

**CS4001NI Programming**

**30% Individual Coursework**

**2023-24 Autumn**

**Student Name: Sandhya Paudel**

**London Met ID:** 23047608

**College ID:**  NP01NT4A230255

**Group: N8**

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

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

*I confirm that I understand my coursework needs to be submitted online via MySecondTeacher under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.*

Contents

[ACKNOWLEDGEMENT 5](#_Toc19816)

[ABSTRACT 6](#_Toc19817)

[1. INTRODUCTION 7](#_Toc19818)

[1.1 About the coursework 7](#_Toc19819)

[2. TOOLS USED 7](#_Toc19820)

[2.1. Bluej 7](#_Toc19821)

[2.2. MS Word 8](#_Toc19822)

[2.3 Draw.io 8](#_Toc19823)

[3. CLASS DIAGRAM 9](#_Toc19824)

[3.1. Teacher Class Diagram 10](#_Toc19825)

[3.2. Lecturer Class Diagram 10](#_Toc19826)

[3.3. Tutor Class Diagram 11](#_Toc19827)

[3.4. Combined Class Diagram 13](#_Toc19828)

[PSEUDOCODE 14](#_Toc19829)

[METHOD DESCRIPTION OF WHAT EACH BUTTONS DOES 18](#_Toc19830)

Figure 1: BLUEJ

.............................................................................................................. 7

Figure 2: MS WORD ....................................................................................................... 8

Figure 3: DRAW.IO ......................................................................................................... 8

Figure 4: CLASS DIAGRAM IN BLUE J .......................................................................... 9

Figure 5: Class diagram of teacher. .............................................................................. 10

Figure 6: Class Diagram of Lecturer.............................................................................. 11

Figure 7: Class diagram of Tutor. .................................................................................. 12

Figure 8: class diagram of TeacherGUI. ........................................................................ 13

Figure 9: COMBINED CLASS DIAGRAM. .................................................................... 13

Figure 10: Screenshot of lecturer button message dialog. ............................................ 23

Figure 11: screenshot of add tutor message dialog. ..................................................... 24

Figure 12: Screenshot of grade assignments message dialog. ..................................... 25

Figure 13: Screenshot of message dialog. of set salary. ............................................... 25

Figure 14: Screenshot of dialog box of remove tutor. .................................................... 26

Figure 15: screenshot of an appropriate message dialog box. ...................................... 28

Figure 16: Syntax error detection. ................................................................................. 29

Figure 17: Syntax error correction. ................................................................................ 30

Figure 18: semantic error detection. .............................................................................. 31

Figure 19: Semantic error correction. ............................................................................ 31

Figure 20: logical error detection. .................................................................................. 32

Figure 21: Logical Error Correction ............................................................................... 32

# ACKNOWLEDGEMENT

I would like to express my sincere gratitude to our excellent lecturer, Mr. Ujjwal Subedi, for his invaluable guidance and support throughout the course. His in-depth knowledge of Java programming and his ability to explain complex concepts in a clear and concise manner have been instrumental in enhancing our understanding of the subject matter.

I am also deeply grateful to our tutor, Ms. Aastha Sharma, for her unwavering dedication and patience. Her insightful feedback and helpful observation have played a pivotal role in shaping our problem-solving abilities and refining our coding skills.

Without the combined efforts of Mr. Subedi and Ms. Sharma, this coursework would not have been possible. Their commitment to fostering an engaging and supportive learning environment has been truly inspiring, and for that, I am forever thankful.

# ABSTRACT

The presented coursework is a Java Swing application called "Teacher Management System" that provides a user interface for managing details of lecturers and tutors in an educational institution. The application consists of a tabbed interface with two panels, one for lecturers and another for tutors.

The Lecturer panel allows users to add new lecturers by entering their ID, name, address, working type (temporary or permanent), employment status (full-time or parttime), working hours, department, years of experience, and graded score. Users can also grade assignments for lecturers by providing the lecturer's ID, graded score, department, and years of experience. The panel includes buttons for adding lecturers, grading assignments, clearing the input fields, and displaying the details of the added lecturers.

The Tutor panel facilitates the management of tutor details, including ID, name, address, working type, employment status, working hours, salary, specialization, academic qualifications, and performance index. Users can add new tutors, set or update their salary based on their performance index, remove existing tutors, clear the input fields, and display the details of the added tutors.

The application utilizes Java Swing components such as JFrame, JTabbedPane, JPanel, JLabel, JTextField, JComboBox, JRadioButton, JButton, and JOptionPane to create the user interface and handle user interactions. The data of lecturers and tutors is stored in an ArrayList, and the application provides methods for adding, modifying, and removing lecturers and tutors from the list.

Overall, the "Teacher Management System" coursework demonstrates the implementation of a graphical user interface using Java Swing, event handling, and object-oriented programming principles to manage and maintain the details of lecturers and tutors in an educational environment.

# 1. INTRODUCTION

## 1.1 About the coursework

This report outlines the development of a graphical user interface (GUI) application for managing teacher records using an ArrayList data structure. The project is the second coursework assignment for the Programming module.

The main objective is to create a class with a main method that implements a GUI for storing, viewing, modifying, and deleting teacher information in an ArrayList. The GUI will provide an intuitive interface for performing these operations.

The application will be tested using the command prompt, and a report will be prepared documenting the design, implementation, and testing processes involved.

The successful completion of this assignment will demonstrate proficiency in objectoriented programming, GUI development using Java's Swing or JavaFX libraries, and the practical application of data structures like ArrayLists. It will also showcase the ability to design and implement user-friendly graphical interfaces for managing data.

# 2. TOOLS USED

## 2.1. Bluej



*Figure 1: BLUEJ*

Blue is an integrated development environment (IDE) specifically designed for Java programming. It provides a user-friendly interface, visual tools, and features like a code editor, debugger, and project management. BlueJ is particularly popular for educational purposes, offering a simplified environment that helps beginners learn Java concepts with ease.

## 2.2. MS Word



*Figure 2: MS WORD*

Microsoft Word is a widely used word processing software that allows users to create, edit, and format documents. It provides a range of features, including text formatting, spell-check, and the ability to insert images and tables. Microsoft Word is part of the Microsoft Office suite and is known for its user-friendly interface and versatile tools for document creation and collaboration.

## 2.3 Draw.io



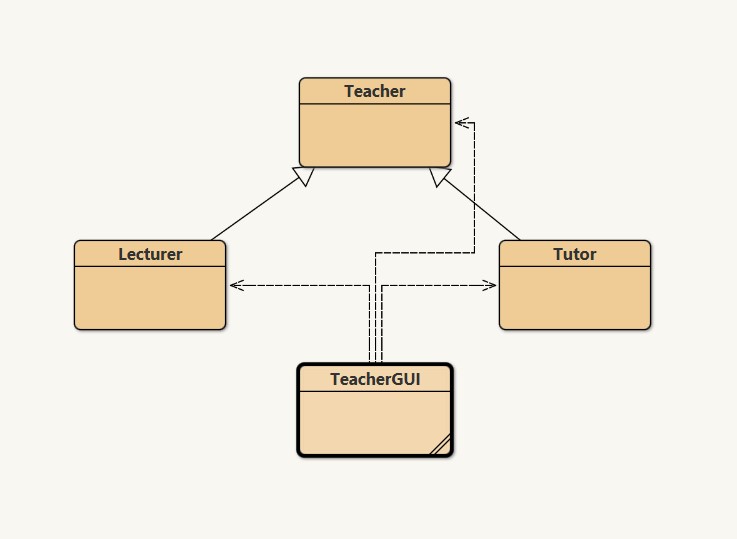
*Figure 3: DRAW.IO*

This a web-based diagramming tool that enables users to create flowcharts, diagrams, and visual representations for various purposes. It offers an intuitive and easy-to-use interface, supporting collaboration with real-time editing and sharing

capabilities. [draw.io](http://draw.io/) is commonly used for creating organizational charts, mind maps, and other visual diagrams, making it a versatile and accessible tool for both professionals and students.

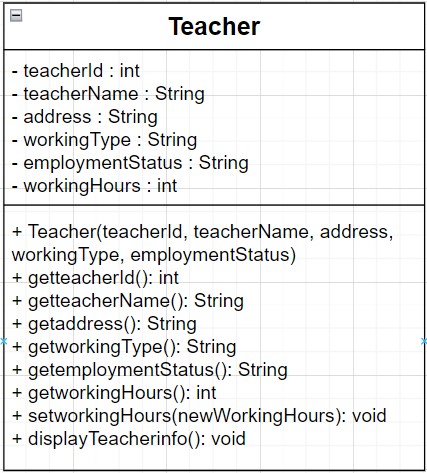
# 3. CLASS DIAGRAM

Class diagram is a visual representation of the structure and relationships within a Java program. It illustrates classes, their attributes, and methods, showcasing how they interact with each other. In the context of our coursework the class diagram prepared shows the framework discussed in the coursework questions. It describes relationships and interconnections. It is a dynamic map that guides us.



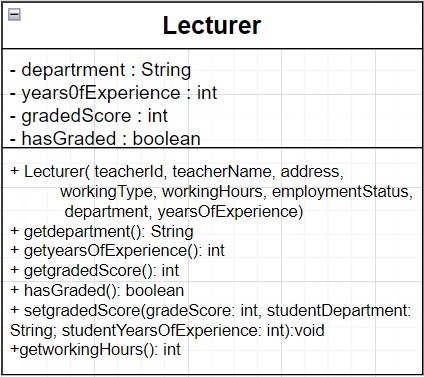
*Figure 4: CLASS DIAGRAM IN BLUE J*

## 3.1. Teacher Class Diagram

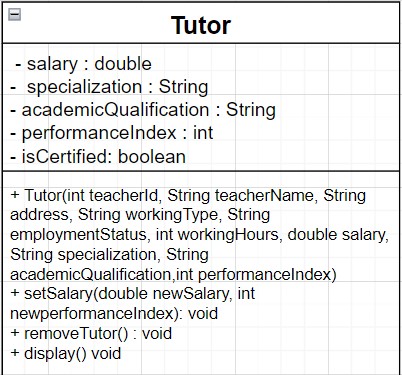


*Figure 5: Class diagram of teacher.*

## 3.2. Lecturer Class Diagram

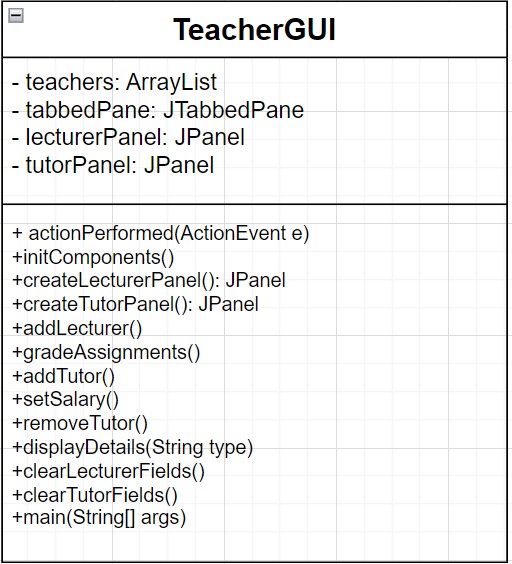
*Figure 6: Class Diagram of Lecturer.*

## 3.3. Tutor Class Diagram

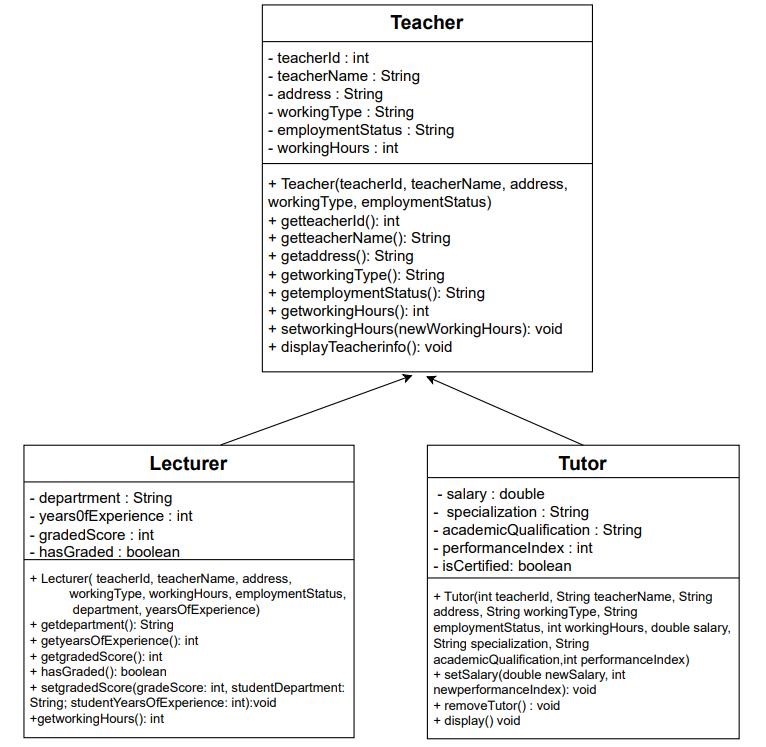


*Figure 7: Class diagram of Tutor.*

## 3.4. Combined Class Diagram



*Figure 8: class diagram of TeacherGUI.*



*Figure 9: COMBINED CLASS DIAGRAM.*

# 4. PSEUDOCODE

Pseudocode is not an actual programming language but an informal way of programming description that does not require any strict programming language syntax or underlying technology considerations. It is used for creating an outline or a rough draft of a program. Pseudocode summarizes a program's flow but excludes underlying details. So, it cannot be compiled into an executable program. It uses short terms or simple English language syntaxes to write code for programs before it is converted into a specific programming language.

START PROGRAM

DEFINE TeacherGUI CLASS

DEFINE teachers LIST

DEFINE lecturerPanel, tutorPanel

DEFINE lecturerFields, lecturerRadioButtons, lecturerComboBoxes

DEFINE tutorFields, tutorRadioButtons, tutorComboBoxes

FUNCTION getInsertedDetails(type)

IF type is "Lecturer"

POPULATE details WITH lecturerFields

ELSE IF type is "Tutor"

POPULATE details WITH tutorFields

END IF

RETURN details

END FUNCTION

FUNCTION displayDetails(type) details = getInsertedDetails(type)

IF details is not empty

DISPLAY details in a message dialog

ELSE

DISPLAY "No details" message

END IF

END FUNCTION

FUNCTION createLecturerPanel()

CREATE lecturerPanel

ADD lecturerFields, lecturerRadioButtons, lecturerComboBoxes to lecturerPanel RETURN lecturerPanel

END FUNCTION

FUNCTION createTutorPanel()

CREATE tutorPanel

ADD tutorFields, tutorRadioButtons, tutorComboBoxes to tutorPanel

RETURN tutorPanel

END FUNCTION

FUNCTION addLecturer()

GET lecturerFields values

CREATE Lecturer object with lecturerFields values

ADD Lecturer object to teachers LIST

DISPLAY success message

END FUNCTION

FUNCTION gradeAssignments()

GET lecturerFields values

FIND Lecturer object in teachers LIST with matching teacherID

IF Lecturer object found

CALL gradedAssignment() method on Lecturer object

DISPLAY success message

ELSE

DISPLAY error message

END IF

END FUNCTION

FUNCTION addTutor()

GET tutorFields values

CREATE Tutor object with tutorFields values

ADD Tutor object to teachers LIST

DISPLAY success message

END FUNCTION

FUNCTION setSalary()

GET tutorFields values

FIND Tutor object in teachers LIST with matching teacherID

IF Tutor object found

CALL SetSalary() method on Tutor object

DISPLAY success message

ELSE

DISPLAY error message

END IF

END FUNCTION

FUNCTION removeTutor()

GET tutorID

FIND Tutor object in teachers LIST with matching teacherID

IF Tutor object found

CALL removeTutor() method on Tutor object

DISPLAY success message

ELSE

DISPLAY error message

END IF

END FUNCTION

FUNCTION clearLecturerFields()

CLEAR lecturerFields

SET default values for lecturerRadioButtons and lecturerComboBoxes

END FUNCTION

FUNCTION clearTutorFields()

CLEAR tutorFields

SET default values for tutorRadioButtons and tutorComboBoxes

END FUNCTION

FUNCTION main()

CREATE TeacherGUI instance

SET visible to true

END FUNCTION

END PROGRAM

# METHOD DESCRIPTION OF WHAT EACH BUTTONS DOES

There are two main methods are in Teacher GUI

**Main method**: This is the entry point of the application. It creates an instance of the teacher GUI class and makes it visible on the screen. This method doesn't directly handle any button actions.

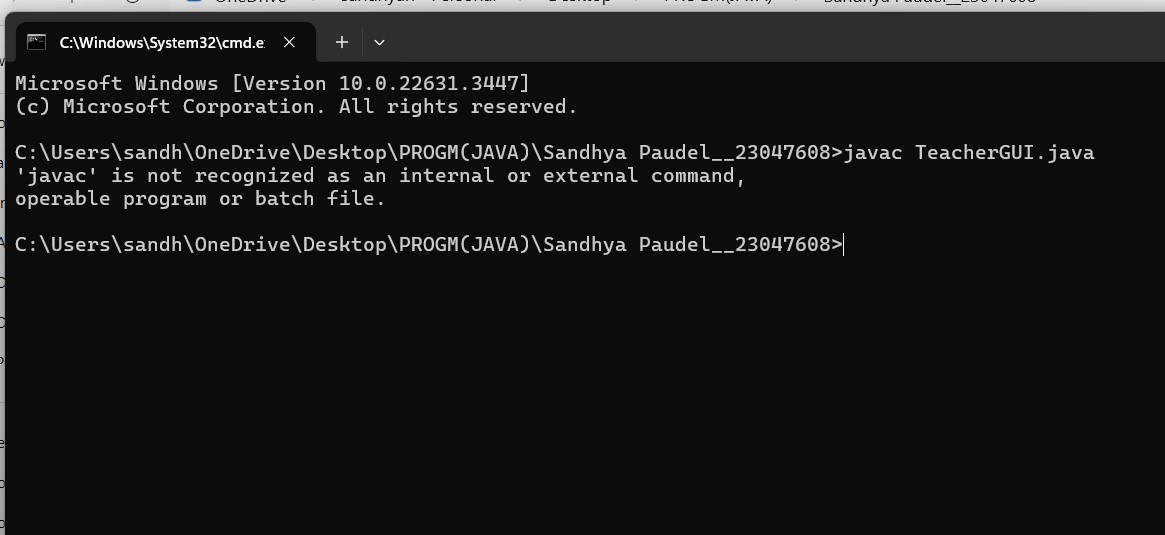
**actionPerformed method:**  This method is typically implemented by classes that implement the ActionListener interface. It is called whenever an action event occurs, such as when a button is clicked.

Description of what each buttons does:

|  |  |  |
| --- | --- | --- |
| Methods | | Description |
| Add Lecturer Button | | Adds a new Lecturer object to the teacher list by obtaining the values from the input fields in the Lecturer panel. |
| Grade Assignments Button | | Grades assignments for a specific Lecturer by obtaining the necessary values from the input fields in the Lecturer panel. |
| Display Lecturer Details Button | | Displays a dialog box containing the details of the Lecturer retrieved from the input fields in the Lecturer panel. |
| Clear Button | | Resets all the input fields in the Lecturer panel to their default or empty values. |
| Methods | Description | | |
| Add Tutor Button | Adds a new Tutor object to the teacher list by retrieving the values from the input fields in the Tutor panel. | | |
| Set Salary Button | Sets or updates the salary of a specific Tutor by retrieving the new salary and performance index values from the input fields in the Tutor panel. | | |
| Display Tutor Details Button | Displays a dialog box containing the details of the Tutor retrieved from the input fields in the Tutor panel. | | |
| Removes Tutor Button | Removes a specific Tutor from the teachers list by retrieving the teacher ID from the input field in the Tutor panel. | | |
| Clear Button | Resets all the input fields in the Tutor panel to their default or empty values. | | |

### 6. TESTING

#### 5.1. Test 1 – Compile and running using command prompt



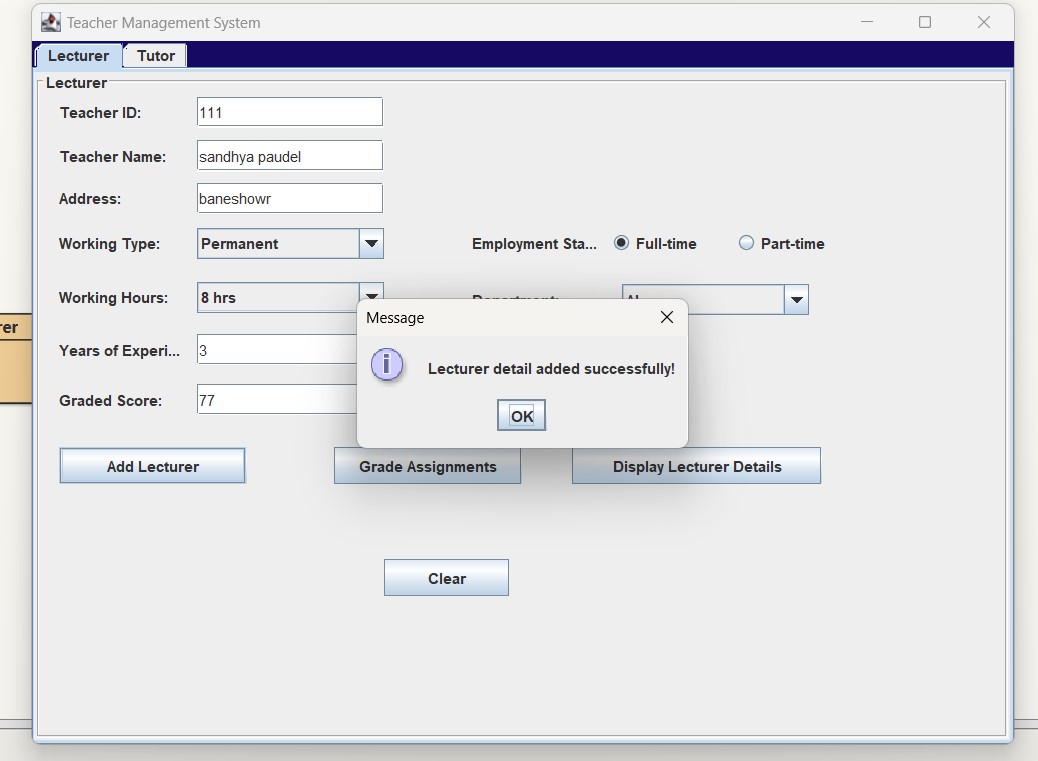
I regret to inform you that despite my best efforts, I was unable to complete the requirements to open the GUI in the command prompt. I devoted considerable time and attention to resolving this matter, but unfortunately, I was unsuccessful. I apologize for any inconvenience this may cause and assure you that I will continue to explore alternative solutions to address this issue promptly. Thank you for your understanding.

#### 5.2. TEST 2 - Evidence For the Buttons Functionalities

|  |  |
| --- | --- |
| Test No. | 2 |
| Objective: | To show the function of the button |
| Action: | -Add Lecturer button is inspected  -Data is added if the user enters the value  -Shows message in both case where the data is added, or data is missing    -Add tutor button is inspected   * Data is added if the user enters the value   -Shows message in both case where the data is added, or data is missing    -Grade Assignment (lecturer) Button is inspected   * Data is added if the user enters the value   -Shows message in both case where the data is added, or data is missing    -Set Salary button is inspected   * Data is added if the user enters the value   -Shows message in both case where the data is added, or data is missing     * Remove tutor button is added   -Removes the data |
| Expected Results | All buttons should function |
| Actual Results | The buttons functions as expected |
| Conclusion | Test is successful. |

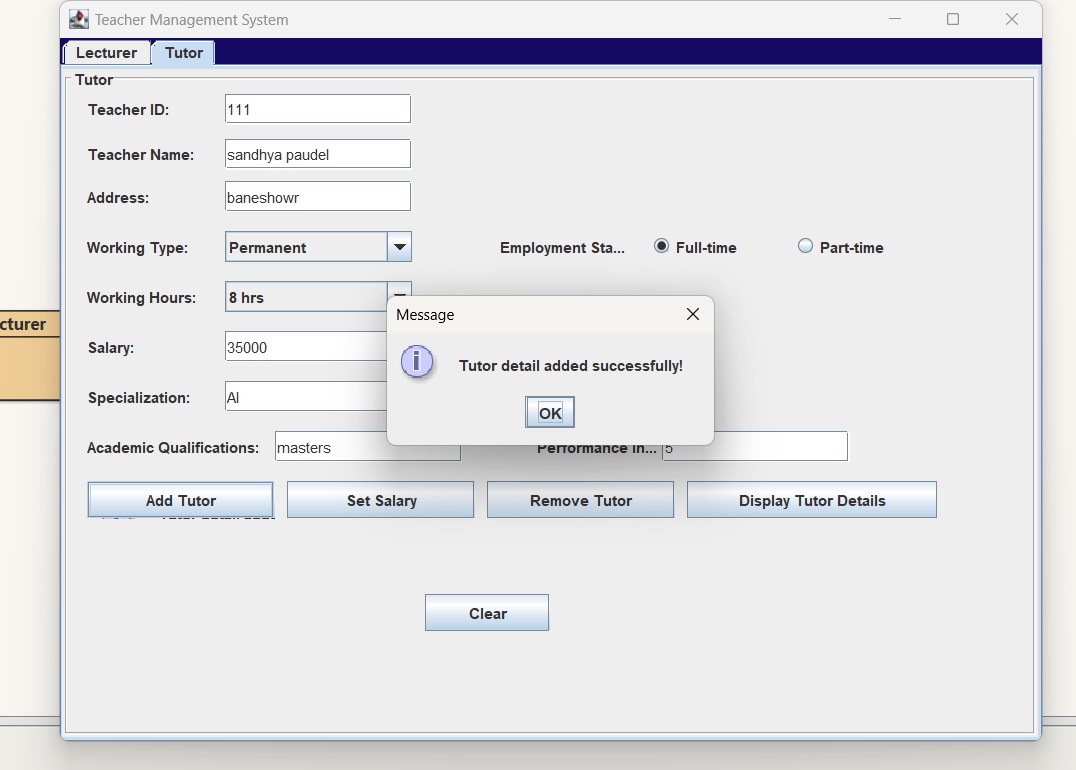
Output results.

1. Add lecturer button



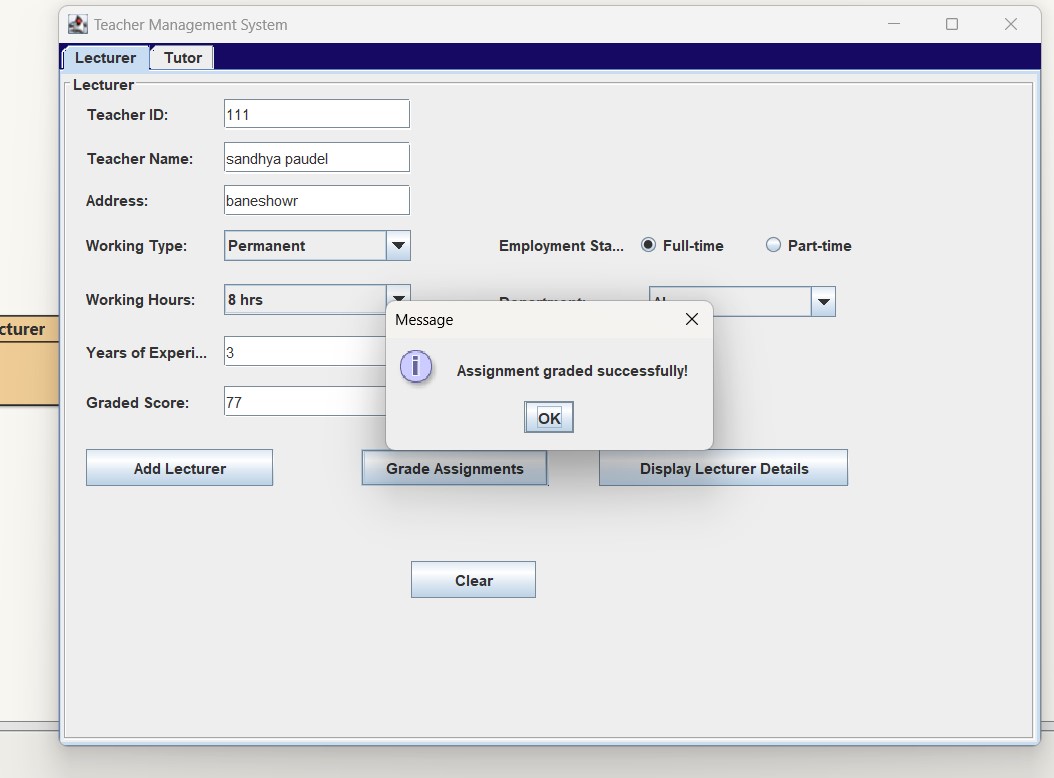
*Figure 10: Screenshot of lecturer button message dialog.*

Add Tutor button.



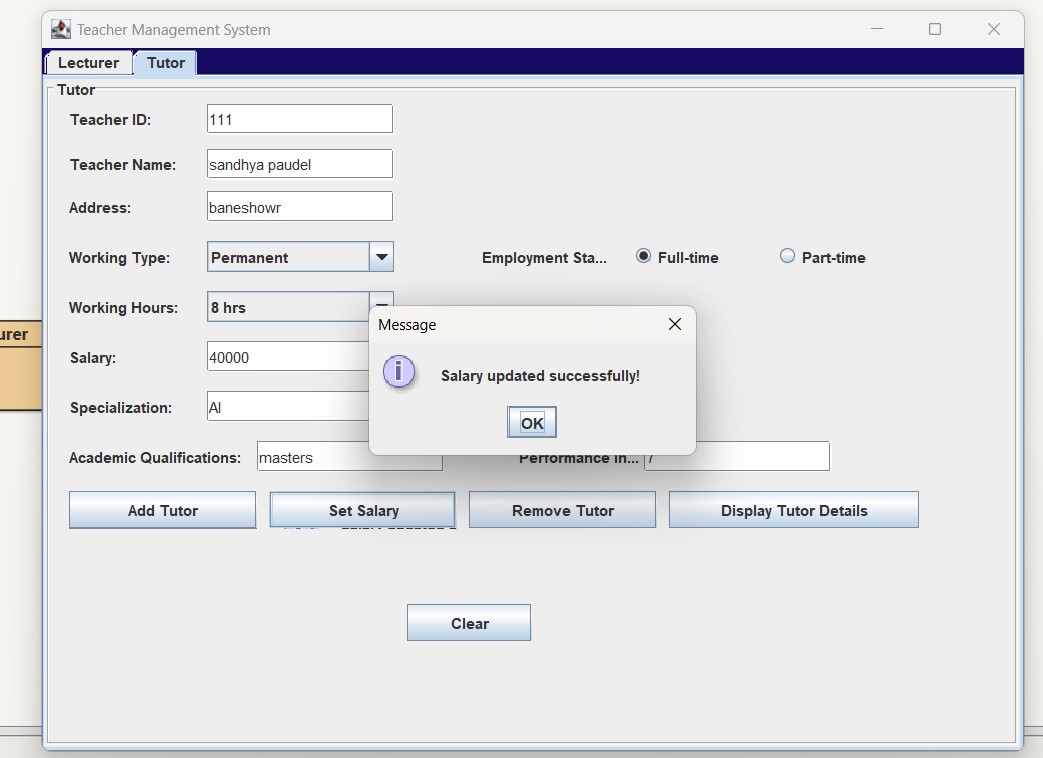
*Figure 11: screenshot of add tutor message dialog.*

1. Grade assignments from lecturer.



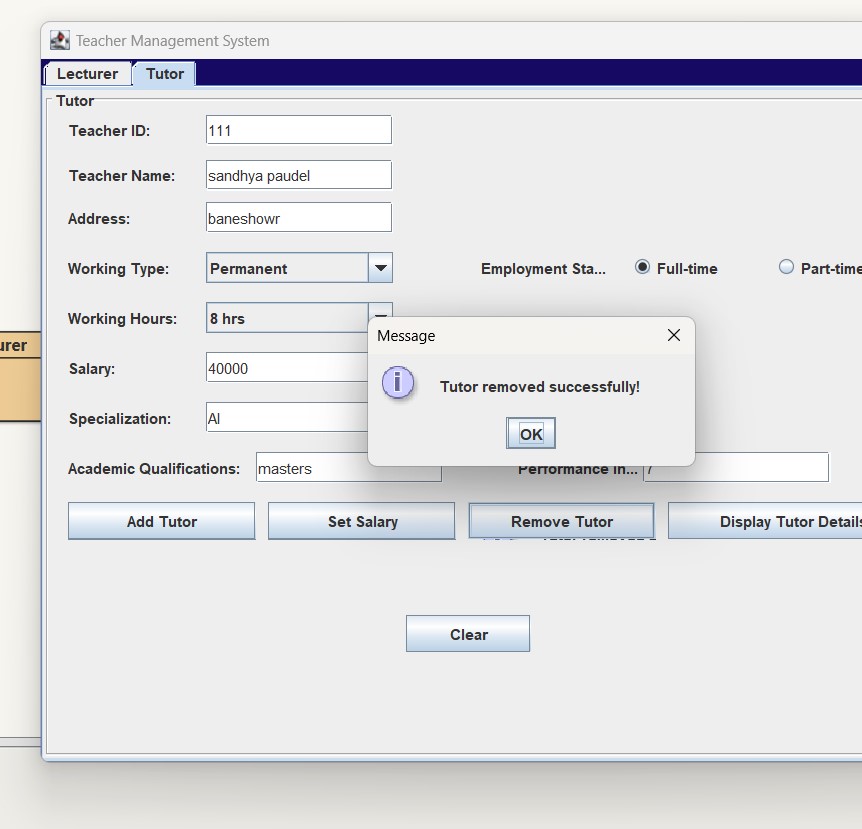
*Figure 12: Screenshot of grade assignments message dialog.*

1. Set the Salary



*Figure 13: Screenshot of message dialog. of set salary.*

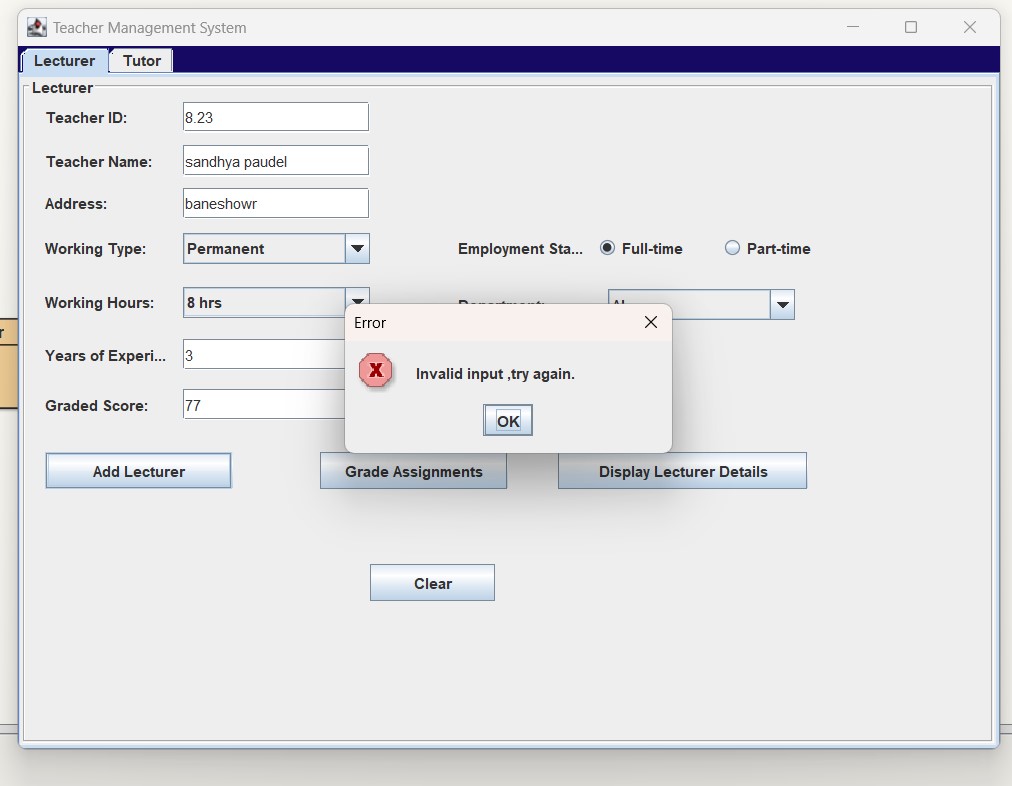
1. Remove the tutor.



*Figure 14: Screenshot of dialog box of remove tutor.*

**5.3. Test 3 - Check if appropriate dialog boxes appear when unsuitable values are added.**

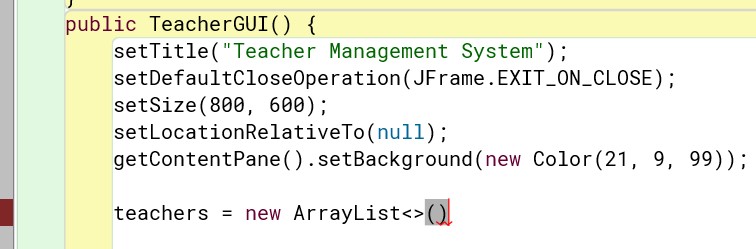
|  |  |
| --- | --- |
| Test 3 | Checking if appropriate value appears when unsuitable values are added |
| Objective: | To enter unsuitable value and check dialog box |
| Action: | Invalid Teacher id was entered. |
| Expected Results: | Dialog box with a message of invalid input would appear. |
| Actual Results: | Dialog box with message of invalid input did appear |
| Conclusion: | When invalid/ unsuitable Teacher id is entered the system does alart. |



*Figure 15: screenshot of an appropriate message dialog box.*

### 7. ERROR DETECTION AND CORRECTION

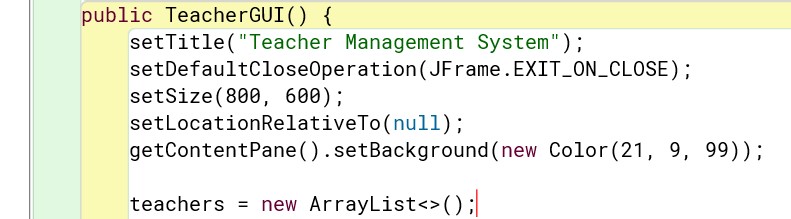
#### 7.1. Syntax Error Detection



*Figure 16: Syntax error detection.*

In this program, a frequently observed mistake is evident. The absence of semicolon at the end of the code line is the most occurring mistake among others.

#### 7.2. Syntax Error Correction

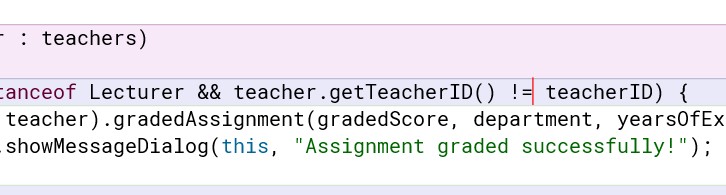


*Figure 17: Syntax error correction.*

After adding semicolon at the end of the code line, the error is corrected.

#### 7.3. Semantic Error Detection

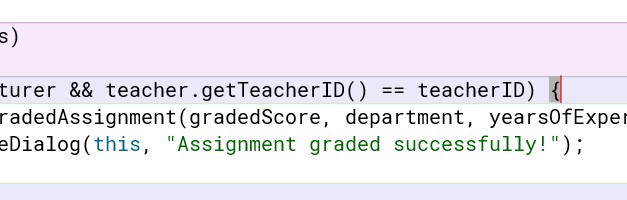
The semantic error means If there is a semantic error in your program, it will run successfully in the sense that the computer will not generate any error messages. However, your program will not do the right thing. It will do something else. Specifically, it will do what you told it to do.



*Figure 18: semantic error detection.*

Not equal sign (! =) was found in place compression to sign (==).

#### 7.4. Semantic error Correction

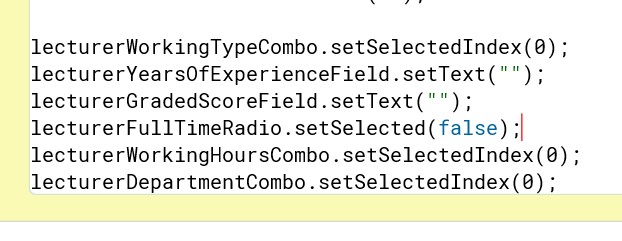


*Figure 19: Semantic error correction.*

#### 7.5. Logical Error

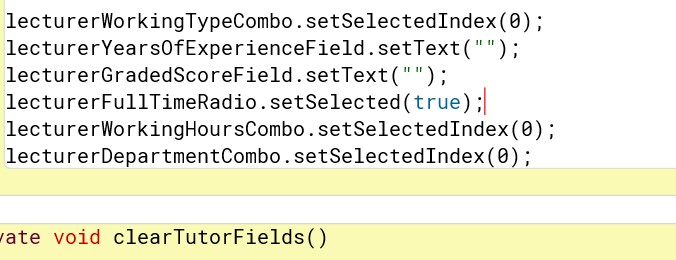
Logical errors in Java occur when the code is syntactically correct but does not produce the expected output due to flawed logic.

### Logical Error Detection



*Figure 20: logical error detection.*

#### 7.6. Logical Error Correction



*Figure 21: Logical Error Correction*

### Conclusion

Developing the Teacher Management System using Java Swing GUI provided invaluable insights into object-oriented programming, event handling, and GUI design. Overcoming challenges such as understanding event-driven programming and mastering layout management improved my skills. Handling user input validation and implementing error handling mechanisms enhanced the application's reliability. This project reinforced my grasp of OOP principles, fostering code reusability and modular design. Moving forward, I intend to delve into advanced topics like multi-threading and database integration while deepening my understanding of design patterns for creating more robust applications.