p3
- JLable l1,l2,l3,l4,l5,l6,l7,l8,l9,l10,l11,l12,l13,l14,l15,l16,l17,l18,l19,l20,l21,l22,l23,l24,l25,l26,l27,l28,la1
- JTextField t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12, t13, t14, t15, t16, t17, t18, t19, t20, t21, t22, t23, t24, t25, t26, t27, t28;
- JButton b1, b2, b3, b4, b5, b6, b7
+ <<Constructor>>TeacherGUI
+ actionPerformed ActionEvent():void
+main (String args[]):void

*Figure 5:Relationship between all classes*

## 3.Pseudocode:

```
Start Program


 // Class Definition
 Declare Class: TeacherGUI
     Initialize `mylistk` as an ArrayList of Teachers


     // Constructor
     Begin Constructor
         Create JFrame with title "Teacher"
             Set frame visibility to true
             Set frame size to 1500x800
             Set frame layout to null
             Set frame background color


         // Initialize and add Adding Tutor panel
         Create and Add Panel: `initializeAddingTutorPanel()`
             Set bounds and background color
             Add title label "Adding Tutor" to panel
             Declare and initialize components for adding a tutor (labels and text fields)
             Add "Add a Tutor" button with action listener
                 On button click
```

```
            Validate input fields
            If fields are filled
                Create Tutor instance with input data
                Add tutor to `mylistk`
                Display success message
            Else
                Display error message for empty fields
        Catch exceptions for invalid input values


    Add title label "Set Salary of Tutor" to panel
    Declare and initialize components for setting tutor's salary (labels and text fields)
    Add "Set Salary of Tutor" button with action listener
        On button click
            Validate input values
            Find Tutor by ID and set new salary and performance index
            Display success or failure message based on tutor ID match
        Catch exceptions for invalid input values
Return initialized panel


// Initialize and add Grade Assignments panel
Create and Add Panel: `initializeGradeAssignmentsPanel()`
    Set bounds and background color
    Add title label "Grade Assignments" to panel
    Declare and initialize components for grading assignments (labels and text fields)
    Add "Grade the Assignments" button with action listener
        On button click
            Validate input values
            Find Lecturer by ID and grade the assignment
            Display grading result or failure message based on lecturer ID match
        Catch exceptions for invalid input values

    Add title label "Remove Tutor" to panel
    Declare and initialize components for removing tutor (labels and text fields)
    Add "Remove the tutor" button with action listener
        On button click
            Validate input value
            Find and remove Tutor by ID from list
            Display success or failure message based on Tutor ID match
        Catch exceptions for invalid input values
Return initialized panel

// Initialize and add Adding Lecturer panel
Create and Add Panel: `initializeAddingLecturerPanel()`
    Set bounds and background color
    Add title label "Adding Lecturer" to panel
    Declare and initialize components for adding a lecturer (labels and text fields)
    Add "Add a Lecturer" button with action listener
        On button click
            Validate input fields
            If fields are filled
                Create Lecturer instance with input data
                Add lecturer to `mylistk`
                Display success message
            Else
                Display error message for empty fields
        Catch exceptions for invalid input values
Return initialized panel

// Initialize and add display and clear buttons
Declare and Add "Display" button with action listener
    On button click
        Iterate through `mylistk`
        Display information of each Teacher
Declare and Add "Clear" button with action listener
    On button click
        Clear all `JTextField` variables (`t1` to `t28`)
        Clear `mylistk`
```

```
            Display success message
        End Constructor


    // Main Function
    Function `main()`
        Create an instance of `TeacherGUI` class to initialize the application


End Program
```

## 4.Method Description:

**Constructor: TeacherGUI()**

- **Purpose**: Initializes the main frame of the application and its components.
- **Functionality**: Sets up the main frame with a specified title, size, and layout. Sets the background color of the frame. Adds the initialized panels (**initializeAddingTutorPanel()**, **initializeGradeAssignmentsPanel()**, and **initializeAddingLecturerPanel()**) to the frame. Also initializes and adds the display and clear buttons (**initializeDisplayAndClearButtons(frame)**) to the frame.

**Method: initializeAddingTutorPanel()**

- **Purpose**: Initializes a panel (**JPanel**) for adding a **Tutor** to the list.
- **Functionality**: Sets up a panel (**p1**) with labels and text fields for entering tutor details such as teacher ID, name, address, working type, employment status, working hours, salary, specialization, academic qualifications, and performance index. Includes a button (**Add a Tutor**) that, when clicked, adds a **Tutor** to the list (**mylistk**). Also includes fields and a button (**Set Salary of Tutor**) for setting a

tutor's salary and performance index based on the teacher ID. Displays appropriate messages depending on whether the addition or setting of a tutor was successful.

**Method: initializeGradeAssignmentsPanel()**

- **Purpose**: Initializes a panel (**JPanel**) for grading assignments and removing tutors.
- **Functionality**: Sets up a panel (**p2**) with labels and text fields for entering details such as teacher ID, graded score, department, and years of experience. Includes a button (**Grade the Assignments**) for grading assignments, which checks if a lecturer with the given teacher ID exists and then grades assignments accordingly. Also includes fields and a button (**Remove the tutor**) for removing a tutor based on the teacher ID provided. Displays appropriate messages depending on whether the grading or removal operation was successful.

**Method: initializeAddingLecturerPanel()**

- **Purpose**: Initializes a panel (**JPanel**) for adding a **Lecturer** to the list.
- **Functionality**: Sets up a panel (**p3**) with labels and text fields for entering lecturer details such as teacher ID, name, address, working type, employment status, graded score, years of experience, and department. Includes a button (**Add a Lecturer**) that, when clicked, adds a **Lecturer** to the list (**mylistk**). Displays appropriate messages depending on whether the addition was successful.

**Method: initializeDisplayAndClearButtons(JFrame frame)**

- **Purpose**: Initializes buttons for displaying and clearing the list of teachers.
- **Functionality**: Sets up and adds two buttons (**Display** and **Clear**) to the provided frame. The **Display** button, when clicked, displays details of all the **Teacher** objects (both **Tutor** and **Lecturer**) in the list (**mylistk**). The **Clear** button, when clicked, clears all text fields and removes all entries from the list of teachers in memory. Displays a message to the user indicating that fields and records have been cleared.

**Main Method: main(String[] args)**

- **Purpose**: Entry point for the application.
- **Functionality**: Instantiates the **TeacherGUI** class, which initializes the application and displays the graphical user interface (w3schools, 2024).

## 5.Testing:

### 5.1Test-1: Test if the program can be complied and run using the command prompt

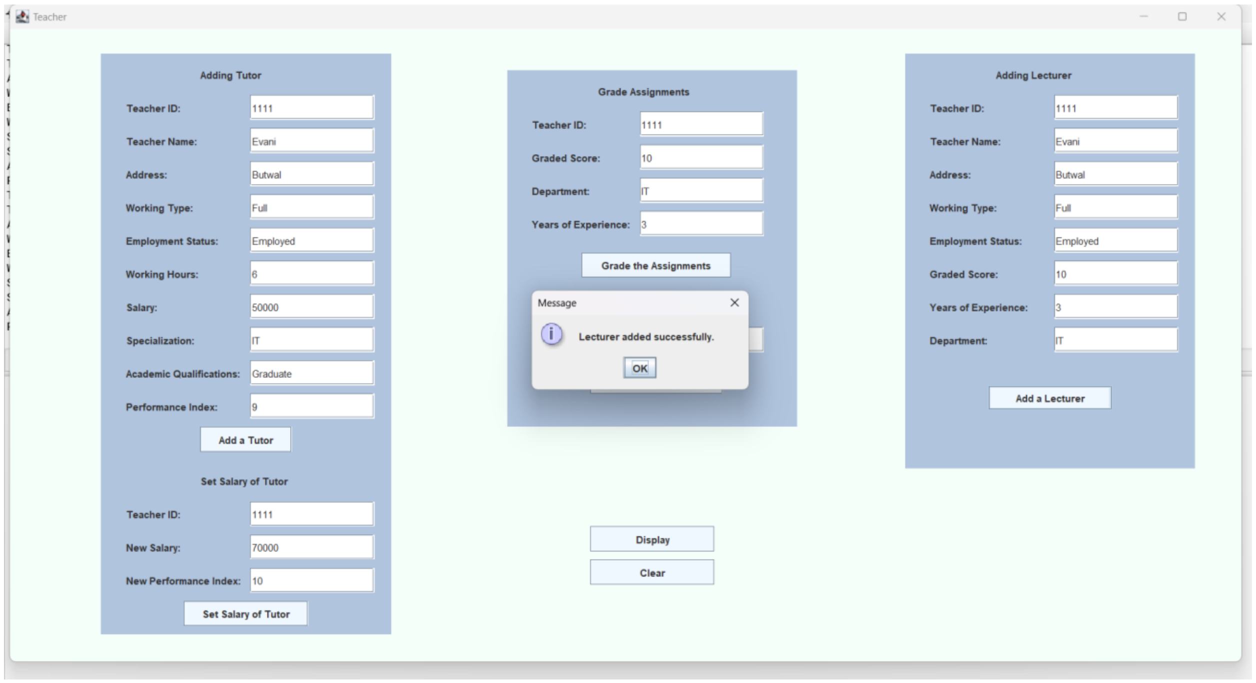
| | |
|---|---|
| Objectives | To verify if the software is operating via the command line. |
| Action | Open the blue J and command prompt then add the file location on cmd. |
| Expected Result | When you type the file name into the command prompt, the software ought to launch. |
| Actual Result | The command is operated from command prompt. |
| Conclusion | The test was successful. |

*Table 1:Test 1*

## 5.2 Test 2.1: To show the evidences of Adding to Lecturer

| | |
|---|---|
| Objectives | The proof of adding to lecturer button the Lecturer is shown. |
| Action | The values are added on GUI for Lecturer. |
| Expected Result | When we tap on add lecturer button the lecture should be added. |
| Actual Result | The lecturer wad added. |
| Conclusion | The test was successful. |

*Table 2:Test 2.1*



Teacher

Adding Tutor

| | |
|---|---|
| Teacher ID: | 1111 |
| Teacher Name: | Evani |
| Address: | Butwal |
| Working Type: | Full |
| Employment Status: | Employed |
| Working Hours: | 6 |
| Salary: | 50000 |
| Specialization: | IT |
| Academic Qualifications: | Graduate |
| Performance Index: | 9 |

Add a Tutor

Set Salary of Tutor

| | |
|---|---|
| Teacher ID: | 1111 |
| New Salary: | 70000 |
| New Performance Index: | 10 |

Set Salary of Tutor

Grade Assignments

| | |
|---|---|
| Teacher ID: | 1111 |
| Graded Score: | 10 |
| Department: | IT |
| Years of Experience: | 3 |

Grade the Assignments

Message

Lecturer added successfully.

OK

Display

Clear

Adding Lecturer

| | |
|---|---|
| Teacher ID: | 1111 |
| Teacher Name: | Evani |
| Address: | Butwal |
| Working Type: | Full |
| Employment Status: | Employed |
| Graded Score: | 10 |
| Years of Experience: | 3 |
| Department: | IT |

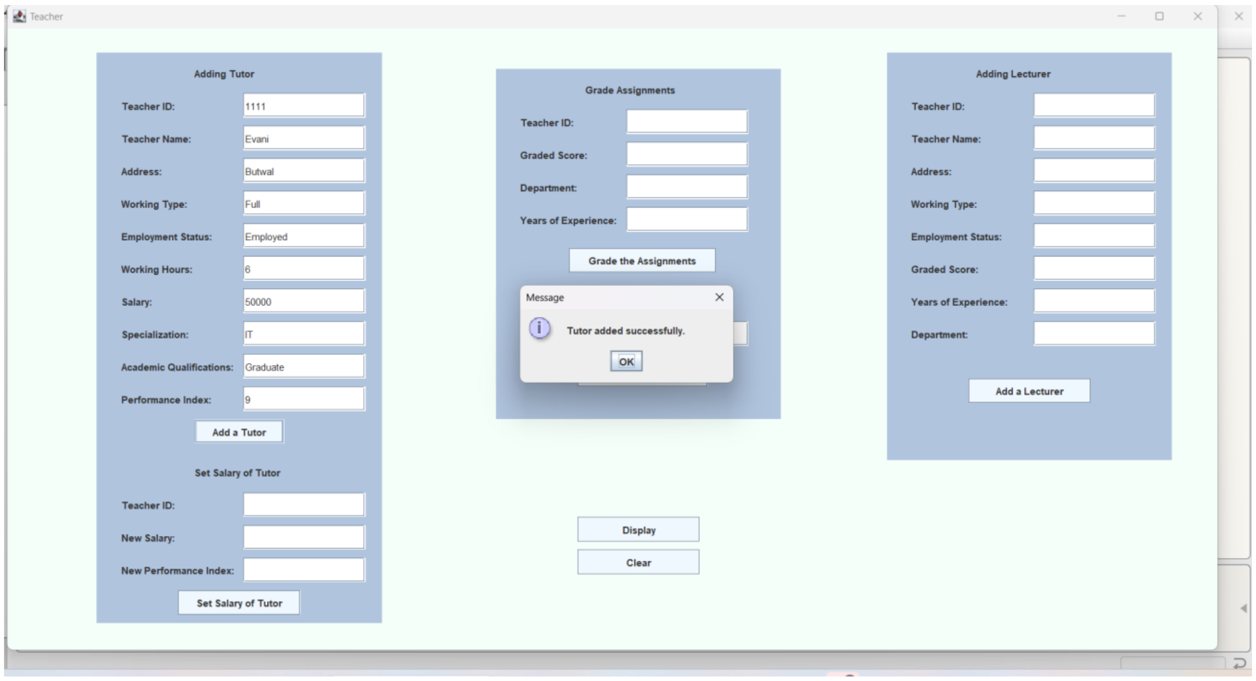
Add a Lecturer

*Figure 6:Test 2.1*

**Test 2.2: To show the evidences of Adding to tutor.**

| | |
|---|---|
| Objectives | The proof of adding to tutor is shown. |
| Action | Added the values on GUI for tutor. |
| Expected Result | When we tap on add tutor button the tutor should be added. |
| Actual Result | The tutor was added. |
| Conclusion | The test was successful. |

*Table 3:Test 2.2*



| | |
|---|---|
| Teacher ID: | 1111 |
| Teacher Name: | Evani |
| Address: | Butwal |
| Working Type: | Full |
| Employment Status: | Employed |
| Working Hours: | 6 |
| Salary: | 50000 |
| Specialization: | IT |
| Academic Qualifications: | Graduate |
| Performance Index: | 9 |

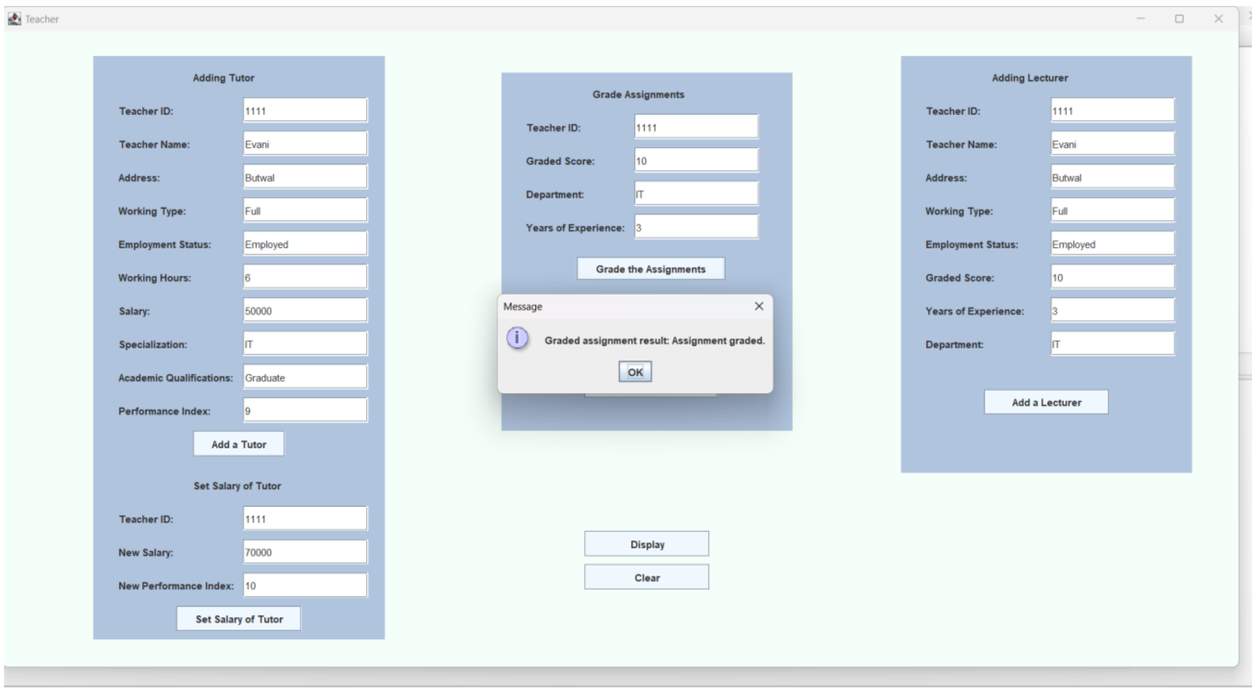
Tutor added successfully.

*Figure 7:Test 2.2*

## Test 2.3: To show the evidences of Grade Assignments.

| | |
|---|---|
| Objective | The proof of adding to Grade assignment is shown. |
| Action | Added the values on GUI to grade assignments. |
| Expected Result | When the tap on add grade assignment button the tutor should be added. |
| Actual Result | The grade assignment was added. |
| Conclusion | The test was successful. |

*Table 4:Test 2.3*


Teacher

Adding Tutor

Teacher ID: 1111
Teacher Name: Evani
Address: Butwal
Working Type: Full
Employment Status: Employed
Working Hours: 6
Salary: 50000
Specialization: IT
Academic Qualifications: Graduate
Performance Index: 9

Set Salary of Tutor

Teacher ID: 1111
New Salary: 70000
New Performance Index: 10

Grade Assignments

Teacher ID: 1111
Graded Score: 10
Department: IT
Years of Experience: 3

Message

Graded assignment result: Assignment graded.

Adding Lecturer

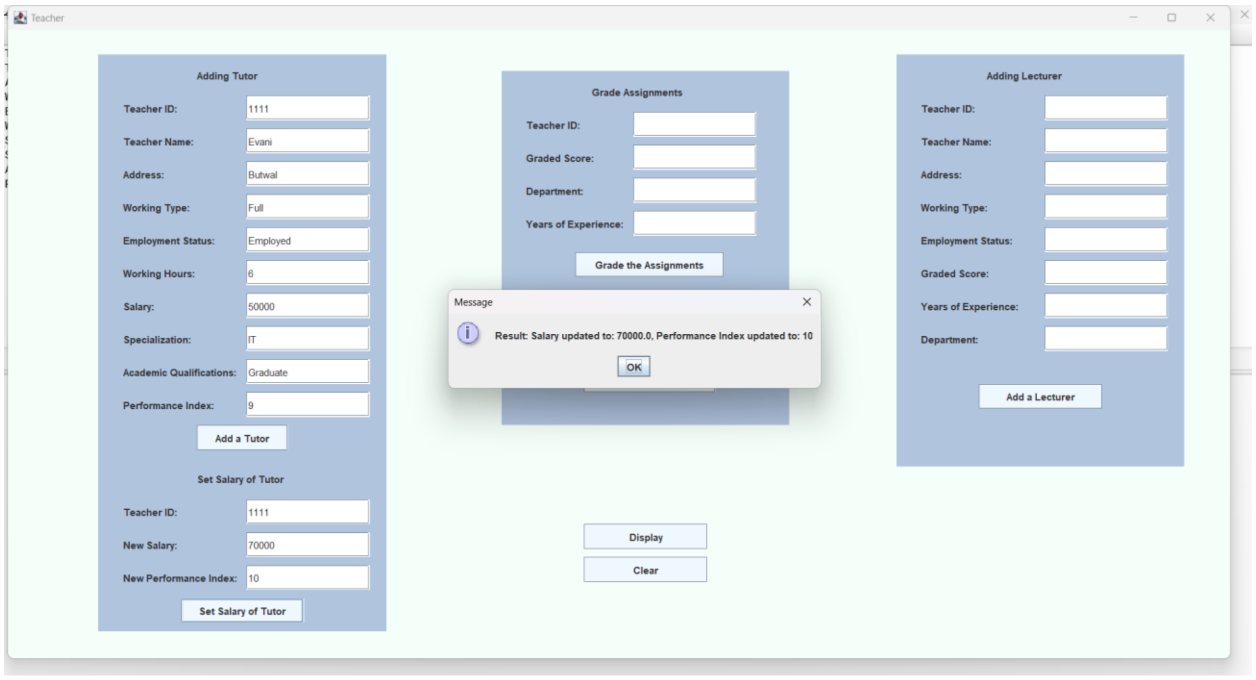
Teacher ID: 1111
Teacher Name: Evani
Address: Butwal
Working Type: Full
Employment Status: Employed
Graded Score: 10
Years of Experience: 3
Department: IT

*Figure 8:Test 2.3*

**Test 2.4** : To show the evidence of Setting salary

| | |
|---|---|
| Objective | The proof of setting salary is shown. |
| Action | Added the new salary on GUI for setting salary. |
| Expected Result | When we tap on Set salary button the salary of the tutor should be updated. |
| Actual Result | When we click on Set Salary button the salary of the tutor was updated. |
| Conclusion | The test was successful. |

*Table 5:Test 2.4*



Teacher ID: 1111
Teacher Name: Evani
Address: Butwal
Working Type: Full
Employment Status: Employed
Working Hours: 6
Salary: 50000
Specialization: IT
Academic Qualifications: Graduate
Performance Index: 9

Teacher ID: 1111
New Salary: 70000
New Performance Index: 10

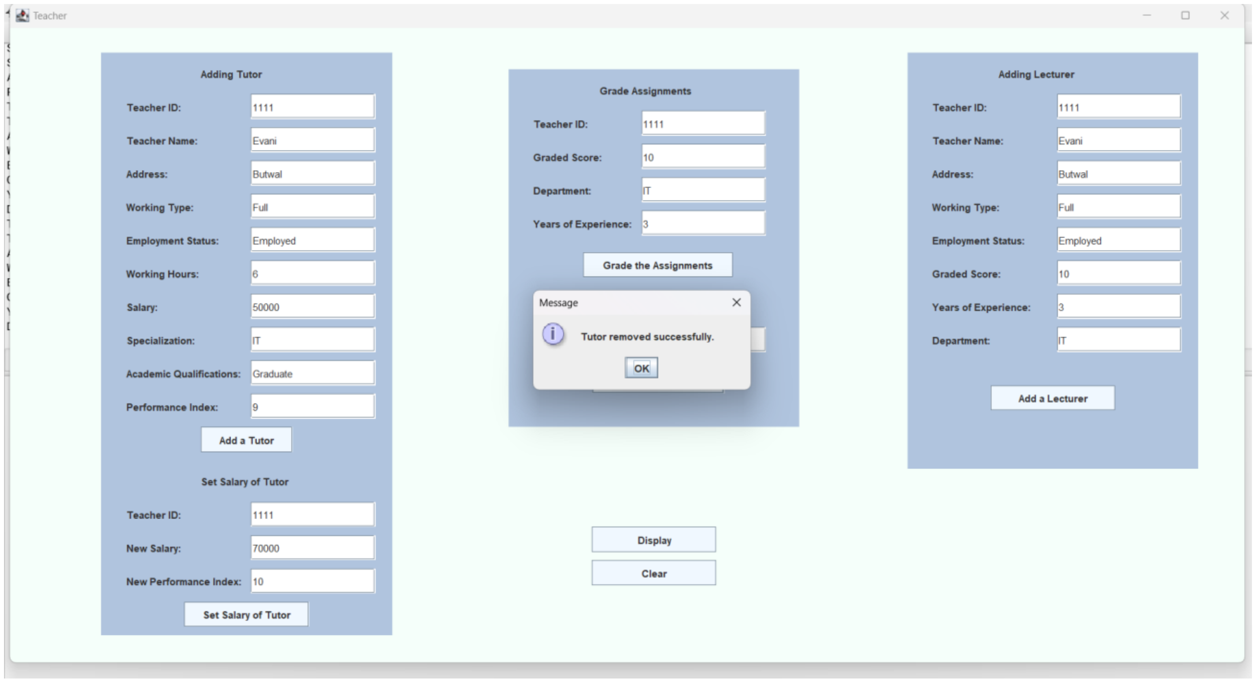
Result: Salary updated to: 70000.0, Performance Index updated to: 10

*Figure 9:Test 2.4*

## 2.5 Test: To show the evidence of Remove Tutor.

| | |
|---|---|
| Objective | The proof of Remove Tutor is shown. |
| Action | Added an existing tutor id then clicked the remove tutor button. |
| Expected Result | When we tap on remove tutor button the tutor should be removed. |
| Actual Result | When we click on remove tutor button the tutor was removed. |
| Conclusion | The test was successful. |

*Table 6:Test 2.5*



Teacher ID: 1111
Teacher Name: Evani
Address: Butwal
Working Type: Full
Employment Status: Employed
Working Hours: 6
Salary: 50000
Specialization: IT
Academic Qualifications: Graduate
Performance Index: 9

Teacher ID: 1111
New Salary: 70000
New Performance Index: 10

Teacher ID: 1111
Graded Score: 10
Department: IT
Years of Experience: 3

Tutor removed successfully.

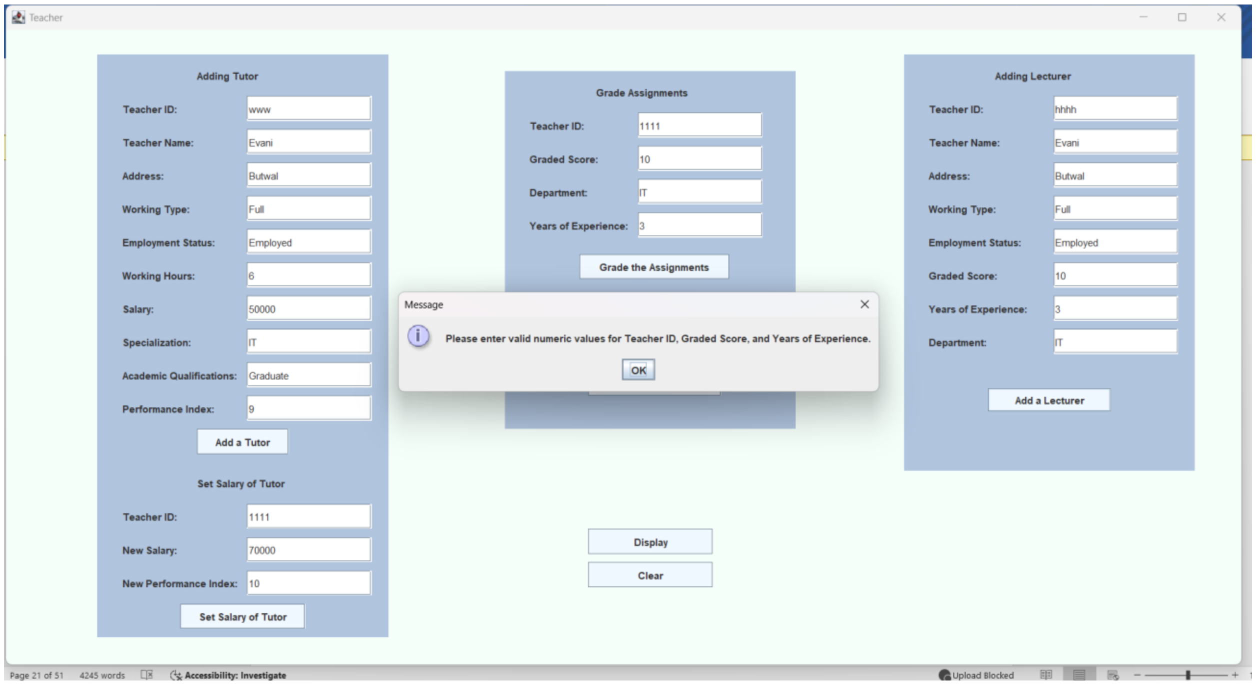
Teacher ID: 1111
Teacher Name: Evani
Address: Butwal
Working Type: Full
Employment Status: Employed
Graded Score: 10
Years of Experience: 3
Department: IT

*Figure 10:Test 2.5*

**Test 3.1** To test that appropriate dialog boxes appear when unsuitable values are entered for the Teacher ID in the lecturer part.

| | |
|---|---|
| Objective | To test that appropriate dialog boxes appear when unsuitable values are entered for the Teacher ID. |
| Action | Passing the string value for the teacher id in lecturer part. |
| Expected Result | While entering string value in the teacher id it should show an appropriate dialogue box. |
| Actual Result | While entering string value in the teacher id it should show an appropriate dialogue box. |
| Conclusion | Test was Successful. |

*Table 7:Test 3.1*



Adding Tutor

| | |
|---|---|
| Teacher ID: | www |
| Teacher Name: | Evani |
| Address: | Butwal |
| Working Type: | Full |
| Employment Status: | Employed |
| Working Hours: | 6 |
| Salary: | 50000 |
| Specialization: | IT |
| Academic Qualifications: | Graduate |
| Performance Index: | 9 |

Add a Tutor

Set Salary of Tutor

| | |
|---|---|
| Teacher ID: | 1111 |
| New Salary: | 70000 |
| New Performance Index: | 10 |

Set Salary of Tutor

Grade Assignments

| | |
|---|---|
| Teacher ID: | 1111 |
| Graded Score: | 10 |
| Department: | IT |
| Years of Experience: | 3 |

Grade the Assignments

Message

Please enter valid numeric values for Teacher ID, Graded Score, and Years of Experience.

OK

Adding Lecturer

| | |
|---|---|
| Teacher ID: | hhhh |
| Teacher Name: | Evani |
| Address: | Butwal |
| Working Type: | Full |
| Employment Status: | Employed |
| Graded Score: | 10 |
| Years of Experience: | 3 |
| Department: | IT |

Add a Lecturer

Display

Clear

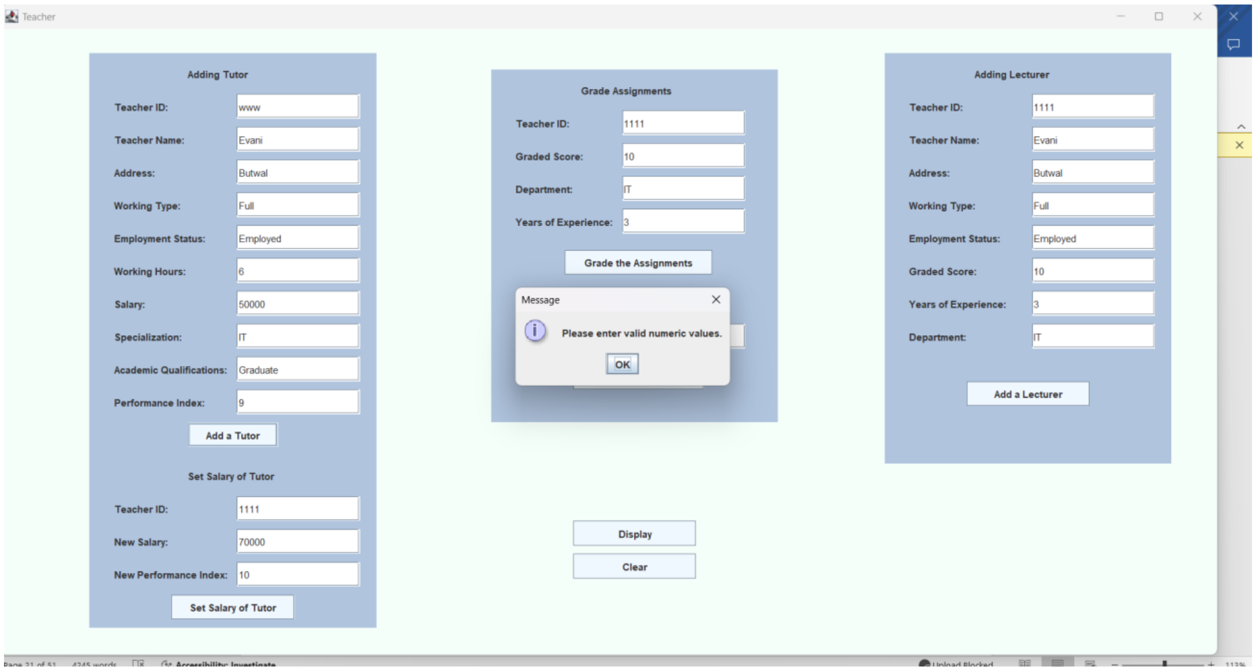
*Figure 11:Test 3.1*

**3.2** To test that appropriate dialog boxes appear when unsuitable values are entered for the Teacher ID in the tutor part.

| | |
|---|---|
| Objective | To test that appropriate dialog boxes appear when unsuitable values are entered for the Teacher ID. |
| Action | Passing the string value for the teacher id in tutor part. |
| Expected Result | While entering string value in the teacher id it should show an appropriate dialogue box. |

| | |
|---|---|
| Actual Result | While entering string value in the teacher id it should show an appropriate dialogue box. |
| Conclusion | Test was Successful. |

Table 8:Test 3.2



Adding Tutor

| | |
|---|---|
| Teacher ID: | www |
| Teacher Name: | Evani |
| Address: | Butwal |
| Working Type: | Full |
| Employment Status: | Employed |
| Working Hours: | 6 |
| Salary: | 50000 |
| Specialization: | IT |
| Academic Qualifications: | Graduate |
| Performance Index: | 9 |

Set Salary of Tutor

| | |
|---|---|
| Teacher ID: | 1111 |
| New Salary: | 70000 |
| New Performance Index: | 10 |

Grade Assignments

| | |
|---|---|
| Teacher ID: | 1111 |
| Graded Score: | 10 |
| Department: | IT |
| Years of Experience: | 3 |

Message

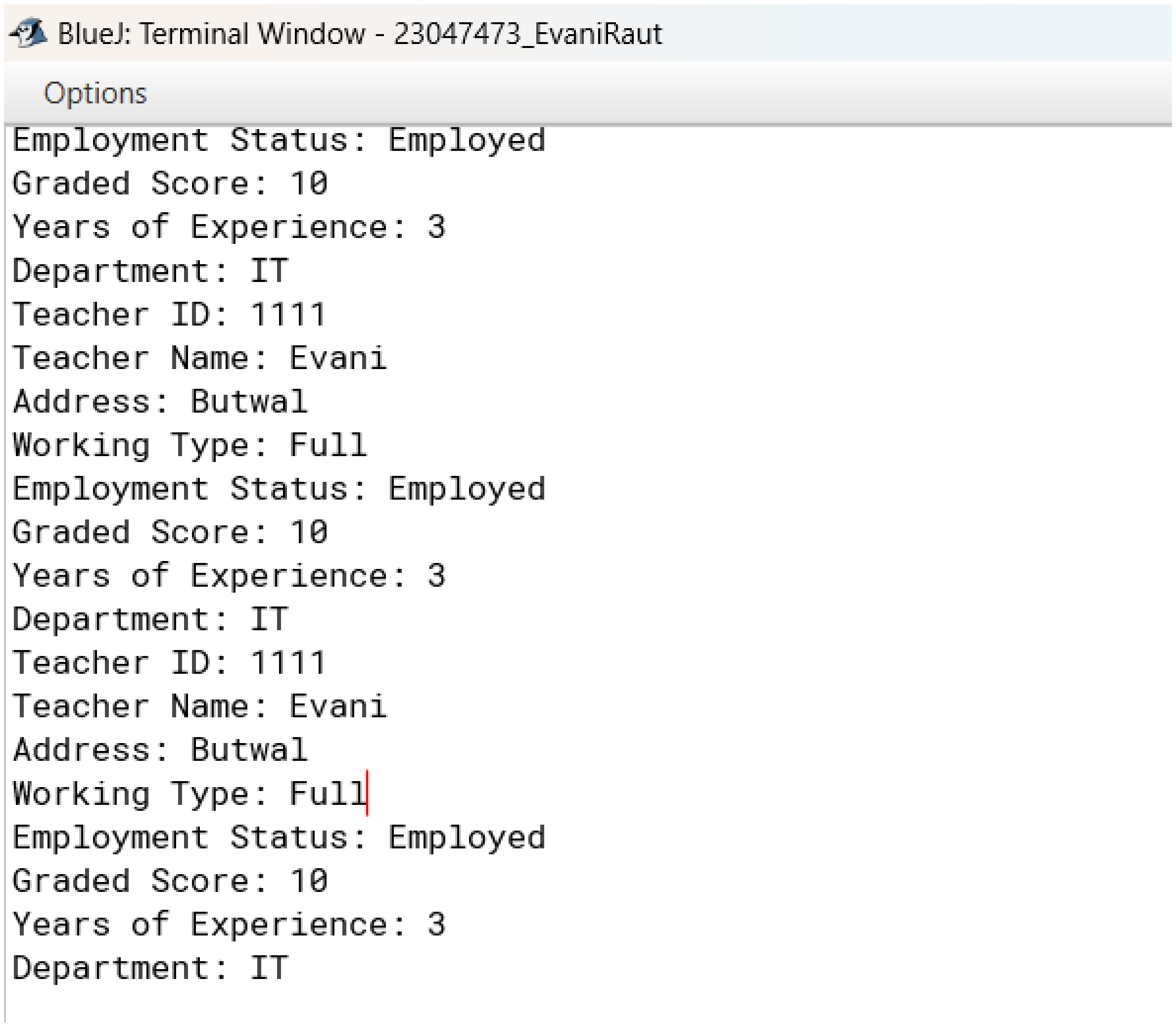
Please enter valid numeric values.

Adding Lecturer

| | |
|---|---|
| Teacher ID: | 1111 |
| Teacher Name: | Evani |
| Address: | Butwal |
| Working Type: | Full |
| Employment Status: | Employed |
| Graded Score: | 10 |
| Years of Experience: | 3 |
| Department: | IT |

Figure 12:Test 3.2

*Figure 13:Display*



BlueJ: Terminal Window - 23047473_EvaniRaut

Options

```
Employment Status: Employed
Graded Score: 10
Years of Experience: 3
Department: IT
Teacher ID: 1111
Teacher Name: Evani
Address: Butwal
Working Type: Full
Employment Status: Employed
Graded Score: 10
Years of Experience: 3
Department: IT
Teacher ID: 1111
Teacher Name: Evani
Address: Butwal
Working Type: Full
Employment Status: Employed
Graded Score: 10
Years of Experience: 3
Department: IT
```

## 6.Error Detection:

Types of error:

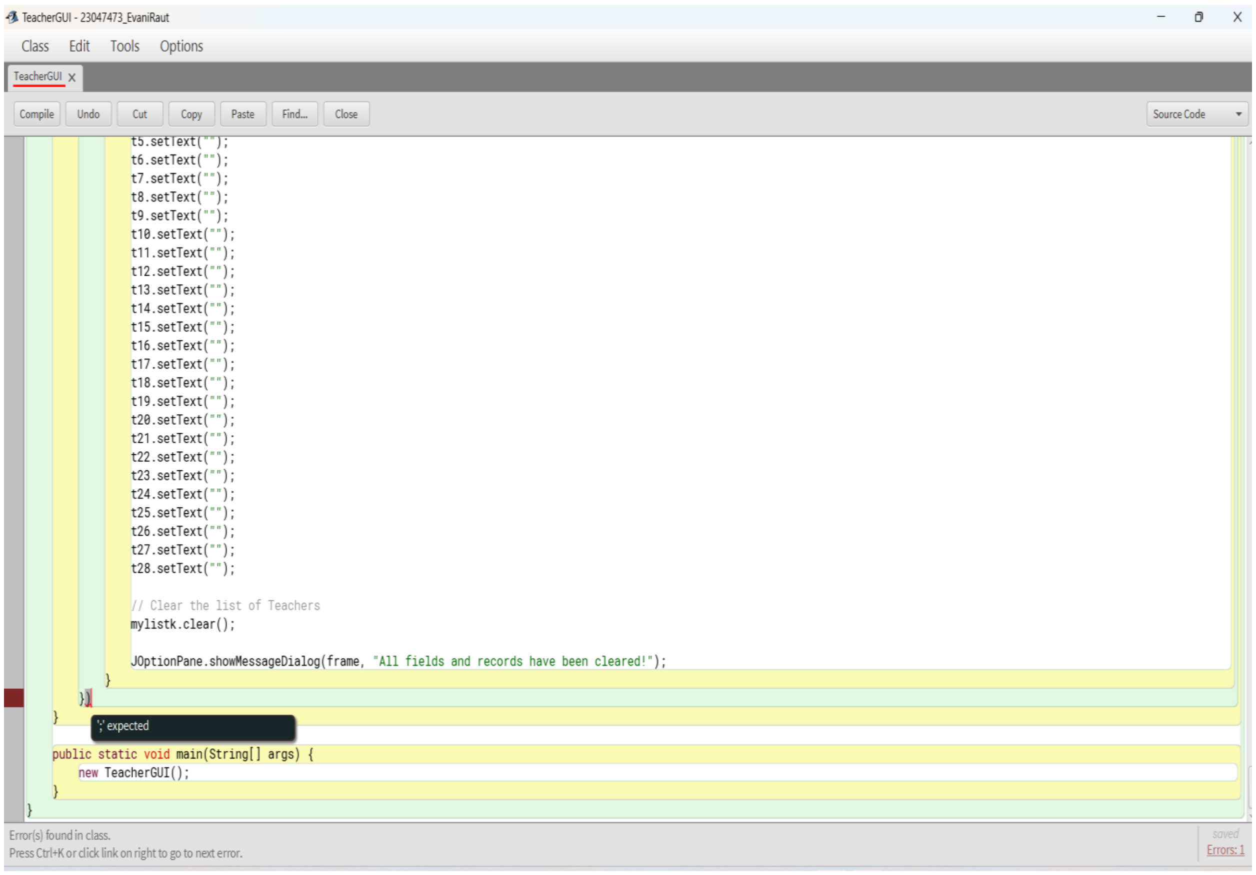
- Syntax Error

- Semantics Error

- Logical Error

## 6.1 Syntax Error

A mistake in a program that violates the rules or structure of the programming language which prevents the code from being properly understood or execute is syntax error.


TeacherGUI - 23047473_EvaniRaut
Class Edit Tools Options
TeacherGUI
Compile Undo Cut Copy Paste Find... Close
Source Code

```
            t5.setText("");
            t6.setText("");
            t7.setText("");
            t8.setText("");
            t9.setText("");
            t10.setText("");
            t11.setText("");
            t12.setText("");
            t13.setText("");
            t14.setText("");
            t15.setText("");
            t16.setText("");
            t17.setText("");
            t18.setText("");
            t19.setText("");
            t20.setText("");
            t21.setText("");
            t22.setText("");
            t23.setText("");
            t24.setText("");
            t25.setText("");
            t26.setText("");
            t27.setText("");
            t28.setText("");

            // Clear the list of Teachers
            mylistk.clear();

            JOptionPane.showMessageDialog(frame, "All fields and records have been cleared!");
        }
    })
    }
    public static void main(String[] args) {
        new TeacherGUI();
    }
}
```

';' expected

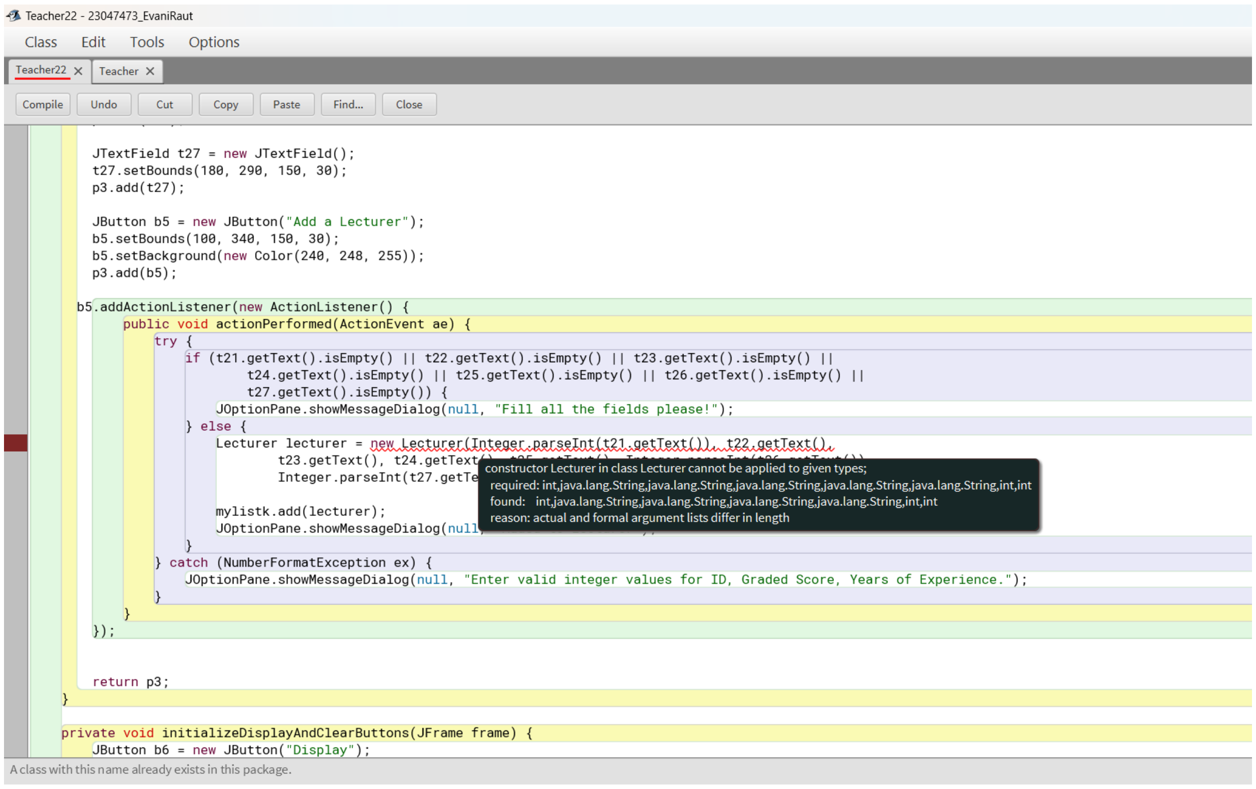
Error(s) found in class.
Press Ctrl+K or click link on right to go to next error.

saved
Errors: 1

Figure 14: Syntax Error

### 6.2 Semantics Error:

Semantics error in a program occurs when code is grammatically correct but conveys a meaning that leads to unintended and incorrect behaviour, it doesn't cause the program to crash but may result in unexpected outputs or logical flaws (dev.java, 2024).


Teacher22 - 23047473_EvaniRaut
Class Edit Tools Options
Teacher22 Teacher
Compile Undo Cut Copy Paste Find... Close

```
        JTextField t27 = new JTextField();
        t27.setBounds(180, 290, 150, 30);
        p3.add(t27);

        JButton b5 = new JButton("Add a Lecturer");
        b5.setBounds(100, 340, 150, 30);
        b5.setBackground(new Color(240, 248, 255));
        p3.add(b5);

      b5.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent ae) {
                try {
                    if (t21.getText().isEmpty() || t22.getText().isEmpty() || t23.getText().isEmpty() ||
                            t24.getText().isEmpty() || t25.getText().isEmpty() || t26.getText().isEmpty() ||
                            t27.getText().isEmpty()) {
                        JOptionPane.showMessageDialog(null, "Fill all the fields please!");
                    } else {
                        Lecturer lecturer = new Lecturer(Integer.parseInt(t21.getText()), t22.getText(),
                                t23.getText(), t24.getText
                                Integer.parseInt(t27.getTe

                        mylistk.add(lecturer);
                        JOptionPane.showMessageDialog(null
                    }
                } catch (NumberFormatException ex) {
                    JOptionPane.showMessageDialog(null, "Enter valid integer values for ID, Graded Score, Years of Experience.");
                }
            }
        });


        return p3;
    }

    private void initializeDisplayAndClearButtons(JFrame frame) {
        JButton b6 = new JButton("Display");
```

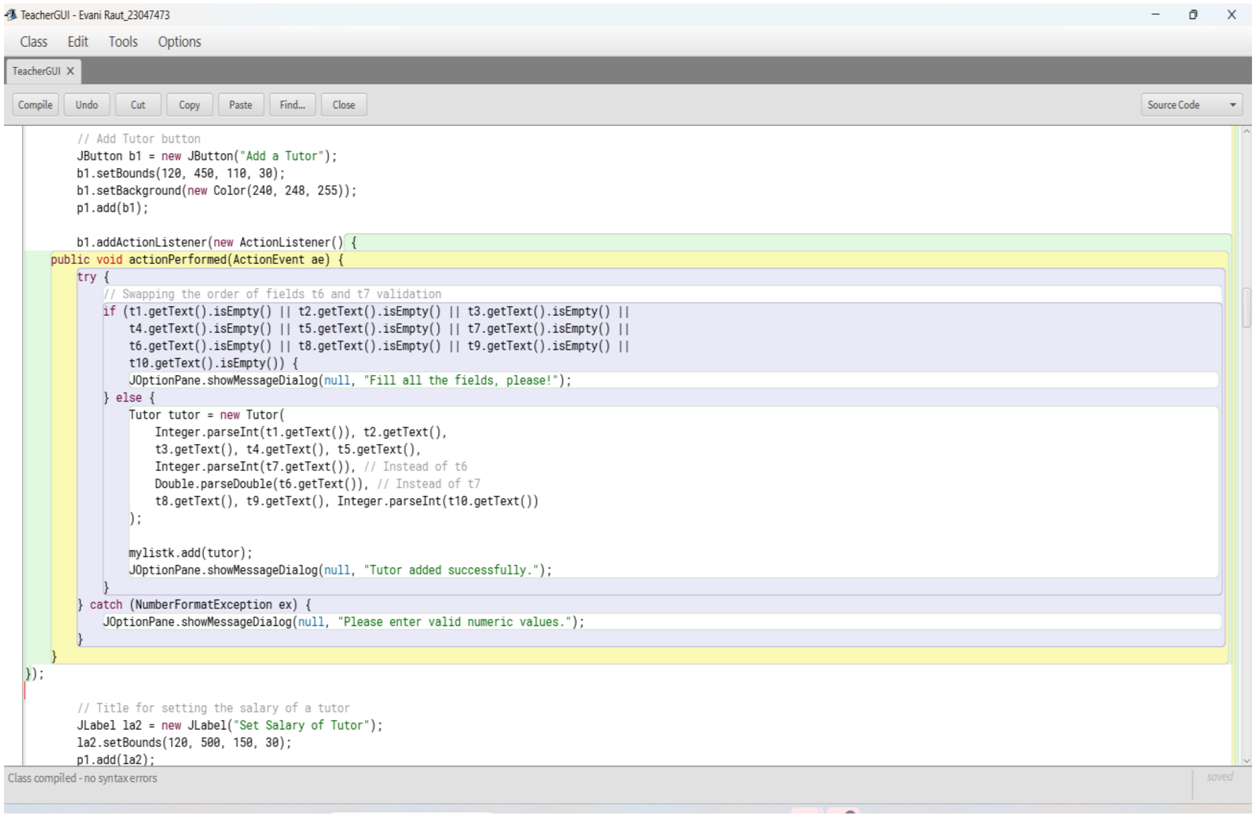
```
constructor Lecturer in class Lecturer cannot be applied to given types;
  required: int,java.lang.String,java.lang.String,java.lang.String,java.lang.String,java.lang.String,int,int
  found:    int,java.lang.String,java.lang.String,java.lang.String,java.lang.String,int,int
  reason: actual and formal argument lists differ in length
```

A class with this name already exists in this package.

*Figure 15:Semantics Error*

## 6.3 Logical Error

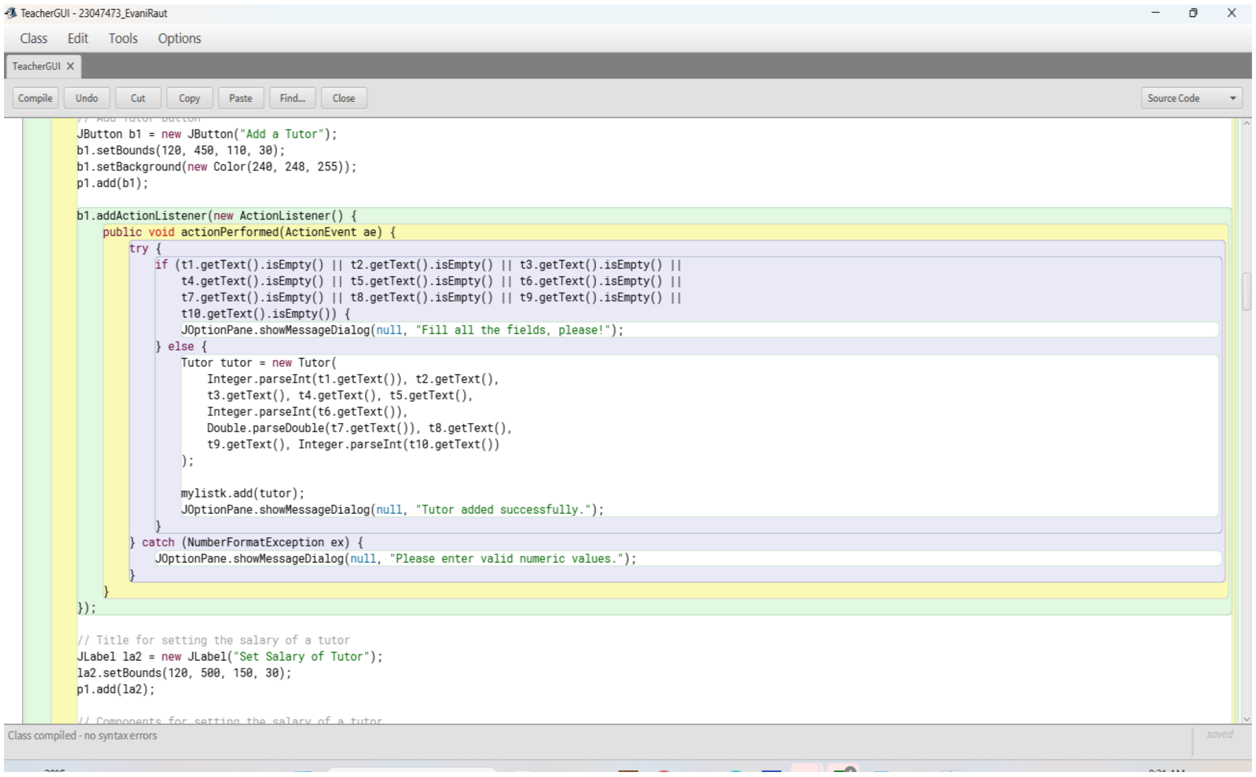
A logical error in programming is error that occurs when the code is syntactically correct but the algorithm logic or instructions is flawed which leads to incorrect and unintended results in the output (geeksforgeeks, 2024).



```
        // Add Tutor button
        JButton b1 = new JButton("Add a Tutor");
        b1.setBounds(120, 450, 110, 30);
        b1.setBackground(new Color(240, 248, 255));
        p1.add(b1);

        b1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
        try {
            // Swapping the order of fields t6 and t7 validation
            if (t1.getText().isEmpty() || t2.getText().isEmpty() || t3.getText().isEmpty() ||
                t4.getText().isEmpty() || t5.getText().isEmpty() || t7.getText().isEmpty() ||
                t6.getText().isEmpty() || t8.getText().isEmpty() || t9.getText().isEmpty() ||
                t10.getText().isEmpty()) {
                JOptionPane.showMessageDialog(null, "Fill all the fields, please!");
            } else {
                Tutor tutor = new Tutor(
                    Integer.parseInt(t1.getText()), t2.getText(),
                    t3.getText(), t4.getText(), t5.getText(),
                    Integer.parseInt(t7.getText()), // Instead of t6
                    Double.parseDouble(t6.getText()), // Instead of t7
                    t8.getText(), t9.getText(), Integer.parseInt(t10.getText())
                );

                mylistk.add(tutor);
                JOptionPane.showMessageDialog(null, "Tutor added successfully.");
            }
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(null, "Please enter valid numeric values.");
        }
    }
});

        // Title for setting the salary of a tutor
        JLabel la2 = new JLabel("Set Salary of Tutor");
        la2.setBounds(120, 500, 150, 30);
        p1.add(la2);
```

Class compiled - no syntax errors

*Figure 16:Logical Error*



```
JButton b1 = new JButton("Add a Tutor");
b1.setBounds(120, 450, 110, 30);
b1.setBackground(new Color(240, 248, 255));
p1.add(b1);

b1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
        try {
            if (t1.getText().isEmpty() || t2.getText().isEmpty() || t3.getText().isEmpty() ||
                t4.getText().isEmpty() || t5.getText().isEmpty() || t6.getText().isEmpty() ||
                t7.getText().isEmpty() || t8.getText().isEmpty() || t9.getText().isEmpty() ||
                t10.getText().isEmpty()) {
                JOptionPane.showMessageDialog(null, "Fill all the fields, please!");
            } else {
                Tutor tutor = new Tutor(
                    Integer.parseInt(t1.getText()), t2.getText(),
                    t3.getText(), t4.getText(), t5.getText(),
                    Integer.parseInt(t6.getText()),
                    Double.parseDouble(t7.getText()), t8.getText(),
                    t9.getText(), Integer.parseInt(t10.getText())
                );

                mylistk.add(tutor);
                JOptionPane.showMessageDialog(null, "Tutor added successfully.");
            }
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(null, "Please enter valid numeric values.");
        }
    }
});

// Title for setting the salary of a tutor
JLabel la2 = new JLabel("Set Salary of Tutor");
la2.setBounds(120, 500, 150, 30);
p1.add(la2);
```

Class compiled - no syntax errors

*Figure 17:Logical Error Correction*

## 7.Conclusion:

In summary, the development of the Java GUI application for managing teacher data has been a comprehensive and instructive journey. Through the creation of this application, we were able to explore and apply various object-oriented programming concepts in Java, such as polymorphism, encapsulation, and inheritance. Additionally, we gained hands-on experience with designing user-friendly graphical interfaces that allow users to interact seamlessly with the data.

The project's primary aim was successfully met, as the application efficiently manages Lecturer and Tutor objects within an array list. Users can add new Lecturers and Tutors, grade assignments, set tutor salaries, and remove tutors with ease. Furthermore, the program provides valuable feedback to the user through clear error messages and validation checks, ensuring robust error handling and a smooth user experience.

The project's functionality was thoroughly tested to confirm that all operations work as intended. Various input scenarios, including edge cases and invalid inputs, were used to assess the application's robustness. The program demonstrated reliable performance in handling these scenarios, with appropriate responses in terms of warnings and error messages.

Throughout the project, we encountered and resolved challenges related to GUI design and implementation. Careful planning and consistent use of best practices in coding helped ensure the success of the application. Additionally, the focus on code quality and readability contributed to a well-structured and maintainable application.

Overall, this project offered an opportunity to deepen our understanding of Java programming and GUI development while producing a practical tool for managing teacher data. The knowledge and skills gained from this experience will prove valuable in future projects and tasks involving software development and user interface design.

# 8.References

dev.java. (2024). Retrieved from dev.java: https://dev.java/learn/

geeksforgeeks. (2024). Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/java/

w3schools. (2024). Retrieved from w3schools: https://www.w3schools.com/java/java_intro.asp

# 9.Appendix:

## TeacherGUI

```
import javax.swing.*;

import java.awt.Color;
import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

import java.util.Iterator;


class TeacherGUI {

    ArrayList<Teacher> mylistk = new ArrayList<>(); // Define the ArrayList


    // Declare JTextField variables globally

    private JTextField t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12, t13, t14, t15, t16, t17, t18,
t19, t20, t21, t22, t23, t24, t25, t26, t27, t28;


    public TeacherGUI() {

        JFrame frame = new JFrame("Teacher");

        frame.setVisible(true);

        frame.setSize(1500, 800);

        frame.setLayout(null);

        frame.getContentPane().setBackground(new Color(245, 255, 250));


        // Initialize and add Panel 1 (Adding Tutor)

        JPanel p1 = initializeAddingTutorPanel();

        frame.add(p1);
        // Initialize and add Panel 2 (Grade Assignments)

        JPanel p2 = initializeGradeAssignmentsPanel();

        frame.add(p2);


        // Initialize and add Panel 3 (Adding Lecturer)

        JPanel p3 = initializeAddingLecturerPanel();

        frame.add(p3);


        // Initialize and add display and clear buttons

        initializeDisplayAndClearButtons(frame);

    }


    // Initialize Panel 1 (Adding Tutor) and JTextField variables

    private JPanel initializeAddingTutorPanel() {

        JPanel p1 = new JPanel();

        p1.setBounds(110, 30, 350, 700);

        p1.setBackground(new Color(176, 196, 222));

        p1.setLayout(null);


        JLabel la1 = new JLabel("Adding Tutor");

        la1.setBounds(120, 10, 100, 30);

        p1.add(la1);
```

```
// Initialize JTextField variables for adding tutor

JLabel l1 = new JLabel("Teacher ID:");

l1.setBounds(30, 50, 100, 30);

p1.add(l1);


t1 = new JTextField();

t1.setBounds(180, 50, 150, 30);

p1.add(t1);


JLabel l2 = new JLabel("Teacher Name:");

l2.setBounds(30, 90, 100, 30);

p1.add(l2);


t2 = new JTextField();

t2.setBounds(180, 90, 150, 30);

p1.add(t2);


JLabel l3 = new JLabel("Address:");

l3.setBounds(30, 130, 100, 30);

p1.add(l3);


t3 = new JTextField();

t3.setBounds(180, 130, 150, 30);
```

```
        p1.add(t3);

        JLabel l4 = new JLabel("Working Type:");
        l4.setBounds(30, 170, 100, 30);
        p1.add(l4);

        t4 = new JTextField();
        t4.setBounds(180, 170, 150, 30);
        p1.add(t4);

        JLabel l5 = new JLabel("Employment Status:");
        l5.setBounds(30, 210, 120, 30);
        p1.add(l5);

        t5 = new JTextField();
        t5.setBounds(180, 210, 150, 30);
        p1.add(t5);

        JLabel l6 = new JLabel("Working Hours:");
        l6.setBounds(30, 250, 100, 30);
        p1.add(l6);

        t6 = new JTextField();
```

```
t6.setBounds(180, 250, 150, 30);

p1.add(t6);


JLabel l7 = new JLabel("Salary:");

l7.setBounds(30, 290, 120, 30);

p1.add(l7);


t7 = new JTextField();

t7.setBounds(180, 290, 150, 30);

p1.add(t7);


JLabel l8 = new JLabel("Specialization:");

l8.setBounds(30, 330, 120, 30);

p1.add(l8);


t8 = new JTextField();

t8.setBounds(180, 330, 150, 30);

p1.add(t8);


JLabel l9 = new JLabel("Academic Qualifications:");

l9.setBounds(30, 370, 150, 30);

p1.add(l9);
```

```
t9 = new JTextField();

t9.setBounds(180, 370, 150, 30);

p1.add(t9);


JLabel l10 = new JLabel("Performance Index:");

l10.setBounds(30, 410, 150, 30);

p1.add(l10);


t10 = new JTextField();

t10.setBounds(180, 410, 150, 30);

p1.add(t10);


// Add Tutor button

JButton b1 = new JButton("Add a Tutor");

b1.setBounds(120, 450, 110, 30);

b1.setBackground(new Color(240, 248, 255));

p1.add(b1);


b1.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent ae) {

        try {

            if (t1.getText().isEmpty() || t2.getText().isEmpty() || t3.getText().isEmpty() ||

                t4.getText().isEmpty() || t5.getText().isEmpty() || t6.getText().isEmpty() ||
                    t7.getText().isEmpty() || t8.getText().isEmpty() || t9.getText().isEmpty() ||
                    t10.getText().isEmpty()) {
                    JOptionPane.showMessageDialog(null, "Fill all the fields, please!");
                } else {
                    Tutor tutor = new Tutor(
                        Integer.parseInt(t1.getText()), t2.getText(),
                        t3.getText(), t4.getText(), t5.getText(),
                        Integer.parseInt(t6.getText()),
                        Double.parseDouble(t7.getText()), t8.getText(),
                        t9.getText(), Integer.parseInt(t10.getText())
                    );

                    mylistk.add(tutor);
                    JOptionPane.showMessageDialog(null, "Tutor added successfully.");
                }
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(null, "Please enter valid numeric
values.");
            }
        }
    });

    // Title for setting the salary of a tutor
```

```
JLabel la2 = new JLabel("Set Salary of Tutor");

la2.setBounds(120, 500, 150, 30);

p1.add(la2);


// Components for setting the salary of a tutor

JLabel l11 = new JLabel("Teacher ID:");

l11.setBounds(30, 540, 120, 30);

p1.add(l11);


t11 = new JTextField();

t11.setBounds(180, 540, 150, 30);

p1.add(t11);


JLabel l12 = new JLabel("New Salary:");

l12.setBounds(30, 580, 120, 30);

p1.add(l12);


t12 = new JTextField();

t12.setBounds(180, 580, 150, 30);

p1.add(t12);


JLabel l13 = new JLabel("New Performance Index:");

l13.setBounds(30, 620, 150, 30);
```

```
p1.add(l13);


t13 = new JTextField();

t13.setBounds(180, 620, 150, 30);

p1.add(t13);


JButton b2 = new JButton("Set Salary of Tutor");

b2.setBounds(100, 660, 150, 30);

b2.setBackground(new Color(240, 248, 255));

p1.add(b2);


// Add action listener to button b2

b2.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent ae) {

        try {

            int teacherId = Integer.parseInt(t11.getText());

            double newSalary = Double.parseDouble(t12.getText());

            int newPerformanceIndex = Integer.parseInt(t13.getText());


            boolean tutorFound = false;

            for (Teacher teacher : mylistk) {

                if (teacher instanceof Tutor && teacher.getTeacherID() == teacherId) {

                    Tutor tutor = (Tutor) teacher;
                    String result = tutor.setSalary(newSalary, newPerformanceIndex);
                    JOptionPane.showMessageDialog(null, "Result: " + result);
                    tutorFound = true;
                    break;
                }
            }

            if (!tutorFound) {
                JOptionPane.showMessageDialog(null, "Tutor with ID " + teacherId + " not found!");
            }
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(null, "Please enter valid numeric values.");
        }
    }
  });

  return p1;
}

// Initialize and return Panel 2 (Grade Assignments)
private JPanel initializeGradeAssignmentsPanel() {
```

```
JPanel p2 = new JPanel();

p2.setBounds(600, 50, 350, 430);

p2.setBackground(new Color(176, 196, 222));

p2.setLayout(null);


JLabel la14 = new JLabel("Grade Assignments");

la14.setBounds(110, 10, 200, 30);

p2.add(la14);


JLabel l15 = new JLabel("Teacher ID:");

l15.setBounds(30, 50, 150, 30);

p2.add(l15);


t15 = new JTextField();

t15.setBounds(160, 50, 150, 30);

p2.add(t15);


JLabel l16 = new JLabel("Graded Score:");

l16.setBounds(30, 90, 150, 30);

p2.add(l16);


t16 = new JTextField();

t16.setBounds(160, 90, 150, 30);
```

```
p2.add(t16);


JLabel l17 = new JLabel("Department:");

l17.setBounds(30, 130, 150, 30);

p2.add(l17);


t17 = new JTextField();

t17.setBounds(160, 130, 150, 30);

p2.add(t17);


JLabel l18 = new JLabel("Years of Experience:");

l18.setBounds(30, 170, 170, 30);

p2.add(l18);


t18 = new JTextField();

t18.setBounds(160, 170, 150, 30);

p2.add(t18);


JButton b3 = new JButton("Grade the Assignments");

b3.setBounds(90, 220, 180, 30);

b3.setBackground(new Color(240, 248, 255));

p2.add(b3);
```

```
        b3.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent ae) {
                try {
                    int teacherId = Integer.parseInt(t15.getText());
                    int gradedScore = Integer.parseInt(t16.getText());
                    String department = t17.getText();
                    int yearsOfExperience = Integer.parseInt(t18.getText());

                    for (Teacher teacher : mylistk) {
                        if (teacher instanceof Lecturer && teacher.getTeacherID() == teacherId) {
                            Lecturer lecturer = (Lecturer) teacher;
                            String grade = lecturer.gradeAssignment(gradedScore, department,
yearsOfExperience);
                            JOptionPane.showMessageDialog(null, "Graded assignment result: " +
grade);
                            return;
                        }
                    }

                    JOptionPane.showMessageDialog(null, "Lecturer with ID " + teacherId + "
not found!");

                } catch (NumberFormatException ex) {
```

```
                JOptionPane.showMessageDialog(null, "Please enter valid numeric values.");
            }
        }
    });

    JLabel l19 = new JLabel("Remove Tutor");
    l19.setBounds(130, 270, 150, 30);
    p2.add(l19);

    JLabel l20 = new JLabel("Teacher ID:");
    l20.setBounds(30, 310, 150, 30);
    p2.add(l20);

    t20 = new JTextField();
    t20.setBounds(160, 310, 150, 30);
    p2.add(t20);

    JButton b4 = new JButton("Remove the tutor");
    b4.setBounds(100, 360, 160, 30);
    b4.setBackground(new Color(240, 248, 255));
    p2.add(b4);
    b4.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent ae) {
            try {
                int teacherId = Integer.parseInt(t20.getText());

                boolean removed = false;
                for (Iterator<Teacher> iterator = mylistk.iterator(); iterator.hasNext();) {
                    Teacher teacher = iterator.next();
                    if (teacher instanceof Tutor && teacher.getTeacherID() == teacherId) {
                        iterator.remove();
                        removed = true;
                        JOptionPane.showMessageDialog(null, "Tutor removed
successfully.");
                        break;
                    }
                }

                if (!removed) {
                    JOptionPane.showMessageDialog(null, "Tutor with ID " + teacherId + " not
found!");
                }
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(null, "Only integer values accepted for
Teacher ID.");
            }
        }
    });

    return p2;
}

// Initialize Panel 3 (Adding Lecturer)
private JPanel initializeAddingLecturerPanel() {
    JPanel p3 = new JPanel();
    p3.setBounds(1080, 30, 350, 500);
    p3.setBackground(new Color(176, 196, 222));
    p3.setLayout(null);

    JLabel la3 = new JLabel("Adding Lecturer");
    la3.setBounds(110, 10, 100, 30);
    p3.add(la3);

    JLabel l21 = new JLabel("Teacher ID:");
    l21.setBounds(30, 50, 100, 30);
    p3.add(l21);
```

```
t21 = new JTextField();

t21.setBounds(180, 50, 150, 30);

p3.add(t21);


JLabel l22 = new JLabel("Teacher Name:");

l22.setBounds(30, 90, 100, 30);

p3.add(l22);


t22 = new JTextField();

t22.setBounds(180, 90, 150, 30);

p3.add(t22);


// Add address field

JLabel l23 = new JLabel("Address:");

l23.setBounds(30, 130, 100, 30);

p3.add(l23);


t23 = new JTextField();

t23.setBounds(180, 130, 150, 30);

p3.add(t23);


// Add working type field
```

```
JLabel l24 = new JLabel("Working Type:");
l24.setBounds(30, 170, 100, 30);
p3.add(l24);

t24 = new JTextField();
t24.setBounds(180, 170, 150, 30);
p3.add(t24);

// Add employment status field
JLabel l25 = new JLabel("Employment Status:");
l25.setBounds(30, 210, 120, 30);
p3.add(l25);

t25 = new JTextField();
t25.setBounds(180, 210, 150, 30);
p3.add(t25);

JLabel l26 = new JLabel("Graded Score:");
l26.setBounds(30, 250, 100, 30);
p3.add(l26);

t26 = new JTextField();
t26.setBounds(180, 250, 150, 30);
```

```
p3.add(t26);


JLabel l27 = new JLabel("Years of Experience:");

l27.setBounds(30, 290, 150, 30);

p3.add(l27);


t27 = new JTextField();

t27.setBounds(180, 290, 150, 30);

p3.add(t27);


JLabel l28 = new JLabel("Department:");

l28.setBounds(30, 330, 150, 30);

p3.add(l28);


t28 = new JTextField();

t28.setBounds(180, 330, 150, 30);

p3.add(t28);


JButton b5 = new JButton("Add a Lecturer");

b5.setBounds(100, 400, 150, 30);

b5.setBackground(new Color(240, 248, 255));

p3.add(b5);
```

```
    b5.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent ae) {
            try {
                if (t21.getText().isEmpty() || t22.getText().isEmpty() || t23.getText().isEmpty() ||
                    t24.getText().isEmpty() || t25.getText().isEmpty() || t26.getText().isEmpty() ||
                    t27.getText().isEmpty() || t28.getText().isEmpty()) {
                    JOptionPane.showMessageDialog(null, "Please fill all the fields.");
                } else {
                    Lecturer lecturer = new Lecturer(
                        Integer.parseInt(t21.getText()),
                        t22.getText(),
                        t23.getText(),
                        t24.getText(),
                        t25.getText(),
                        Integer.parseInt(t26.getText()),
                        Integer.parseInt(t27.getText()),
                        t28.getText()
                    );

                    mylistk.add(lecturer);
                    JOptionPane.showMessageDialog(null, "Lecturer added successfully.");
                }
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(null, "Please enter valid numeric values for Teacher ID, Graded Score, and Years of Experience.");
            }
        }
    });

    return p3;
}

// Initialize display and clear buttons
private void initializeDisplayAndClearButtons(JFrame frame) {
    JButton b6 = new JButton("Display");
    b6.setBounds(700, 600, 150, 30);
    b6.setBackground(new Color(240, 248, 255));
    frame.add(b6);

    JButton b7 = new JButton("Clear");
    b7.setBounds(700, 640, 150, 30);
    b7.setBackground(new Color(240, 248, 255));
    frame.add(b7);
```

```
b6.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
        // Display the list of Teachers
        for (Teacher teacher : mylistk) {
            if (teacher instanceof Lecturer) {
                teacher.display();
            } else if (teacher instanceof Tutor) {
                teacher.display();
            }
        }
    }
});

b7.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
        // Clear all JTextFields and list of Teachers
        t1.setText("");
        t2.setText("");
        t3.setText("");
        t4.setText("");
        t5.setText("");
        t6.setText("");
        t7.setText("");
```

```
t8.setText("");

t9.setText("");

t10.setText("");

t11.setText("");

t12.setText("");

t13.setText("");

t14.setText("");

t15.setText("");

t16.setText("");

t17.setText("");

t18.setText("");

t19.setText("");

t20.setText("");

t21.setText("");

t22.setText("");

t23.setText("");

t24.setText("");

t25.setText("");

t26.setText("");

t27.setText("");

t28.setText("");


// Clear the list of Teachers
```

```
                mylistk.clear();


                JOptionPane.showMessageDialog(frame, "All fields and records have been cleared!");
            }
        });
    }


    public static void main(String[] args) {
        new TeacherGUI();
    }
}
```

**Teacher**

```
//Creating Teacher class
public class Teacher{
    //Attributes
    private int teacherID;
    private String teacherName;
    private String address;
    private String workingType;
    private String employmentStatus;
    private int workingHours;
   // constructor with parameters
```

```
  public Teacher(int teacherID, String teacherName, String address, String
workingType,
  String employmentStatus){
     // instance variables
     this.teacherID = teacherID;
     this.teacherName = teacherName;
     this.address = address;
     this.workingType = workingType;
     this.employmentStatus = employmentStatus;


  }


 //Getter Method (Accessor Method) for attributes
  public int getTeacherID(){
     return teacherID;
  }
  public String getTeacherName(){
     return teacherName;
  }
  public String getAddress(){
     return address;
  }
  public String getWorkingType(){
     return workingType;
  }
  public String getEmploymentStatus(){
     return employmentStatus;
  }
  public int getWorkingHours() {
     return workingHours;
  }

  //Setter Method
  public void setWorkingHours(int workingHours) {
     this.workingHours = workingHours;
  }

   // method to display the details of the teacher
  public void display() {
     System.out.println("Teacher ID: " + teacherID);
     System.out.println("Teacher Name: " + teacherName);
     System.out.println("Address: " + address);
     System.out.println("Working Type: " + workingType);
     System.out.println("Employment Status: " + employmentStatus);
     if (workingHours > 0) {
         System.out.println("Working Hours: " + workingHours);
        } else {
            System.out.println("Working Hours: Not assigned");
}
}
}
```

**Lecturer**

```
// Creating child class of teacher class
public class Lecturer extends Teacher
{

    private String department;
    private int yearsOfExperience;
    private int gradedScore;
    private boolean hasGraded;


        public Lecturer(int teacherId, String teacherName, String address, String workingType, String employmentStatus, String department, int yearsOfExperience, int gradedScore) {
        super(teacherId, teacherName, address, workingType, employmentStatus);
        this.department = department;
        this.yearsOfExperience = yearsOfExperience;
        this.gradedScore = gradedScore;
        this.hasGraded = false;
    }

    //Getter method(acessor methods) for the attributes

    public String getDepartment(){
        return department;
    }

    public int getYearsOfExperience(){
        return yearsOfExperience;
    }

    public int getGradedScore(){
        return gradedScore;
    }

    public boolean getHasGraded(){
        return hasGraded;
    }

    //setter method(Mutator method) for graded score
    public void setGradedScore(int gradedScore){
        this.gradedScore = gradedScore;
    }


        // Method to grade assignments
          public String gradeAssignment(int score, String department, int yearsOfExperience) {
        if (!getHasGraded()) {
          if (this.department.equals(department) && this.yearsOfExperience >= yearsOfExperience) {
            gradedScore = score;
            if (score >= 90) {
              hasGraded = true;
              return "Result: A";


            } else if (score >= 80) {
              hasGraded = true;
              return "Result: B";
            } else if (score >= 70) {
              hasGraded = true;
              return "Result: C";
            } else if (score >= 60) {
              hasGraded = true;
              return "Result: D";
```

```
                } else {
                    return "Result: E";
                }
            } else {
                return "Unable to grade assignments at this time.";
            }
        } else {
            return "Assignment already graded.";
        }
    }

    // method to display the details of the lecturer
    public void display() {
        super.display();
        System.out.println("Department: " + department);
        System.out.println("Years of Experience: " + yearsOfExperience);
        if (getHasGraded()) {
            System.out.println("Graded Score: " + gradedScore);
        } else {
            System.out.println("Graded Score: Not available");
        }
    }
}
```

**Tutor**

```
//Class Tutor is a child of Teacher class
//creating child class of teacher class
public class Tutor extends Teacher{
  private int workingHours;
  private double salary;
  private String specialization;
  private String academicQualifications;
  private int performanceIndex;
  private boolean isCertified;
   //using constructor for Tutor
   public Tutor(int teacherId, String teacherName, String address, String workingType, String employmentStatus,
   int workingHours, double salary, String specialization, String academicQualifications, int performanceIndex) {
      super(teacherId, teacherName, address, workingType, employmentStatus);
      this.workingHours = workingHours;
      this.salary = salary;
      this.specialization = specialization;
      this.academicQualifications = academicQualifications;
      this.performanceIndex = performanceIndex;
      this.isCertified = false;
      super.setWorkingHours(workingHours);// set working hours for Tutor object
```

```
}
//Using Getter Method (accessor method)
public int getWorkingHours() {
   return workingHours;
}
public double getSalary(){
   return salary;
}
public String getSpecialization(){
   return specialization;
}
public String getAcademicQualifications(){
   return academicQualifications;
}
public int getPerformanceIndex(){
   return performanceIndex;
}
public boolean getIsCertified(){
   return isCertified;
}
//Using Setter Method (mutator method)
    public String setSalary(double newSalary, int newPerformanceIndex) {
   if (!isCertified && newPerformanceIndex > 5 && workingHours > 20) {
        double appraisalPercentage;
        if (newPerformanceIndex >= 5 && newPerformanceIndex <= 7) {
            appraisalPercentage = 5;
        } else if (newPerformanceIndex >= 8 && newPerformanceIndex <= 9) {
            appraisalPercentage = 10;
        } else { // newPerformanceIndex is 10
            appraisalPercentage = 20;
        }
        double appraisal = (appraisalPercentage / 100) * salary;
        salary += appraisal + newSalary - salary;
        performanceIndex = newPerformanceIndex;
        isCertified = true;
        return "Salary approved for " + getTeacherName() + ". New salary: " + salary;
    } else {
        return "Salary cannot be approved at this time for " + getTeacherName();
    }
}


  public void removeTutor() {
    if (!isCertified) {
        salary = 0;
        specialization = "";
```

```
            academicQualifications = "";

            performanceIndex = 0;

            isCertified = false;

            System.out.println("Tutor removed successfully.");

        } else {

            System.out.println("Cannot remove certified tutor.");

        }

        }

        //Display method

      @Override

      public void display() {

        super.display();

        System.out.println("Specialization: " + specialization);

        System.out.println("Academic Qualifications: " + academicQualifications);

        System.out.println("Performance Index: " + performanceIndex);

        System.out.println("Salary: " + salary);

}

}
```