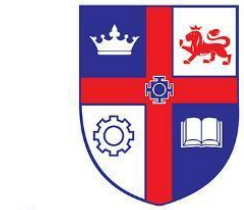
LONDON METROPOLITAN UNIVERSITY



islington college
(इस्लिङ्टन कलेज)

CS4001NI Programming

30% Individual Coursework

2023-24 Autumn

Student Name: Pawan Pokhrel

London Met ID: 23048667

College ID: NP01AI4A230127

Group: AI3

Assignment Due Date: Sunday, March 10, 2024

Assignment Submission Date: Sunday, March 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.

TABLE OF CONTENT

Contents

LIST OF FIGURES

LIST OF TABLES

CHAPTER I: INTRODUCTION

1.1. Background

The progression of the CS4001NI Programming Module at Islington College into its second phase underscores a deepening engagement with Java programming principles. Building upon the foundation laid in the initial coursework, this phase introduces a pivotal shift towards the integration of graphical user interface (GUI) elements into Java applications. The incorporation of GUI components not only enhances user interaction but also reflects the practical applicability of Java in developing dynamic, responsive, and user-friendly software solutions. As students delve into the intricacies of GUI design and implementation, they embark on a journey towards mastering the holistic development lifecycle of Java applications, from conceptualization to execution. As students navigate through the complexities of GUI design and implementation, they gain valuable insights into the iterative process of refining user interfaces to meet evolving user needs and preferences, thereby fostering a deeper understanding of user-centric design principles and best practices.

In this stage of the CS4001NI Programming Module, the emphasis expands beyond mere code implementation to encompass the creation of an intuitive and interactive interface for managing teacher details. With the integration of a graphical user interface (GUI), students are poised to explore the symbiotic relationship between backend functionality and frontend usability. This transition represents a pivotal moment in the learning journey, as students confront the complexities of translating program logic into visually engaging user experiences. Through hands-on experimentation and problem-solving, students not only refine their Java programming skills but also gain invaluable insights into the iterative design process essential for crafting robust and user-centric software applications. As they navigate the intricacies of GUI development, students are primed to cultivate a comprehensive understanding of Java's versatility and its capacity to empower developers in addressing real-world challenges with ingenuity and precision.

1.2. Rationale

The rationale for extending the CS4001NI Programming Module to incorporate GUI implementation lies in its alignment with industry demands and trends. In contemporary software development, graphical user interfaces (GUIs) play a pivotal role in enhancing user experience and facilitating intuitive interaction with applications. By integrating GUI components into Java programming coursework, the module equips students with essential skills that are directly applicable in industrial settings. GUIs are ubiquitous across various domains, from enterprise software to consumer applications, underscoring their significance in modern software development practices. Mastery of GUI design and implementation empowers developers to create software solutions that are not only functional but also aesthetically pleasing and user-friendly, thereby meeting the evolving needs and expectations of end-users. As such, the incorporation of GUIs into the curriculum reflects a forward-thinking approach aimed at preparing students for careers in software development by equipping them with the practical skills needed to excel in a competitive and dynamic industry landscape.

1.3. Methodology

During the preparation of the programming code and the report for the project, a combination of tools was employed. BlueJ, a user-friendly integrated development environment (IDE) for Java, served as the primary platform for writing and compiling Java code, facilitating seamless navigation between classes and methods. Additionally, MS Word was utilized for the creation and formatting of the report, ensuring clarity and coherence in documenting the project's methodology and findings. Draw.io, an intuitive diagramming tool, proved instrumental in crafting visual representations such as class diagrams, aiding in the visualization of program structures and relationships between classes. Through the synergistic utilization of these tools, an efficient and organized approach was adopted to execute the project tasks and deliverables effectively.

Tools Used:

1.3.1. BlueJ

BlueJ served as the primary integrated development environment (IDE) for writing and compiling Java code. Its user-friendly interface and seamless integration with Java Development Kit (JDK) provided an intuitive platform for coding, debugging, and executing Java programs. With features such as visual class diagrams and interactive object creation, BlueJ facilitated a hands-on approach to learning and understanding object-oriented programming concepts.

1.3.2. Ms-Word

Microsoft Word (MS Word) was utilized for the creation and formatting of the project report. As a versatile word processing software, MS Word offered a range of formatting options and templates to streamline the documentation process. Its collaborative features enabled multiple team members to contribute to the report simultaneously, ensuring coherence and consistency in content presentation.

1.3.3. Draw.io

Draw.io was employed for the creation of visual representations, such as class diagrams, to aid in the visualization of program structures and relationships between classes. As an intuitive diagramming tool with a wide range of shapes and symbols, Draw.io facilitated the creation of clear and concise diagrams. Its collaborative features allowed team members to collaborate in real-time, making it an effective tool for visualizing complex concepts and designs.

1.4. Objective

1.4.1. General Objective

The general objective of the coursework is to demonstrate the practical application of Java programming principles in real-world contexts by integrating graphical user interface (GUI) elements into Java applications. This assignment aims to showcase how GUI components can enhance user interaction and improve the usability of software solutions. Through this assignment, students will gain valuable experience in developing dynamic and intuitive user interfaces, preparing us for roles where user-centric design and interface development are essential.

1.4.2. Specific Objectives

The specific objectives of the coursework:

- Implement a GUI for the existing Java application.
- Integrate GUI components for user interaction.
- Add functionality for managing teachers (lecturers and tutors) within the GUI.
- Ensure error handling and provide informative feedback to users.

CHAPTER II: CLASS DIAGRAM

2.1. Definition

Class diagram is a form of static structure diagram that demonstrates the classes, attributes, behaviors (or methods), and relationships that exists between objects or classes in a system to describe the structure of the system (Uzunbayir, 2018). There are three rows and one column in this table diagram. The class name, or Class Name, is shown in the first row of the column. The information about the characteristics or properties of that class, together with the matching datatype and access modifier, is contained in the second row. The information about the class's methods or behaviors, together with the return type, datatypes for the arguments, and the method's access modifier, are finally contained in the third row. Certain symbols or signs are used in class diagrams when indicating the access modifier of attributes or methods, such as '+' for public modifiers, '-' for private modifiers, '#' for protected modifiers, and '~' for the default modifier.

A general conceptual class diagram can be observed below:

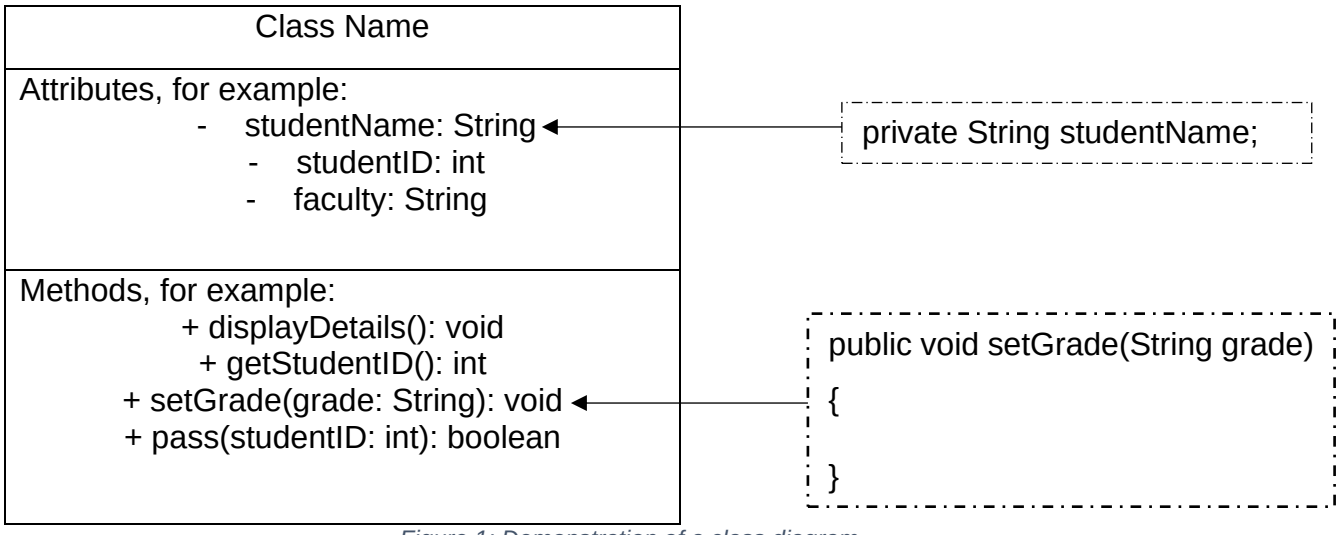


Figure 1: Demonstration of a class diagram

2.2. Class Diagram for each class

2.2.1. Teacher Class

The class diagram of the Teacher class lists the six private characteristics along with the associated datatypes. The constructor of the class, Teacher(), is the first method on the list of methods for the Teacher class.

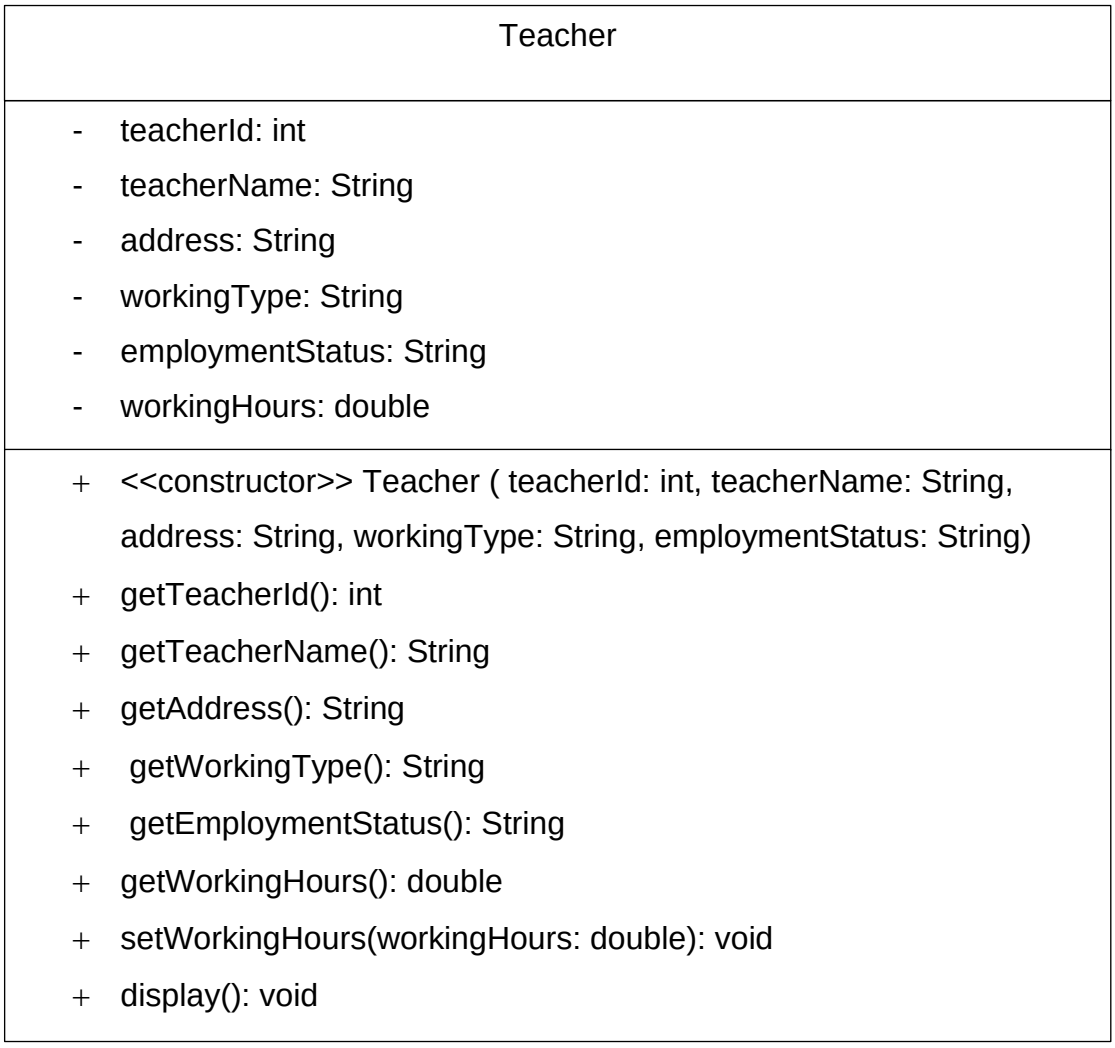


Figure 2: Class Diagram for Teacher Class

The methods list portion of the Teacher Class diagram contains a list of all the methods, including the constructor, getter methods for attributes with private modifiers, a setter method for setting the working hours, and a show() method for displaying all the teacher's data.

2.2.2. Lecture Class

Four private characteristics plus an attribute termed "grade" with a default access modifier make up the Lecturer Class. The class diagram below lists these features together with the appropriate datatypes and access modifiers. The class constructor is at the top of the list of methods in the class diagram, i.e. The Lecturer().

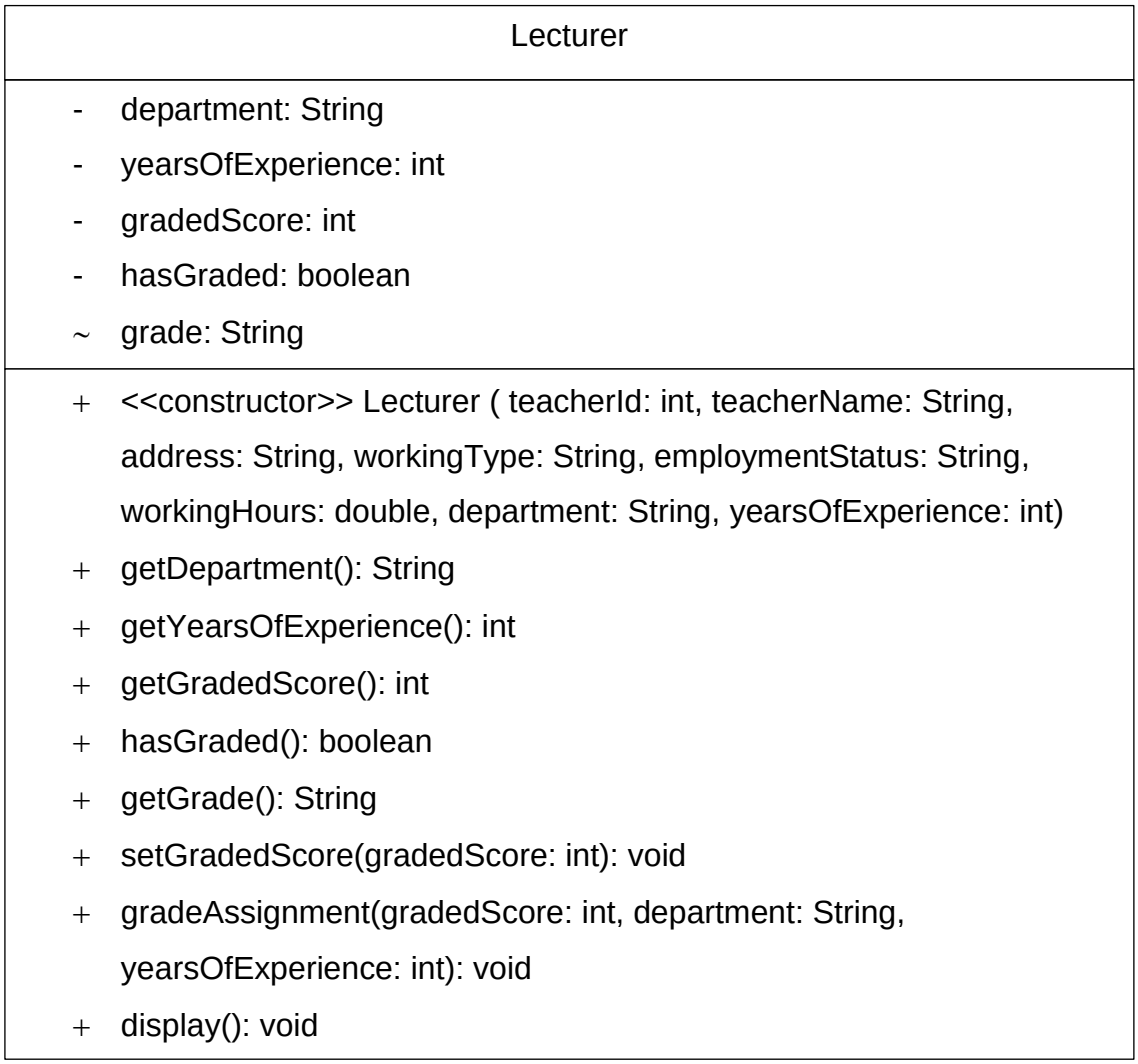


Figure 3: Class Diagram for Lecturer Class

The methods list section of the lecturer class diagram contains a list of all the methods, including the constructor, getter methods for each attribute with private or default modifiers, a setter method for setting the graded score, and a display() method for displaying all the lecturer's details.

2.2.3. Tutor Class

There are five attributes in the tutor class, each having a private access modifier. The class diagram below lists these features together with the appropriate datatypes and access modifiers. The class constructor, Tutor(), is at the top of the list of methods in the class diagram.

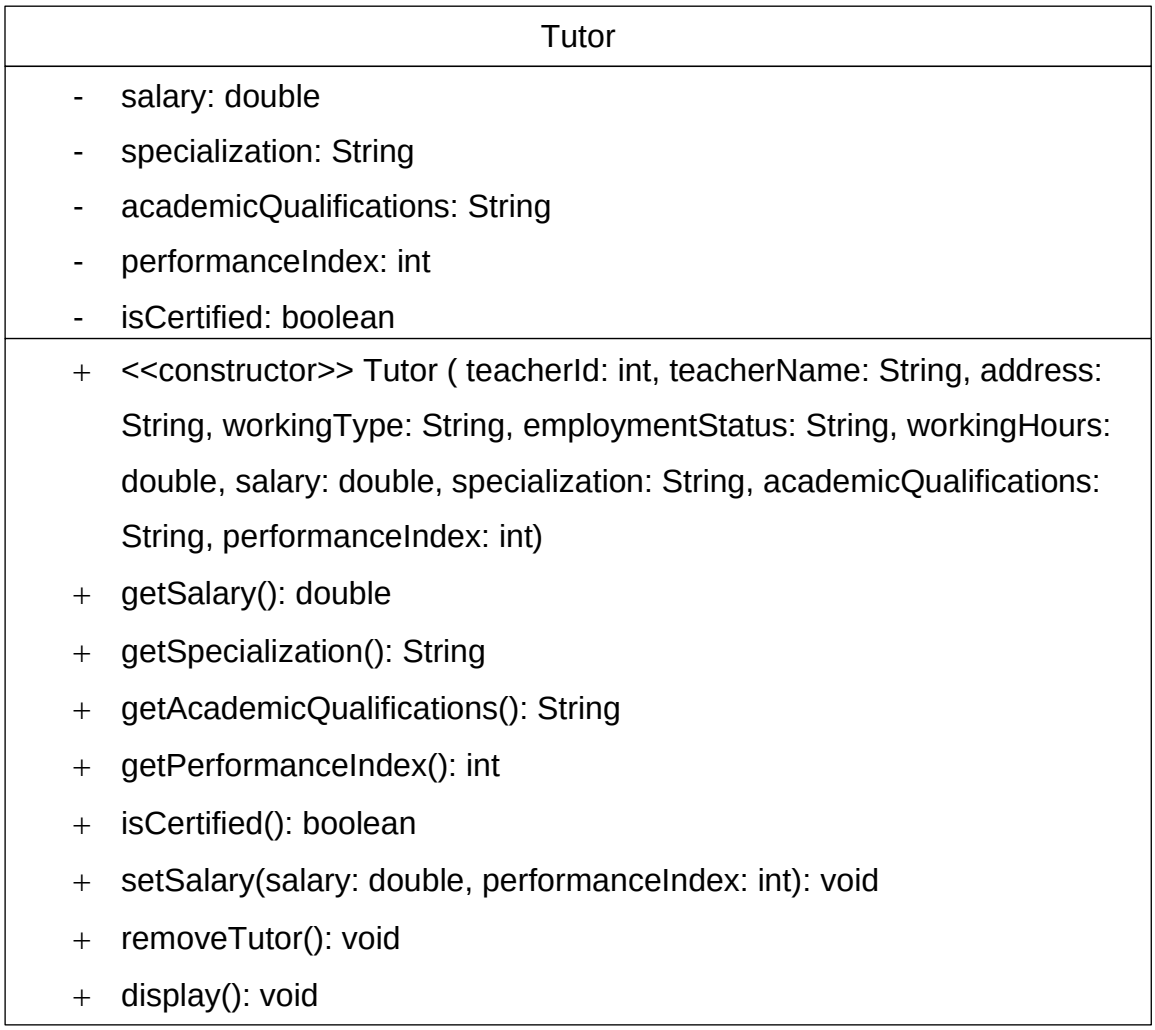


Figure 4: Class Diagram for Tutor Class

The methods list section in the class diagram of the tutor class contains a list of all the methods, including the constructor, getter methods for each attribute with private modifiers, a setter method for setting the new salary, a removeTutor() method for removing the data of the tutor who has not been certified, and a display() method for displaying all the tutor's details.

2.2.4. TeacherGUI Class

There is a single attribute in the TeacherGUI class for storing the objects as the data using ArrayLists. The use of the ActionListeners in the code provides with the responsiveness and functionality of the program. The single method used in the class is the main method.

TeacherGUI
~ teachers: ArrayList<Teacher> ~ teachers: ArrayList<Teacher> ~ body, welcomPage: JFrame ~ addLecturer, addTutor, gradeAssignment, setSalary, dsplay, clear: JButton ~ teacherId,teacherName……:JLabel ~ buttonsPanel, lecurerBody, tutorBody, navbar: JPanel ~ titleFont, textFont, labelFont: Font ~ teacherIdField , teachernameField, salaryField……..: JTextField
+ main(args: String[]): void + actionPerfromed(ActionEvent e): void + actionPerfromed(ActionEvent e): void + actionPerfromed(ActionEvent e): void + actionPerfromed(ActionEvent e): void + actionPerfromed(ActionEvent e): void + actionPerfromed(ActionEvent e): void + actionPerfromed(ActionEvent e): void + actionPerfromed(ActionEvent e): void

Figure 5: Class Diagram for TeacherGUI Class

2.3. Inheritance Diagram

The TeacherGUI class implements a graphical user interface (GUI) for the Java application, allowing users to interact visually. It enables actions like adding, grading, and removing teachers (lecturers and tutors) with ease.

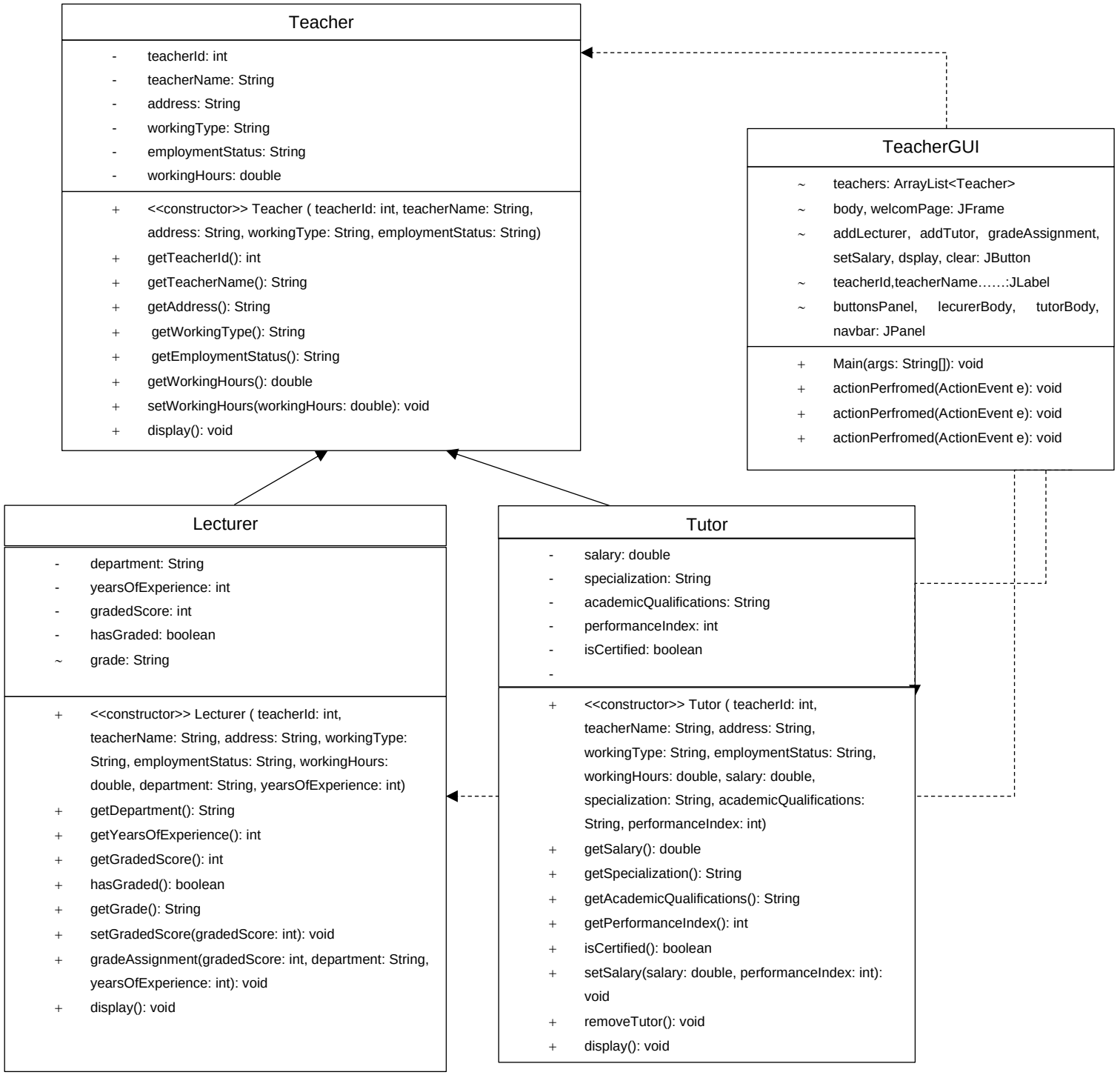


Figure 6: Inheritance Diagram

CHAPTER III: PSEUDOCODE

3.1. Definition

A popular method to articulate algorithms at a high level is by means of pseudo code. It's a non-standard language that can convey the algorithm flow structure in a way that's similar to natural language. Before the actual coding process, it is frequently used in technical documents and scientific publications to communicate algorithms and the business logic (Shan–shan, 2021). The first word is capitalized and there is only one statement per line. The structure of the program should be illustrated by appropriate indentation management. Multiline structures need to have the appropriate keyword to close them. For instance, IF must terminate with IF END or END IF. All actions in the code should be carried out using the action verbs CREATE, GENERATE, GET, COMPUTE, ASSIGN, etc.

Let's take a Java code and observe its corresponding pseudocode:-

- **Java Code for creation of class**

```
public class [class_name]
{
        private int var_1;
        public String var_2;
        protected double var_3;
        boolean var_4;
}
```

- **Corresponding Pseudocode**

CREATE CLASS [class_name]
    DECLARE instance variable var_1 as int using private access
    DECLARE instance variable var_2 as String using public access
    DECLARE instance variable var_3 as double using protected access
    DECLARE instance variable var_4 as boolean using default access
END CLASS

3.2. Pseudocode for TeacherGUI Class

```
IMPORT all the Libraries  in java.awt

IMPORT all the Libraries in javax.swing

IMPORT all the Libraries in java.awt.event

IMPORT all the Libraries in java.util


CLASS TeacherGUI

    METHOD main

        DECLARE teachers AS ArrayList of Teacher

        DECLARE body AS JFrame

        DECLARE titleFont AS Font

        DECLARE textFont AS Font

        DECLARE labelFont AS Font

        DECLARE welcomePage AS JFrame

        DECLARE welcomeTitle AS JLabel

        DECLARE titlePanel AS JPanel

        DECLARE welcomeText AS JTextArea

        DECLARE welcomePanel AS JPanel

        DECLARE navBar AS JPanel

        DECLARE lecturerBody AS JPanel

        DECLARE lecturerLeftBody AS JPanel

        DECLARE lecturerRightBody AS JPanel
```

DECLARE tutorBody AS JPanel

DECLARE tutorLeftBody AS JPanel

DECLARE tutorRightBody AS JPanel

DECLARE eastGap AS JPanel

DECLARE westGap AS JPanel

DECLARE southGap AS JPanel

DECLARE headerLabel AS JLabel

DECLARE teacherIdLabel AS JLabel

DECLARE teacherIdField AS JTextField

DECLARE teacherNameLabel AS JLabel

DECLARE teacherNameField AS JTextField

DECLARE addressLabel AS JLabel

DECLARE addressField AS JTextField

DECLARE workingTypeLabel AS JLabel

DECLARE workingTypeField AS JTextField

DECLARE employmentStatusLabel AS JLabel

DECLARE employmentStatusField AS JTextField

DECLARE workingHoursLabel AS JLabel

DECLARE workingHoursField AS JTextField

DECLARE departmentLabel AS JLabel

DECLARE departmentField AS JTextField

DECLARE yearsOfExperienceLabel AS JLabel

DECLARE yearsOfExperienceField AS JTextField

DECLARE gradedScoreLabel AS JLabel

DECLARE gradedScoreField AS JTextField

DECLARE salaryLabel AS JLabel

DECLARE salaryField AS JTextField

DECLARE specializationLabel AS JLabel

DECLARE specializationField AS JTextField

DECLARE academicQualificationsLabel AS JLabel

DECLARE academicQualificationsField AS JTextField

DECLARE performanceIndexLabel AS JLabel

DECLARE performanceIndexField AS JTextField

DECLARE headerPanel AS JPanel

DECLARE teacherIdPanel AS JPanel

DECLARE teacherNamePanel AS JPanel

DECLARE addressPanel AS JPanel

DECLARE workingTypePanel AS JPanel

DECLARE employmentStatusPanel AS JPanel

DECLARE workingHoursPanel AS JPanel

DECLARE departmentPanel AS JPanel

DECLARE yearsOfExperiencePanel AS JPanel

DECLARE gradedScorePanel AS JPanel

DECLARE salaryPanel AS JPanel

DECLARE specializationPanel AS JPanel

DECLARE academicQualificationsPanel AS JPanel

DECLARE performanceIndexPanel AS JPanel

DECLARE welcomeNav AS JPanel

DECLARE welcomeButtons AS JPanel

DECLARE directToLecturerPage AS JButton

DECLARE directToTutorPage AS JButton

DECLARE addLecturer AS JButton

DECLARE addTutor AS JButton

DECLARE gradeAssignments AS JButton

DECLARE setSalary AS JButton

DECLARE removeTutor AS JButton

DECLARE displayLecturer AS JButton

DECLARE displayTutor AS JButton

DECLARE clear AS JButton

DECLARE welcomeToLecturer AS JButton

DECLARE welcomeToTutor AS JButton

DECLARE lecturerButtons AS JPanel

DECLARE tutorButtons AS JPanel

DECLARE commonButtons AS JPanel

SET visibility of welcomePage to true

SET extended state of welcomePage to maximize both horizontally and vertically

SET layout of welcomePage to null

SET background color of welcomeNav to RGB(54, 80, 107)

SET background color of titlePanel to RGB(54, 80, 107)

SET background color of welcomePanel to RGB(112, 154, 191)

SET preferred size of welcomeText to 800 width and 150 height

Enable line wrap and word wrapping for welcomeText

Add welcomeNav to welcomePage

SET bounds of welcomeNav to (0, 0, 1463, 120)

SET layout of welcomeNav to FlowLayout with center alignment

Add titlePanel to welcomeNav

Add welcomeTitle to titlePanel

Add welcomePanel to welcomePage

SET bounds of welcomePanel to (250, 140, 900, 400)

SET layout of welcomePanel to a 2x1 grid layout with 0 horizontal and 50 vertical gaps

Add welcomeText to welcomePanel

SET preferred size of welcomeButtons to 1000 width and 100 height

Add welcomeToLecturer button to welcomeButtons

Add welcomeToTutor button to welcomeButtons

SET preferred size of welcomeToLecturer button to 100 width and 50 height

SET preferred size of welcomeToTutor button to 100 width and 50 height

SET layout of body to BorderLayout

Add navBar to body in the NORTH position

Add eastGap to body in the EAST position

Add westGap to body in the WEST position

SET layout of navBar to FlowLayout with center alignment

ADD ActionListener to the JButton goToGradeAssignments

METHOD actionPerformed(ActionEvent e)

SET visibility of body to true

SET visibility of welcomePage to false

SET extended state of body to maximize both horizontally and vertically

Add gradeAssignmentBody to body in the CENTER position

ADD headerPanel to navBar

ADD headerLabel to headerPanel

SET background color of headerPanel to RGB(112, 146, 190)

SET horizontal alignment of headerLabel to CENTER

SET preferred size of headerPanel to 1463 width and 60 height

SET visibility of gradeAssignmentBody to true

SET visibility of setSalaryBody to false

SET visibility of lecturerBody to false

SET visibility of tutorBody to false

SET layout of gradeAssignmentBody to a 5x1 grid layout with 0 horizontal and 50 vertical gaps

Add teacherIdPanel to gradeAssignmentBody

Add gradedScorePanel to gradeAssignmentBody

Add departmentPanel to gradeAssignmentBody

Add yearsOfExperiencePanel to gradeAssignmentBody

Add gradeButtons to gradeAssignmentBody

CREATE a new FlowLayout GapFlowLayout

SET horizontal gap of GapFlowLayout to 86

SET layout of gradeButtons to GapFlowLayout

Add backToLecturerPage button to gradeButtons

SET font of backToLecturerPage button to textFont

Add gradeAssignments button to gradeButtons

SET background color of gradeButtons to RGB(54, 80, 107)

SET preferred size of gradeAssignmentBody to 500 width and 700 height

SET background color of gradeAssignmentBody to RGB(161, 200, 242)

END METHOD

ADD ActionListener to the JButton welcomeToTutor

METHOD actionPerformed(ActionEvent e)

SET visibility of body to true

SET visibility of welcomePage to false

SET extended state of body to maximize both horizontally and vertically

Add tutorBody to body in the CENTER position

ADD headerPanel to navBar

ADD headerLabel to headerPanel

SET background color of headerPanel to RGB(112, 146, 190)

SET horizontal alignment of headerLabel to CENTER

SET preferred size of headerPanel to 1463 width and 60 height

SET visibility of lecturerBody to false

SET visibility of tutorBody to true

SET layout of tutorBody to FlowLayout

Add tutorLeftBody to tutorBody

Add tutorRightBody to tutorBody

SET layout of tutorLeftBody to a 6x1 grid layout with 0 horizontal and 50 vertical gaps

Add teacherIdPanel to tutorLeftBody

Add teacherNamePanel to tutorLeftBody

Add addressPanel to tutorLeftBody

Add workingTypePanel to tutorLeftBody

Add specializationPanel to tutorLeftBody

Add tutorButtons to tutorLeftBody

SET layout of tutorRightBody to a 6x1 grid layout with 0 horizontal and 50 vertical gaps

Add employmentStatusPanel to tutorRightBody

Add workingHoursPanel to tutorRightBody

Add salaryPanel to tutorRightBody

Add academicQualificationsPanel to tutorRightBody

Add performanceIndexPanel to tutorRightBody

Add commonButtons to tutorRightBody

SET layout of tutorButtons to FlowLayout

Add addTutor button to tutorButtons

Add setSalary button to tutorButtons

Add removeTutor button to tutorButtons

SET layout of commonButtons to FlowLayout

Add displayTutor button to commonButtons

SET visibility of displayLecturer to false

Add clear button to commonButtons

SET background color of tutorButtons to RGB(54, 80, 107)

SET background color of commonButtons to RGB(54, 80, 107)

SET preferred size of tutorBody to 1200 width and 700 height

SET background color of tutorBody to RGB(112, 154, 191)

SET preferred size of tutorLeftBody to 600 width and 700 height

SET preferred size of tutorRightBody to 600 width and 700 height

SET background color of tutorLeftBody to RGB(161, 200, 242)

SET background color of tutorRightBody to RGB(161, 200, 242)

END METHOD

ADD ActionListener to the JButton goToSetSalary

METHOD actionPerformed(ActionEvent e)

SET visibility of body to true

SET visibility of welcomePage to false

SET extended state of body to maximize both horizontally and vertically

Add setSalaryBody to body in the CENTER position

ADD headerPanel to navBar

ADD headerLabel to headerPanel

SET background color of headerPanel to RGB(112, 146, 190)

SET horizontal alignment of headerLabel to CENTER

SET preferred size of headerPanel to 1463 width and 60 height

SET visibility of gradeAssignmentBody to false

SET visibility of setSalaryBody to true

SET visibility of lecturerBody to false

SET visibility of tutorBody to false

SET layout of setSalaryBody to a 4x1 grid layout with 0 horizontal and 50 vertical gaps

Add teacherIdPanel to setSalaryBody

Add salaryPanel to setSalaryBody

Add performanceIndexPanel to setSalaryBody

Add setButtons to setSalaryBody

CREATE a new FlowLayout GapFlowLayout

SET horizontal gap of GapFlowLayout to 86

SET layout of setButtons to GapFlowLayout

Add backToTutorPage button to setButtons

SET font of backToTutorPage button to textFont

Add setSalary button to setButtons

SET background color of setButtons to RGB(54, 80, 107)

SET preferred size of setSalaryBody to 500 width and 700 height

SET background color of setSalaryBody to RGB(161, 200, 242)

END METHOD

ADD ActionListener to the JButton backToLecturerPage

METHOD actionPerformed(ActionEvent e)

SET visibility of body to true

SET visibility of welcomePage to false

SET extended state of body to maximize both horizontally and vertically

Add lecturerBody to body in the CENTER position

ADD headerPanel to navBar

ADD headerLabel to headerPanel

SET background color of headerPanel to RGB(112, 146, 190)

SET horizontal alignment of headerLabel to CENTER

SET preferred size of headerPanel to 1463 width and 60 height

SET visibility of gradeAssignmentBody to false

SET visibility of setSalaryBody to false

SET visibility of lecturerBody to true

SET visibility of tutorBody to false

SET layout of lecturerBody to FlowLayout

Add lecturerLeftBody to lecturerBody

Add lecturerRightBody to lecturerBody

SET layout of lecturerLeftBody to a 5x1 grid layout with 0 horizontal and 50 vertical gaps

Add teacherIdPanel to lecturerLeftBody

Add teacherNamePanel to lecturerLeftBody

Add addressPanel to lecturerLeftBody

Add workingTypePanel to lecturerLeftBody

Add lecturerButtons to lecturerLeftBody

SET layout of lecturerRightBody to a 5x1 grid layout with 0 horizontal and 50 vertical gaps

Add employmentStatusPanel to lecturerRightBody

Add workingHoursPanel to lecturerRightBody

Add departmentPanel to lecturerRightBody

Add yearsOfExperiencePanel to lecturerRightBody

Add commonButtons to lecturerRightBody

CREATE a new FlowLayout GapFlowLayout

SET horizontal gap of GapFlowLayout to 86

SET layout of lecturerButtons to GapFlowLayout

Add addLecturer button to lecturerButtons

Add goToGradeAssignments button to lecturerButtons

SET layout of commonButtons to GapFlowLayout

Add displayLecturer button to commonButtons

SET visibility of displayTutor to false

Add clear button to commonButtons

SET background color of lecturerButtons to RGB(54, 80, 107)

SET background color of commonButtons to RGB(54, 80, 107)

SET preferred size of lecturerBody to 1200 width and 700 height

SET background color of lecturerBody to RGB(112, 154, 191)

SET preferred size of lecturerLeftBody to 600 width and 700 height

SET preferred size of lecturerRightBody to 600 width and 700 height

SET background color of lecturerLeftBody to RGB(161, 200, 242)

SET background color of lecturerRightBody to RGB(161, 200, 242)

END METHOD

ADD ActionListener to the JButton backToTutorPage

METHOD actionPerformed(ActionEvent e)

SET visibility of body to true

SET visibility of welcomePage to false

SET extended state of body to maximize both horizontally and vertically

Add tutorBody to body in the CENTER position

ADD headerPanel to navBar

ADD headerLabel to headerPanel

SET background color of headerPanel to RGB(112, 146, 190)

SET horizontal alignment of headerLabel to CENTER

SET preferred size of headerPanel to 1463 width and 60 height

SET visibility of gradeAssignmentBody to false

SET visibility of setSalaryBody to false

SET visibility of lecturerBody to false

SET visibility of tutorBody to true

SET layout of tutorBody to FlowLayout

Add tutorLeftBody to tutorBody

Add tutorRightBody to tutorBody

SET layout of tutorLeftBody to a 6x1 grid layout with 0 horizontal and 50 vertical gaps

Add teacherIdPanel to tutorLeftBody

Add teacherNamePanel to tutorLeftBody

Add addressPanel to tutorLeftBody

Add workingTypePanel to tutorLeftBody

Add specializationPanel to tutorLeftBody

Add tutorButtons to tutorLeftBody

SET layout of tutorRightBody to a 6x1 grid layout with 0 horizontal and 50 vertical gaps

Add employmentStatusPanel to tutorRightBody

Add workingHoursPanel to tutorRightBody

Add salaryPanel to tutorRightBody

Add academicQualificationsPanel to tutorRightBody

Add performanceIndexPanel to tutorRightBody

Add commonButtons to tutorRightBody

SET layout of tutorButtons to FlowLayout

Add addTutor button to tutorButtons

Add goToSetSalary button to tutorButtons

Add removeTutor button to tutorButtons

CREATE a new FlowLayout GapFlowLayout

SET horizontal gap of GapFlowLayout to 86

SET layout of commonButtons to GapFlowLayout

Add displayTutor button to commonButtons

SET visibility of displayLecturer to false

Add clear button to commonButtons

SET background color of tutorButtons to RGB(54, 80, 107)

SET background color of commonButtons to RGB(54, 80, 107)

SET preferred size of tutorBody to 1200 width and 700 height

SET background color of tutorBody to RGB(112, 154, 191)

SET preferred size of tutorLeftBody to 600 width and 700 height

SET preferred size of tutorRightBody to 600 width and 700 height

SET background color of tutorLeftBody to RGB(161, 200, 242)

SET background color of tutorRightBody to RGB(161, 200, 242)

END METHOD

ADD ActionListener to the JButton welcomeToTutor

METHOD actionPerformed(ActionEvent e)

SET visibility of body to true

SET visibility of welcomePage to false

SET extended state of body to maximize both horizontally and vertically

Add tutorBody to body in the CENTER position

ADD headerPanel to navBar

ADD headerLabel to headerPanel

SET background color of headerPanel to RGB(112, 146, 190)

SET horizontal alignment of headerLabel to CENTER

SET preferred size of headerPanel to 1463 width and 60 height

SET visibility of lecturerBody to false

SET visibility of tutorBody to true

SET layout of tutorBody to FlowLayout

Add tutorLeftBody to tutorBody

Add tutorRightBody to tutorBody

SET layout of tutorLeftBody to a 6x1 grid layout with 0 horizontal and 50 vertical gaps

Add teacherIdPanel to tutorLeftBody

Add teacherNamePanel to tutorLeftBody

Add addressPanel to tutorLeftBody

Add workingTypePanel to tutorLeftBody

Add specializationPanel to tutorLeftBody

Add tutorButtons to tutorLeftBody

SET layout of tutorRightBody to a 6x1 grid layout with 0 horizontal and 50 vertical gaps

Add employmentStatusPanel to tutorRightBody

Add workingHoursPanel to tutorRightBody

Add salaryPanel to tutorRightBody

Add academicQualificationsPanel to tutorRightBody

Add performanceIndexPanel to tutorRightBody

Add commonButtons to tutorRightBody

SET layout of tutorButtons to FlowLayout

Add addTutor button to tutorButtons

Add setSalary button to tutorButtons

Add removeTutor button to tutorButtons

SET layout of commonButtons to FlowLayout

Add displayTutor button to commonButtons

SET visibility of displayLecturer to false

Add clear button to commonButtons

SET background color of tutorButtons to RGB(54, 80, 107)

SET background color of commonButtons to RGB(54, 80, 107)

SET preferred size of tutorBody to 1200 width and 700 height

SET background color of tutorBody to RGB(112, 154, 191)

SET preferred size of tutorLeftBody to 600 width and 700 height

SET preferred size of tutorRightBody to 600 width and 700 height

SET background color of tutorLeftBody to RGB(161, 200, 242)

SET background color of tutorRightBody to RGB(161, 200, 242)

END METHOD

ADD ActionListener to the JButton directToLecturerPage

METHOD actionPerformed(ActionEvent e)

SET visibility of body to true

SET visibility of welcomePage to false

SET extended state of body to maximize both horizontally and vertically

Add lecturerBody to body in the CENTER position

ADD headerPanel to navBar

ADD headerLabel to headerPanel

SET background color of headerPanel to RGB(112, 146, 190)

SET horizontal alignment of headerLabel to CENTER

SET preferred size of headerPanel to 1463 width and 60 height

SET visibility of lecturerBody to true

SET visibility of tutorBody to false

SET layout of lecturerBody to FlowLayout

Add lecturerLeftBody to lecturerBody

Add lecturerRightBody to lecturerBody

SET layout of lecturerLeftBody to a 5x1 grid layout with 0 horizontal and 50 vertical gaps

Add teacherIdPanel to lecturerLeftBody

Add teacherNamePanel to lecturerLeftBody

Add addressPanel to lecturerLeftBody

Add workingTypePanel to lecturerLeftBody

Add lecturerButtons to lecturerLeftBody

SET layout of lecturerRightBody to a 5x1 grid layout with 0 horizontal and 50 vertical gaps

Add employmentStatusPanel to lecturerRightBody

Add workingHoursPanel to lecturerRightBody

Add departmentPanel to lecturerRightBody

Add yearsOfExperiencePanel to lecturerRightBody

Add commonButtons to lecturerRightBody

SET layout of lecturerButtons to FlowLayout

Add addLecturer button to lecturerButtons

Add gradeAssignments button to lecturerButtons

SET layout of commonButtons to FlowLayout

Add displayLecturer button to commonButtons

Add clear button to commonButtons

SET background color of lecturerButtons to RGB(54, 80, 107)

SET background color of commonButtons to RGB(54, 80, 107)

SET preferred size of lecturerBody to 1200 width and 700 height

SET background color of lecturerBody to RGB(112, 154, 191)

SET preferred size of lecturerLeftBody to 600 width and 700 height

SET preferred size of lecturerRightBody to 600 width and 700 height

SET background color of lecturerLeftBody to RGB(161, 200, 242)

SET background color of lecturerRightBody to RGB(161, 200, 242)

END METHOD

ADD ActionListener to the JButton directToTutorPage

METHOD actionPerformed(ActionEvent e)

SET visibility of body to true

SET visibility of welcomePage to false

SET extended state of body to maximize both horizontally and vertically

Add tutorBody to body in the CENTER position

ADD headerPanel to navBar

ADD headerLabel to headerPanel

SET background color of headerPanel to RGB(112, 146, 190)

SET horizontal alignment of headerLabel to CENTER

SET preferred size of headerPanel to 1463 width and 60 height

SET visibility of lecturerBody to false

SET visibility of tutorBody to true

SET layout of tutorBody to FlowLayout

Add tutorLeftBody to tutorBody

Add tutorRightBody to tutorBody

SET layout of tutorLeftBody to a 6x1 grid layout with 0 horizontal and 50 vertical gaps

Add teacherIdPanel to tutorLeftBody

Add teacherNamePanel to tutorLeftBody

Add addressPanel to tutorLeftBody

Add workingTypePanel to tutorLeftBody

Add specializationPanel to tutorLeftBody

Add tutorButtons to tutorLeftBody

SET layout of tutorRightBody to a 6x1 grid layout with 0 horizontal and 50 vertical gaps

Add employmentStatusPanel to tutorRightBody

Add workingHoursPanel to tutorRightBody

Add salaryPanel to tutorRightBody

Add academicQualificationsPanel to tutorRightBody

Add performanceIndexPanel to tutorRightBody

Add commonButtons to tutorRightBody

SET layout of tutorButtons to FlowLayout

Add addTutor button to tutorButtons

Add setSalary button to tutorButtons

Add removeTutor button to tutorButtons

SET layout of commonButtons to FlowLayout

Add displayTutor button to commonButtons

Add clear button to commonButtons

SET background color of tutorButtons to RGB(54, 80, 107)

SET background color of commonButtons to RGB(54, 80, 107)

SET preferred size of tutorBody to 1200 width and 700 height

SET background color of tutorBody to RGB(112, 154, 191)

SET preferred size of tutorLeftBody to 600 width and 700 height

SET preferred size of tutorRightBody to 600 width and 700 height

SET background color of tutorLeftBody to RGB(161, 200, 242)

SET background color of tutorRightBody to RGB(161, 200, 242)

END METHOD

SET preferred size of eastGap to 100 width and 914 height

SET preferred size of westGap to 100 width and 914 height

SET preferred size of southGap to 1463 width and 100 height

ADD teacherIdLabel to teacherIdPanel

ADD teacherIdField to teacherIdPanel

ADD teacherNameLabel to teacherNamePanel

ADD teacherNameField to teacherNamePanel

ADD addressLabel to addressPanel

ADD addressField to addressPanel

ADD workingTypeLabel to workingTypePanel

ADD workingTypeField to workingTypePanel

ADD employmentStatusLabel to employmentStatusPanel

ADD employmentStatusField to employmentStatusPanel

ADD workingHoursLabel to workingHoursPanel

ADD workingHoursField to workingHoursPanel

ADD departmentLabel to departmentPanel

ADD departmentField to departmentPanel

ADD yearsOfExperienceLabel to yearsOfExperiencePanel

ADD yearsOfExperienceField to yearsOfExperiencePanel

ADD gradedScoreLabel to gradedScorePanel

ADD gradedScoreField to gradedScorePanel

ADD salaryLabel to salaryPanel

ADD salaryField to salaryPanel

ADD specializationLabel to specializationPanel

ADD specializationField to specializationPanel

ADD academicQualificationsLabel to academicQualificationsPanel

ADD academicQualificationsField to academicQualificationsPanel

ADD performanceIndexLabel to performanceIndexPanel

ADD performanceIndexField to performanceIndexPanel

SET preferred size of teacherIdLabel to 300 width and 50 height

SET preferred size of teacherIdField to 300 width and 50 height

SET preferred size of teacherNameLabel to 300 width and 50 height

SET preferred size of teacherNameField to 300 width and 50 height

SET preferred size of addressLabel to 300 width and 50 height

SET preferred size of addressField to 300 width and 50 height

SET preferred size of workingTypeLabel to 300 width and 50 height

SET preferred size of workingTypeField to 300 width and 50 height

SET preferred size of employmentStatusLabel to 300 width and 50 height

SET preferred size of employmentStatusField to 300 width and 50 height

SET preferred size of workingHoursLabel to 300 width and 50 height

SET preferred size of workingHoursField to 300 width and 50 height

SET preferred size of departmentLabel to 300 width and 50 height

SET preferred size of departmentField to 300 width and 50 height

SET preferred size of yearsOfExperienceLabel to 300 width and 50 height

SET preferred size of yearsOfExperienceField to 300 width and 50 height

SET preferred size of gradedScoreLabel to 300 width and 50 height

SET preferred size of gradedScoreField to 300 width and 50 height

SET preferred size of salaryLabel to 300 width and 50 height

SET preferred size of salaryField to 300 width and 50 height

SET preferred size of specializationLabel to 300 width and 50 height

SET preferred size of specializationField to 300 width and 50 height

SET preferred size of academicQualificationsLabel to 300 width and 50 height

SET preferred size of academicQualificationsField to 300 width and 50 height

SET preferred size of performanceIndexLabel to 300 width and 50 height

SET preferred size of performanceIndexField to 300 width and 50 height

SET opaque property of teacherIdPanel to false

SET opaque property of teacherNamePanel to false

SET opaque property of addressPanel to false

SET opaque property of workingTypePanel to false

SET opaque property of employmentStatusPanel to false

SET opaque property of workingHoursPanel to false

SET opaque property of departmentPanel to false

SET opaque property of yearsOfExperiencePanel to false

SET opaque property of gradedScorePanel to false

SET opaque property of salaryPanel to false

SET opaque property of specializationPanel to false

SET opaque property of academicQualificationsPanel to false

SET opaque property of performanceIndexPanel to false

SET opaque property of eastGap to false

SET opaque property of westGap to false

SET opaque property of southGap to false

SET opaque property of lecturerBody to false

SET opaque property of tutorBody to false

SET preferred size of body to 1463 width and 914 height

ADD directToLecturerPage to navBar

ADD directToTutorPage to navbar

ADD ActionListener to addLecturer button

METHOD actionPerformed(ActionEvent e)

IF teacherIdField.getText() is empty OR teacherNameField.getText() is empty OR addressField.getText() is empty OR workingTypeField.getText() is empty OR employmentStatusField.getText() is empty OR departmentField.getText() is empty OR workingHoursField.getText() is empty OR yearsOfExperienceField.getText() is empty THEN

DISPLAY "Please, fill out the empty fields!!!" message with "Empty Fields !!" title and WARNING_MESSAGE type

RETURN

ELSE

PARSE teacherIdField.getText() as integer and store it in teacherId variable

STORE teacherNameField.getText() in teacherName variable

STORE addressField.getText() in address variable

STORE workingTypeField.getText() in workingType variable

STORE employmentStatusField.getText() in employmentStatus variable

PARSE workingHoursField.getText() as double and store it in workingHours variable

STORE departmentField.getText() in department variable

PARSE yearsOfExperienceField.getText() as integer and store it in yearsOfExperience variable

IF teachers list is empty THEN

CREATE a new Lecturer object with teacherId, teacherName, address, workingType, employmentStatus, workingHours, department, yearsOfExperience

ADD the Lecturer object to teachers list

DISPLAY "Lecturer has been added successfully" message with "Lecture Added" title and INFORMATION_MESSAGE type

ELSE

FOR each Teacher object teacher in teachers list DO

IF teacher's teacherId is equal to teacherId THEN

IF teacher is an instance of Lecturer THEN

DISPLAY "A lecturer with the same ID has been detected!" message with "Duplication Error" title and WARNING_MESSAGE type

RETURN

ELSE

CREATE a new Lecturer object with teacherId, teacherName, address, workingType, employmentStatus, workingHours, department, yearsOfExperience

ADD the Lecturer object to teachers list

DISPLAY "Lecturer has been added successfully" message with "Lecture Added" title and INFORMATION_MESSAGE type

END METHOD

ADD ActionListener to addTutor button

METHOD actionPerformed(ActionEvent e)

IF teacherIdField.getText() is empty OR teacherNameField.getText() is empty OR addressField.getText() is empty OR workingTypeField.getText() is empty OR employmentStatusField.getText() is empty OR departmentField.getText() is empty OR workingHoursField.getText() is empty OR yearsOfExperienceField.getText() is empty OR salaryField.getText() is empty OR specializationField.getText() is empty OR academicQualificationsField.getText() is empty OR performanceIndexField.getText() is empty THEN

DISPLAY "Please, fill out the empty fields!!!" message with "Empty Fields !!" title and WARNING_MESSAGE type

RETURN

ELSE

PARSE teacherIdField.getText() as integer and store it in teacherId variable

STORE teacherNameField.getText() in teacherName variable

STORE addressField.getText() in address variable

STORE workingTypeField.getText() in workingType variable

STORE employmentStatusField.getText() in employmentStatus variable

PARSE workingHoursField.getText() as double and store it in workingHours variable

PARSE salaryField.getText() as double and store it in salary variable

STORE specializationField.getText() in specialization variable

STORE academicQualificationsField.getText() in academicQualifications variable

PARSE performanceIndexField.getText() as integer and store it in performanceIndex variable

IF teachers list is empty THEN

CREATE a new Tutor object with teacherId, teacherName, address, workingType, employmentStatus, workingHours, salary, specialization, academicQualifications, performanceIndex

ADD the Tutor object to teachers list

DISPLAY "Tutor has been added successfully" message with "Tutor Added" title and INFORMATION_MESSAGE type

ELSE

FOR each Teacher object teacher in teachers list DO

IF teacher's teacherId is equal to teacherId THEN

IF teacher is an instance of Tutor THEN

DISPLAY "A tutor with the same ID has been detected!" message with "Duplication Error" title and WARNING_MESSAGE type

RETURN

ELSE

CREATE a new Tutor object with teacherId, teacherName, address, workingType, employmentStatus, workingHours, salary, specialization, academicQualifications, performanceIndex

ADD the Tutor object to teachers list

DISPLAY "Tutor has been added successfully" message with "Tutor Added" title and INFORMATION_MESSAGE type

END METHOD

ADD ActionListener to gradeAssignments button

METHOD actionPerformed(ActionEvent e)

IF teacherIdField.getText() is empty OR teacherNameField.getText() is empty OR addressField.getText() is empty OR workingTypeField.getText() is empty OR employmentStatusField.getText() is empty OR departmentField.getText() is empty OR workingHoursField.getText() is empty OR yearsOfExperienceField.getText() is empty OR salaryField.getText() is empty OR specializationField.getText() is empty OR academicQualificationsField.getText() is empty OR performanceIndexField.getText() is empty THEN

DISPLAY "Please, fill out the empty fields!!!" message with "Empty Fields !!" title and WARNING_MESSAGE type

RETURN

ELSE

PARSE teacherIdField.getText() as integer and store it in teacherId variable

PARSE gradedScoreField.getText() as integer and store it in gradedScore variable

STORE departmentField.getText() in department variable

PARSE yearsOfExperienceField.getText() as integer and store it in yearsOfExperience variable

SET validityCheck to false

IF gradedScore is greater than 100 THEN

DISPLAY "Marking limit exceeded. Cannot be above 100!!" message with "Marking Error" title and WARNING_MESSAGE type

RETURN

ELSE IF gradedScore is less than 0 THEN

DISPLAY "Marking limit subceeded. Cannot be below 0!!" message with "Marking Error" title and WARNING_MESSAGE type

RETURN

ELSE IF yearsOfExperience is less than or equal to 5 THEN

DISPLAY "Experience Requirement not fulfilled. Must be 5 years or above!!" message with "Criteria Error" title and WARNING_MESSAGE type

RETURN

ELSE

FOR each Teacher object teacher in teachers list DO

IF teacher's teacherId is equal to teacherId THEN

IF teacher is an instance of Lecturer THEN

CAST teacher to Lecturer and store it in lecturer variable

CALL lecturer.gradeAssignment(gradedScore, department, yearsOfExperience)

IF lecturer has not graded THEN

DISPLAY "The assignment has not been graded. Error!!" message with "Grade Error!!" title and ERROR_MESSAGE type

ELSE

DISPLAY "Assignment graded successfully!! \nGrade --> lecturer's grade" message with "Marks graded" title and INFORMATION_MESSAGE type

END IF

END METHOD

ADD ActionListener to setSalary button

METHOD actionPerformed(ActionEvent e)

IF teacherIdField.getText() is empty OR salaryField.getText() is empty OR performanceIndexField.getText() is empty THEN

DISPLAY "Please, fill out the empty fields!!!" message with "Empty Fields!!" title and WARNING_MESSAGE type

RETURN

ELSE

PARSE teacherIdField.getText() as integer and store it in teacherId variable

PARSE salaryField.getText() as double and store it in salary variable

PARSE performanceIndexField.getText() as integer and store it in performanceIndex variable

IF performanceIndex is greater than 10 THEN

DISPLAY "Performance Index limit exceeded. Cannot be above 10!!" message with "Index Error" title and WARNING_MESSAGE type

RETURN

ELSE IF performanceIndex is less than 0 THEN

DISPLAY "Performance Index limit subceeded. Cannot be below 0!!" message with "Index Error" title and WARNING_MESSAGE type

RETURN

ELSE

FOR each Teacher object teacher in teachers list DO

IF teacher's teacherId is equal to teacherId THEN

IF teacher is an instance of Tutor THEN

CAST teacher to Tutor and store it in tutor variable

CALL tutor.setSalary(salary, performanceIndex)

SET salary to tutor's salary

IF performanceIndex is less than 5 AND tutor's workingHours is less than 20 THEN

DISPLAY "New Salary Requirements not fulfilled.\nPerformance Index must be above 5 or Working Hours must be above 20 hours!!" message with "Salary Not Approved" title and INFORMATION_MESSAGE type

ELSE

DISPLAY "Salary set successfull with salary being salary!" message with "Setting Salary" title and INFORMATION_MESSAGE type

END METHOD

ADD ActionListener to removeTutor button

METHOD actionPerformed(ActionEvent e)

IF teacherIdField.getText() is empty THEN

DISPLAY "Please, fill out the empty Teacher Id field!!!" message with "Empty Fields!!" title and WARNING_MESSAGE type

RETURN

ELSE

PARSE teacherIdField.getText() as integer and store it in teacherId variable

FOR each Teacher object teacher in teachers list DO

IF teacher's teacherId is equal to teacherId THEN

IF teacher is an instance of Tutor THEN

CAST teacher to Tutor and store it in tutor variable

STORE tutor.isCertified() in isCertified variable

IF isCertified is false THEN

CALL tutor.removeTutor()

DISPLAY "The tutor has been successfully removed" message with "Tutor Removed" title and INFORMATION_MESSAGE type

ELSE

DISPLAY "Certified tutors cannot be removed" message with "Remove Tutor Error" title and INFORMATION_MESSAGE type

END METHOD

ADD ActionListener to displayLecturer button

METHOD actionPerformed(ActionEvent e)

PARSE teacherIdField.getText() as integer and store it in teacherId variable

STORE teacherNameField.getText() in teacherName variable

STORE addressField.getText() in address variable

STORE workingTypeField.getText() in workingType variable

STORE employmentStatusField.getText() in employmentStatus variable

PARSE workingHoursField.getText() as integer and store it in workingHours variable

STORE departmentField.getText() in department variable

PARSE yearsOfExperienceField.getText() as integer and store it in yearsOfExperience variable

DISPLAY "Teacher ID: teacherId" concatenated with "\nTeacher Name: teacherName" concatenated with "\nAddress: address" concatenated with "\nWorking Type: workingType" concatenated with "\nEmployment Status: employmentStatus" concatenated with "\nWorking Hours: workingHours" concatenated with "\nDepartment: department" concatenated with "\nYears of Experience: yearsOfExperience" message

END METHOD

ADD ActionListener to displayTutor button

METHOD actionPerformed(ActionEvent e)

PARSE teacherIdField.getText() as integer and store it in teacherId variable

STORE teacherNameField.getText() in teacherName variable

STORE addressField.getText() in address variable

STORE workingTypeField.getText() in workingType variable

STORE employmentStatusField.getText() in employmentStatus variable

PARSE workingHoursField.getText() as integer and store it in workingHours variable

PARSE salaryField.getText() as double and store it in salary variable

STORE specializationField.getText() in specialization variable

STORE academicQualificationsField.getText() in academicQualifications variable

PARSE performanceIndexField.getText() as integer and store it in performanceIndex variable

DISPLAY "Teacher ID: teacherId" concatenated with "\nTeacher Name: teacherName" concatenated with "\nAddress: address" concatenated with "\nWorking Type: workingType" concatenated with "\nEmployment Status: employmentStatus" concatenated with "\nWorking Hours: workingHours" concatenated with "\nSalary: salary" concatenated with "\nSpecialization: specialization" concatenated with "\nAcademic Qualification: academicQualifications" concatenated with "\nPerformance Index: performanceIndex" message

END METHOD

ADD ActionListener to clear button

METHOD actionPerformed(ActionEvent e)

// Clearing the text fields

SET teacherIdField's text to empty string

SET teacherNameField's text to empty string

SET addressField's text to empty string

```
        SET workingTypeField's text to empty string
        SET employmentStatusField's text to empty string
        SET workingHoursField's text to empty string
        SET departmentField's text to empty string
        SET yearsOfExperienceField's text to empty string
        SET gradedScoreField's text to empty string
        SET salaryField's text to empty string
        SET specializationField's text to empty string
        SET academicQualificationsField's text to empty string
        SET performanceIndexField's text to empty string
    END METHOD
END CLASS
```

CHAPTER IV: METHODS

4.1. Method

Methods in java can be defined as the block of code which performs a specific task. These methods run only when the method is called. There are 4 types of methods classified on the basis of its return type and the parameters passed to the method (Pokhrel, 2024). The methods are listed below:

- Methods with no returning value and no parameter passed
- Methods with no returning value but parameter passed
- Methods with returning value but without passing parameter
- Methods with returning value when passing parameter

4.2. Methods of TeacherGUI Class

Methods	Method's Description
ActionListener for welcomeToTutor	welcomeToTutor() method is an ActionListener attached to the "Tutor" button in the welcome page. When triggered, it switches the GUI from the welcome page to the tutor section, hiding the welcome page and displaying the tutor section with its respective components. It sets the frame to be maximized and adjusts the layout accordingly to accommodate the tutor panel, including navigation and buttons. The method configures the appearance and functionality of the tutor section, setting up text fields, buttons, and action listeners for further interaction.
ActionListener for welcomeToLecturer	welcomeToLecturer() method serves as an ActionListener linked to the "Lecturer" button in the welcome page. Upon activation, it transitions the GUI from the welcome page to the lecturer section, hiding the welcome page and revealing the lecturer section with its associated components. The method maximizes the frame and organizes the layout to incorporate the lecturer panel, navigation, and buttons. It initializes and configures text fields, buttons, and action listeners to facilitate user interaction within the lecturer section.

ActionListener for directToTutor	directToTutor() method is an ActionListener connected to the "Tutor" button, allowing direct navigation to the tutor section from any part of the GUI. Upon invocation, it switches the GUI to the tutor section, hiding other sections if visible, and displaying the tutor section with its elements. The method adjusts the layout and appearance to accommodate the tutor panel, navigation, and buttons, ensuring seamless transition and user experience.
ActionListener for directToLecturer	directToLecturer() method functions as an ActionListener associated with the "Lecturer" button, enabling immediate access to the lecturer section from any part of the GUI. When activated, it transitions the GUI to the lecturer section, hiding other sections if visible, and presenting the lecturer section with its components. The method manages layout adjustments and visual settings to integrate the lecturer panel, navigation, and buttons smoothly, ensuring user-friendly navigation.
ActionListener for addLecturer	addLecturer() method is an ActionListener linked to the "Add a Lecturer" button in the lecturer section. When triggered, it initiates the process of adding a new lecturer to the system by capturing input data from text fields and validating it. The method retrieves and validates information such as teacher ID, name, address, employment status, working hours, department, years of experience, etc., and creates a new lecturer object with the provided data. It then adds the lecturer to the system and updates the display accordingly.
ActionListener for gradeAssignments	gradeAssignments() method serves as an ActionListener attached to the "Grade the Assignments" button in the lecturer section. Upon activation, it facilitates the grading process for assignments submitted by students. The method retrieves relevant data such as student submissions, assignment details, and grading criteria. It presents the necessary interface for assigning grades, calculating scores, and updating student records accordingly. Additionally, it may include functionalities for providing feedback and tracking grading progress.

ActionListener for setSalary	setSalary() method functions as an ActionListener associated with the "Set the salary of Tutor" button in the tutor section. When invoked, it allows administrators or authorized users to set the salary for tutors within the educational institution. The method prompts for input of necessary details such as tutor ID and salary amount, verifies the data, and updates the salary information in the system. It ensures accurate salary management and maintains records of salary adjustments.
ActionListener for addTutor	addTutor() method is an ActionListener connected to the "Add a Tutor" button in the tutor section. Upon activation, it initiates the process of adding a new tutor to the system by collecting and validating input data from text fields. The method captures essential information such as tutor ID, name, address, employment status, specialization, working hours, academic qualifications, etc., and creates a new tutor object with the provided details. It then adds the tutor to the system and updates the display accordingly.
ActionListener for displayTutor	displayTutor() method serves as an ActionListener linked to the "Display" button in the tutor section. When triggered, it retrieves and displays information about tutors currently registered in the system. The method fetches data such as tutor details, specialization, working hours, salary, etc., and presents it in a user-friendly format for easy viewing. It ensures transparency and accessibility of tutor information for administrators or users requiring access.
ActionListener for displayLecturer	displayLecturer() method functions as an ActionListener associated with the "Display" button in the lecturer section. Upon activation, it retrieves and displays information regarding lecturers currently registered in the system. The method accesses data such as lecturer details, employment status, department, years of experience, salary, etc., and presents it in a comprehensible format for administrators or users. It ensures visibility and accessibility of lecturer information as needed.

ActionListener for clear	clear() method is an ActionListener connected to the "Clear" button in both the lecturer and tutor sections. When invoked, it clears all input fields and resets any temporary data or selections made within the respective section. The method facilitates user interaction by providing a convenient way to reset input fields and start anew, ensuring a clean slate for entering fresh data or performing new actions within the section.
ActionListener for removeTutor	removeTutor() method serves as an ActionListener attached to the "Remove the Tutor" button in the tutor section. Upon activation, it initiates the process of removing a tutor from the system based on user input or selection. The method prompts for confirmation and verifies the tutor's identity or relevant details before proceeding with the removal process. It ensures secure and accurate removal of tutors from the system, updating records and display accordingly.

Table 1: Method Descriptions for TeacherGUI Class

CHAPTER V: TESTINGS

5.1. Test 1 – Compile and run using the command prompt

Test No.:	1
Objective:	To compile and run the program using command prompt.
Action:	i) Navigate to folder location ii) Open command prompt iii) Type and enter the code to compile and run TeacherGUI.java file
Expected Result:	The TeacherGUI Class is expected to be compiled and run without any errors
Actual Result:	The TeacherGUI Class was successfully compiled and run with zero erros.
Conclusion	The test was completed with the expected result successfully.

Table 2: Test 1

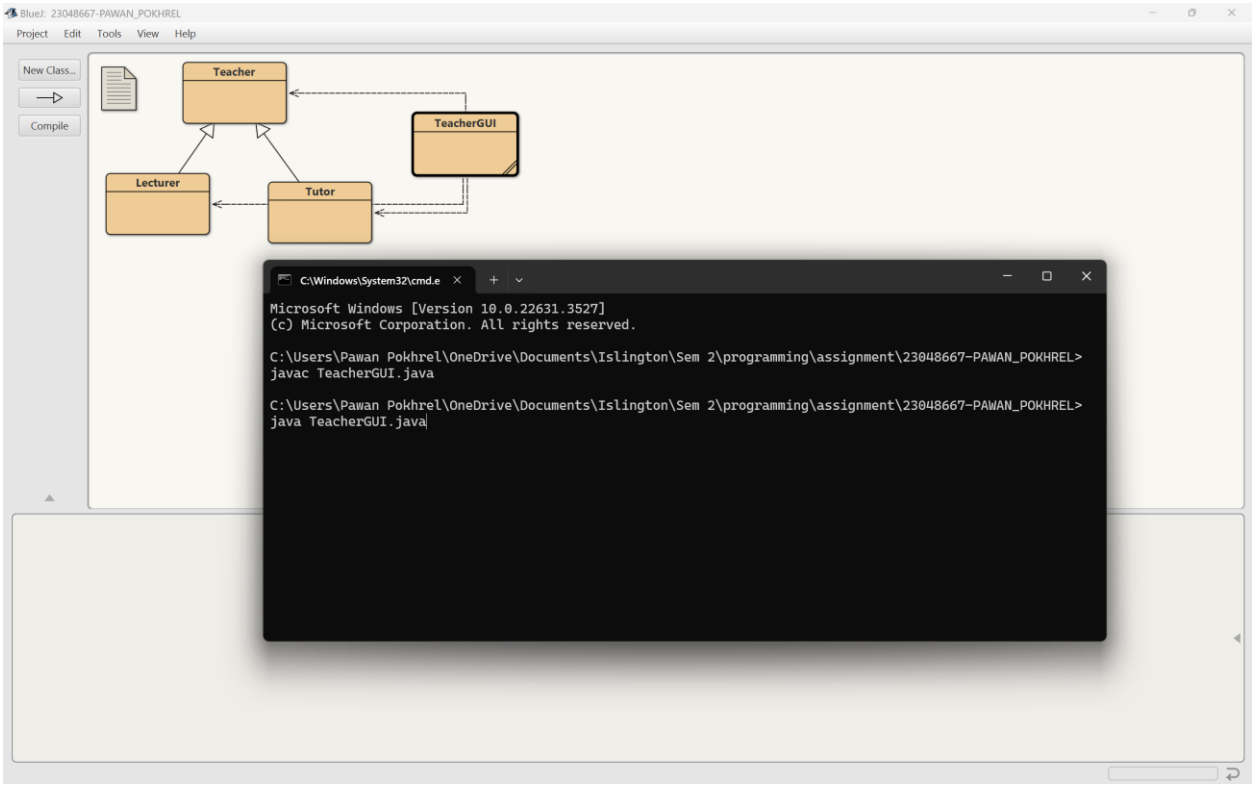


Figure 7: Screenshot of the command prompt while compiling and running the code through cmd

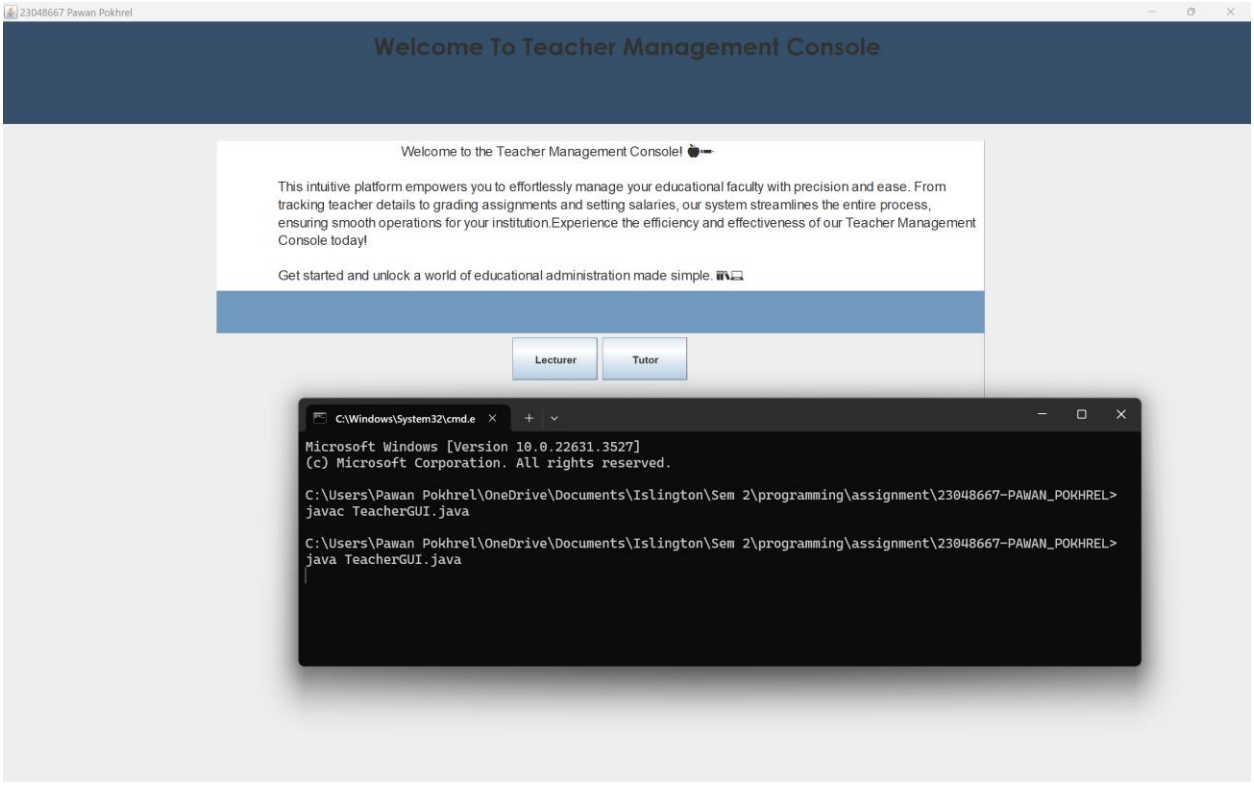


Figure 8: Screenshot of the GUI when compiled and run through the command prompt

5.2. Test 2 - Test for the functionality of addLecturer, addTutor, gradeAssignments, setSalary, removeTutor

Test No.:	2
Objective:	To test the functionality of the JButtons in the program
Action:	i) Add a Lecturer with all the details ii) Add a Tutor with all the details iii) Perform necessary steps to trigger the gradeAssignments JButton iv) Perform necessary steps to trigger the setSalary JButton v) Remove the added tutor
Expected Result:	The TeacherGUI Class is expected to run these functionalities without any errors and handling the exceptions.
Actual Result:	The TeacherGUI Class successfully operated all the functions in the program.
Conclusion	The test was completed with the expected result successfully.

Table 3: Test 2

Step i)

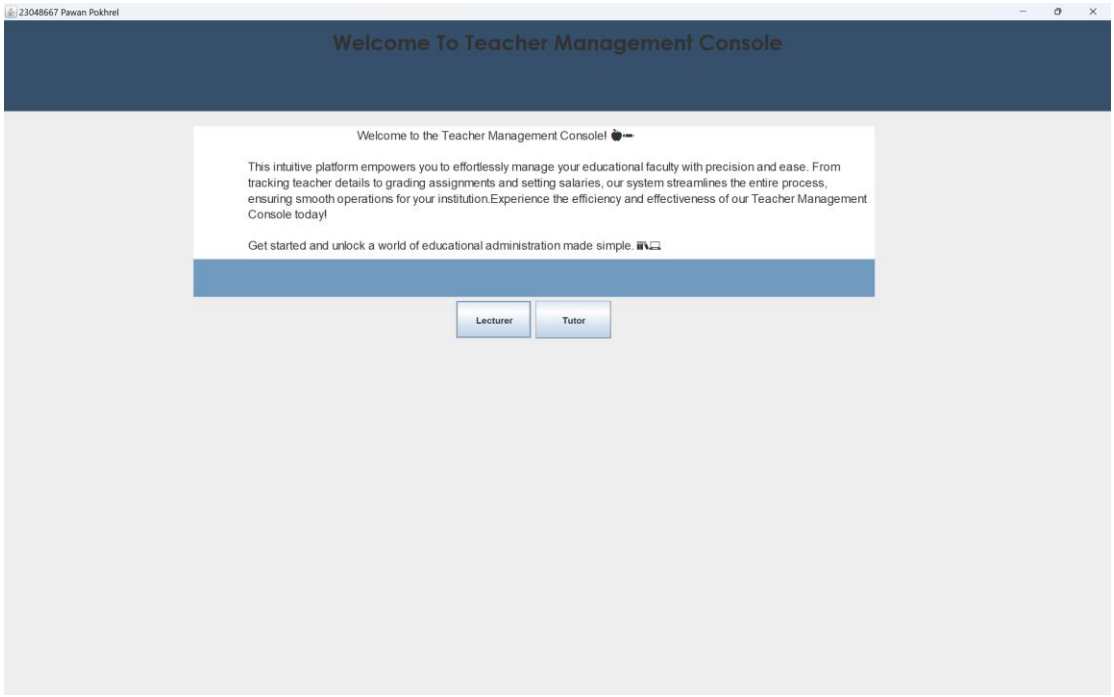


Figure 9: Screenshot of the HomePage of the Program

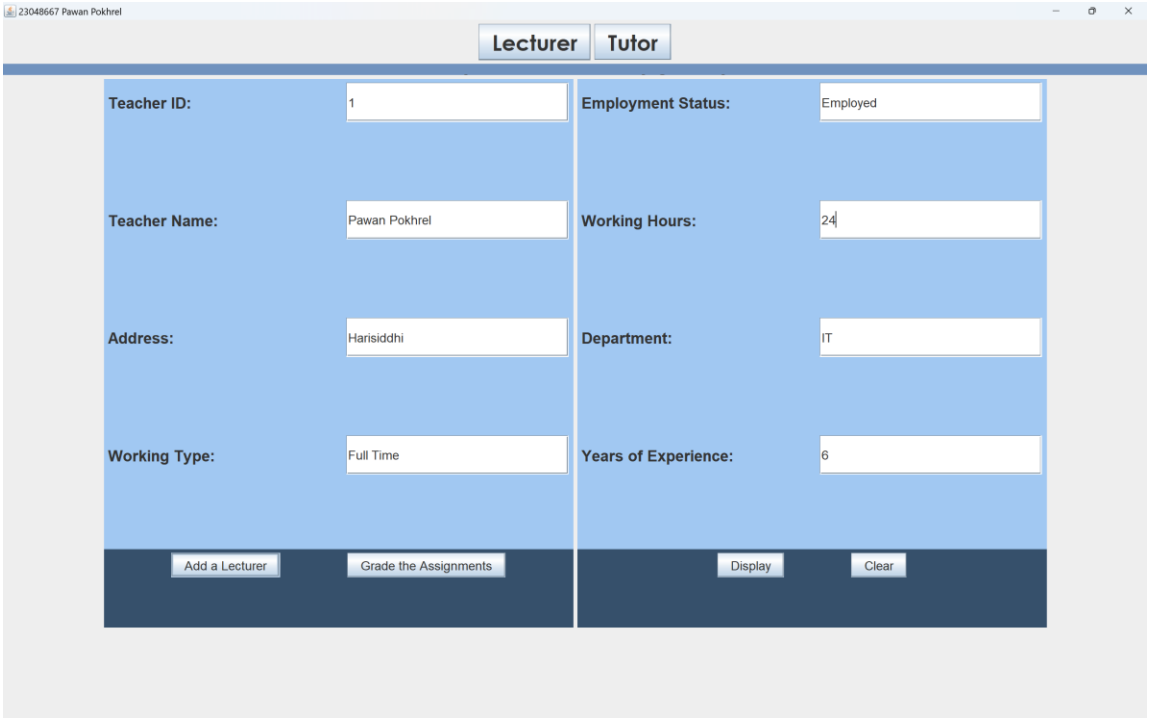


Figure 10: Screenshot of the process during adding of Lecturer

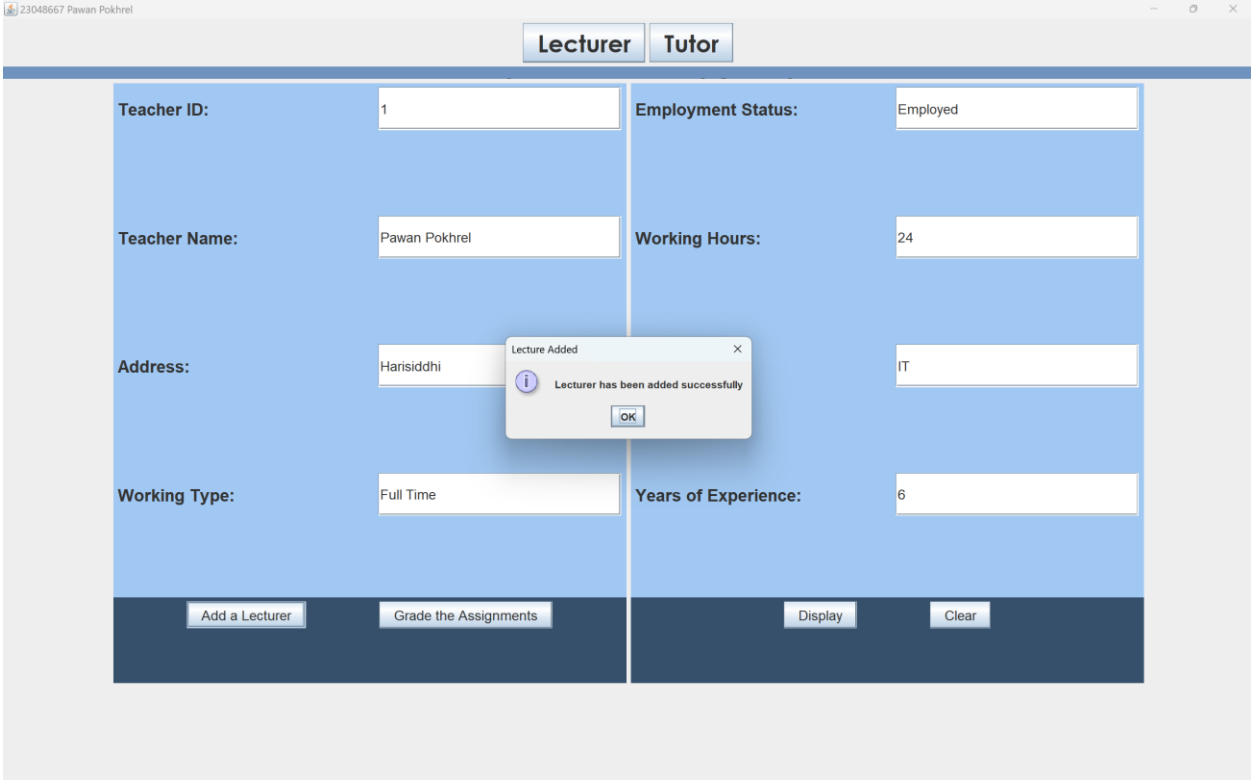


Figure 11: Screenshot of the successful addition of Lecturer

Step ii)

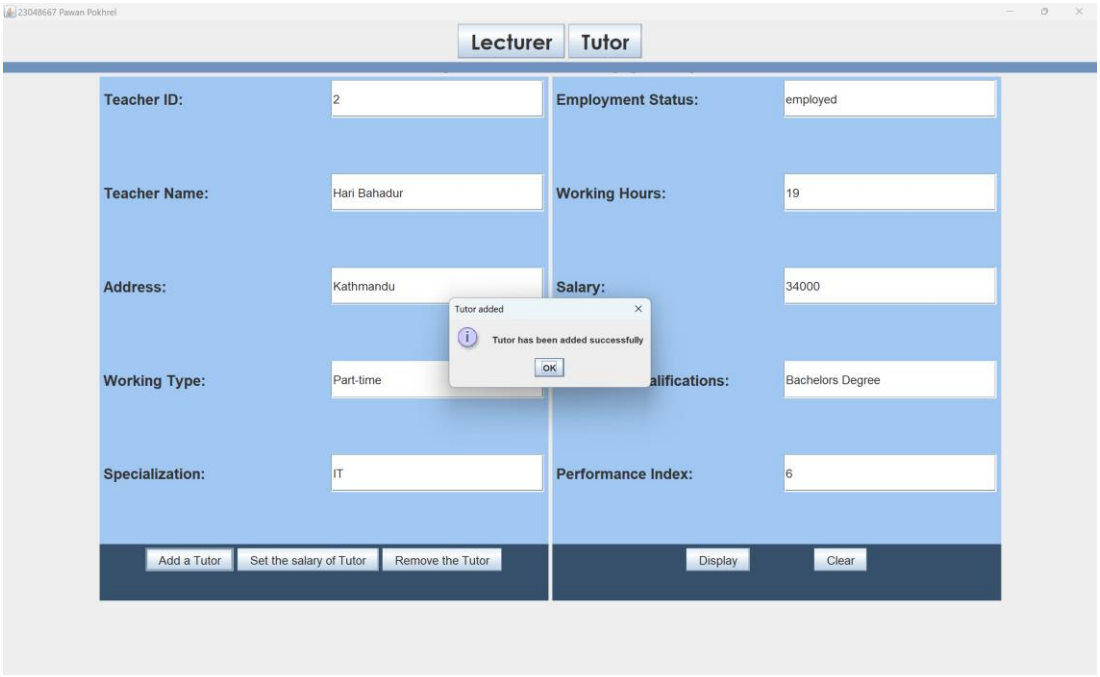


Figure 12: Screenshot of the program during the process of add Tutor

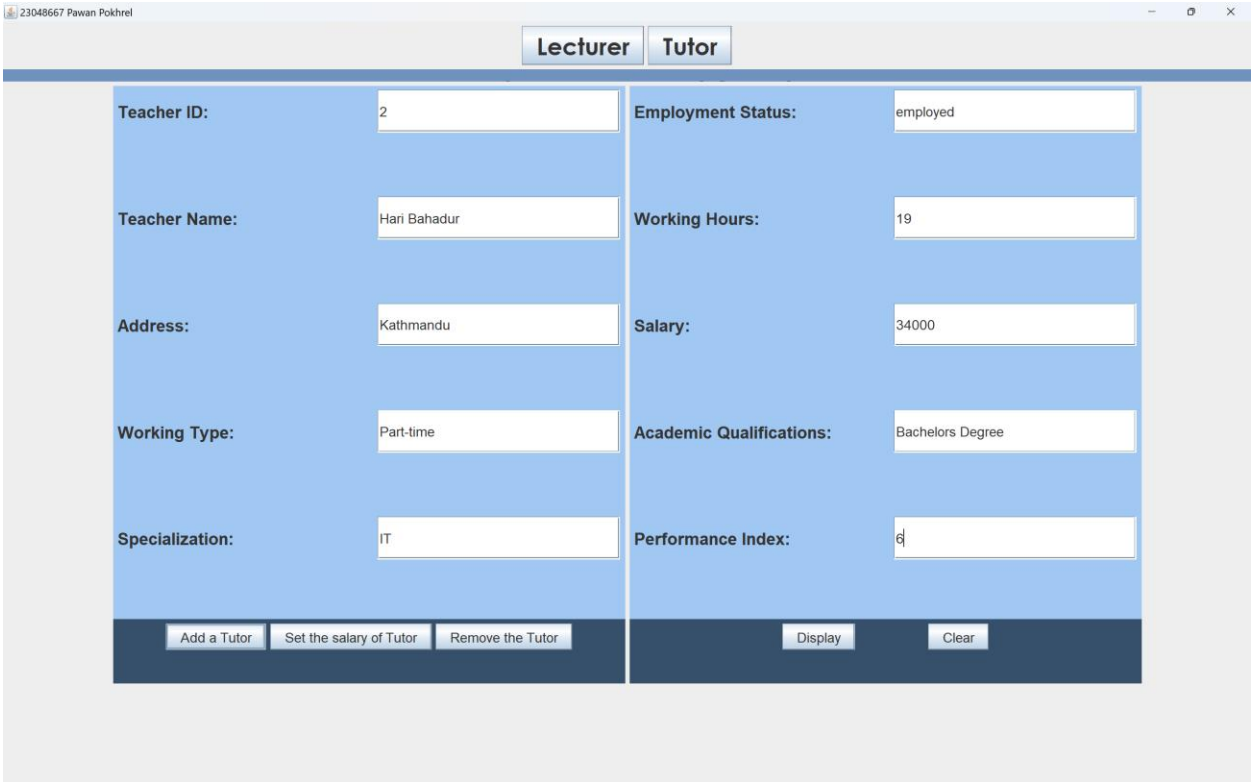


Figure 14: Screenshot of the successful addition of Tutor

Step iii)

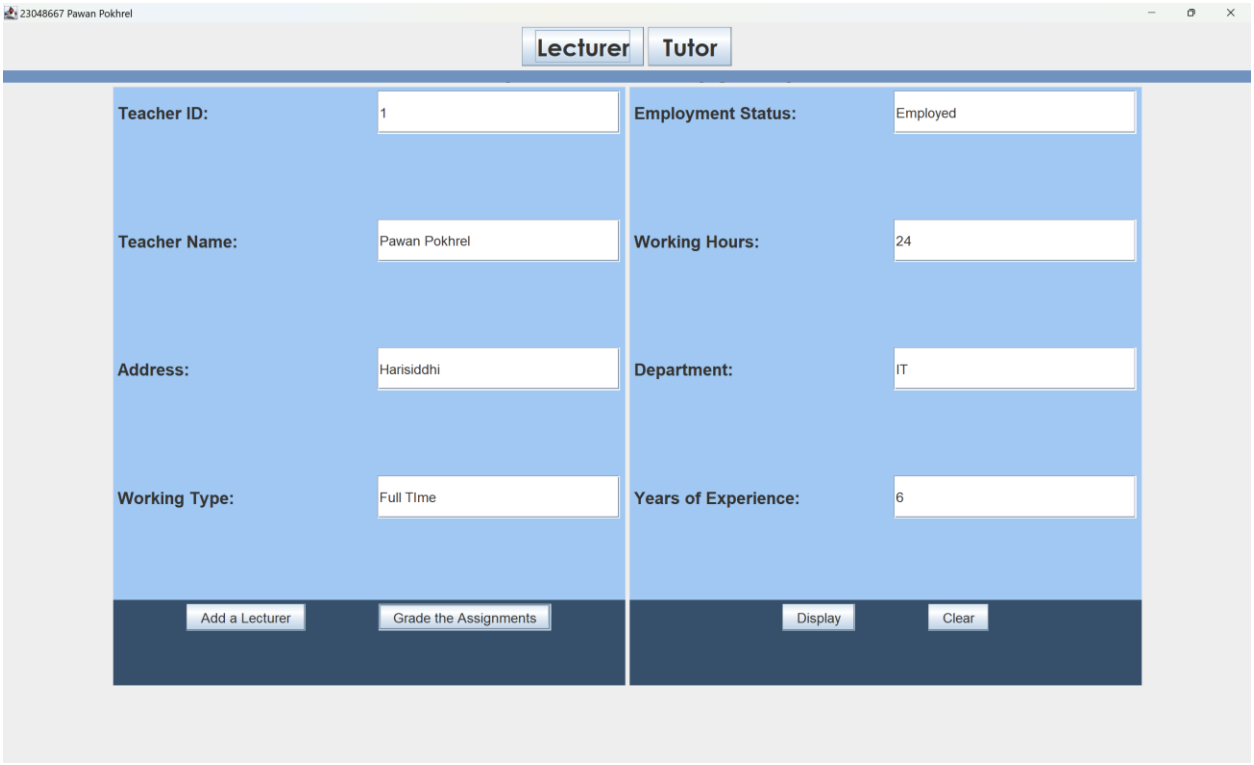


Figure 13: Screenshot of the program to go to the grade assignment section from add lecturer section

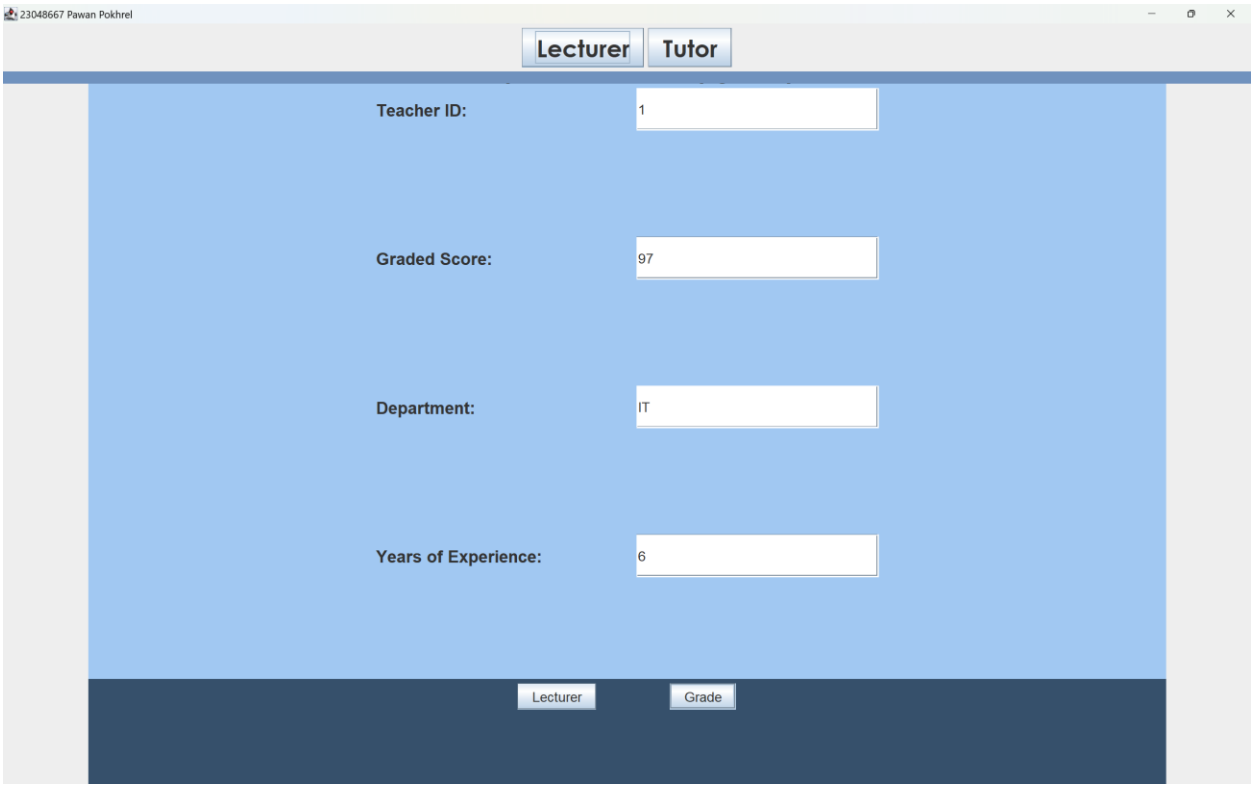


Figure 15: Screen of the grade assignment section

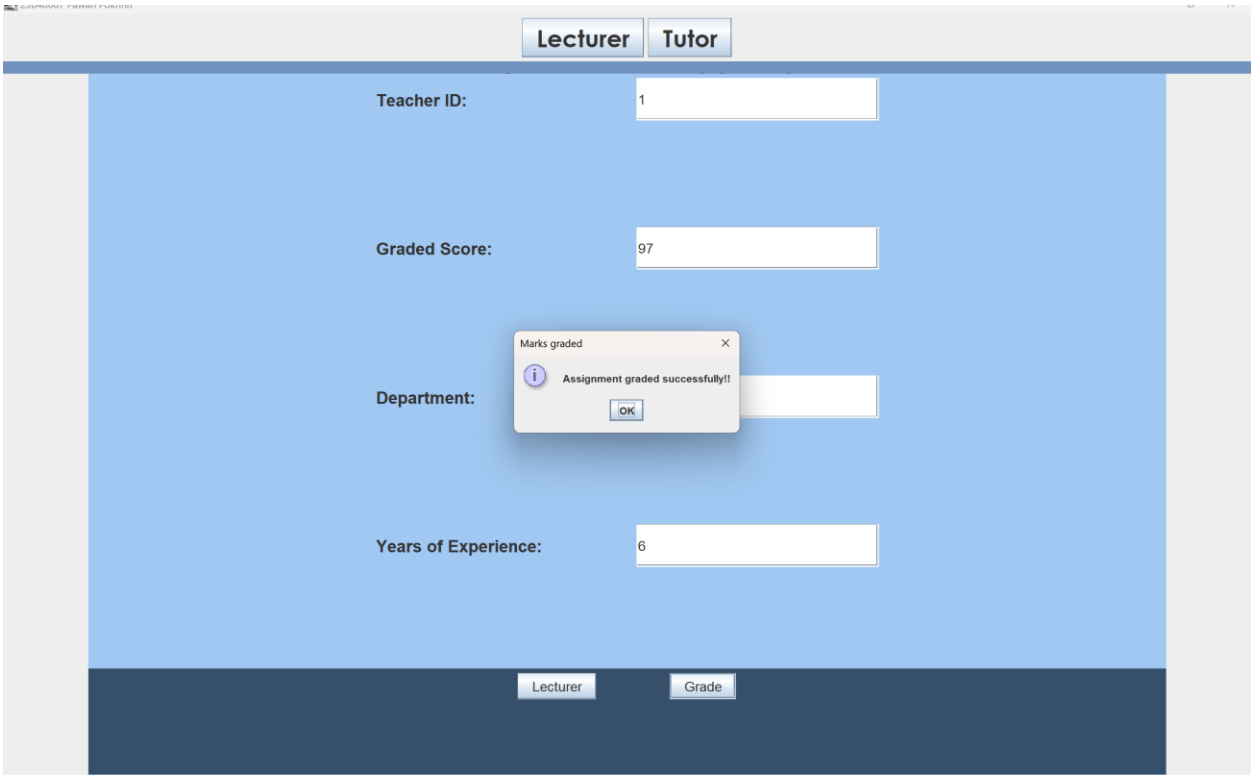


Figure 16: Screenshot of the successful grading of the assignment

Step iv)

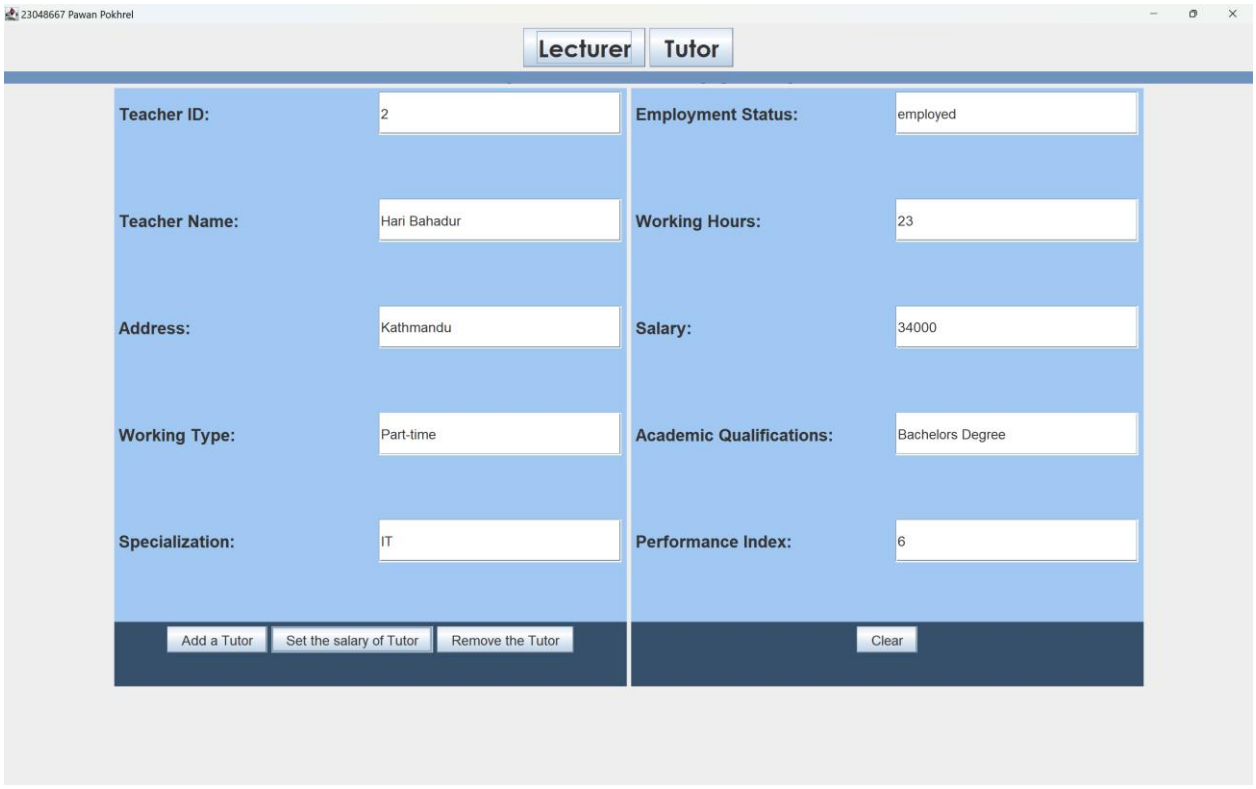


Figure 18: Screenshot of the add tutor section

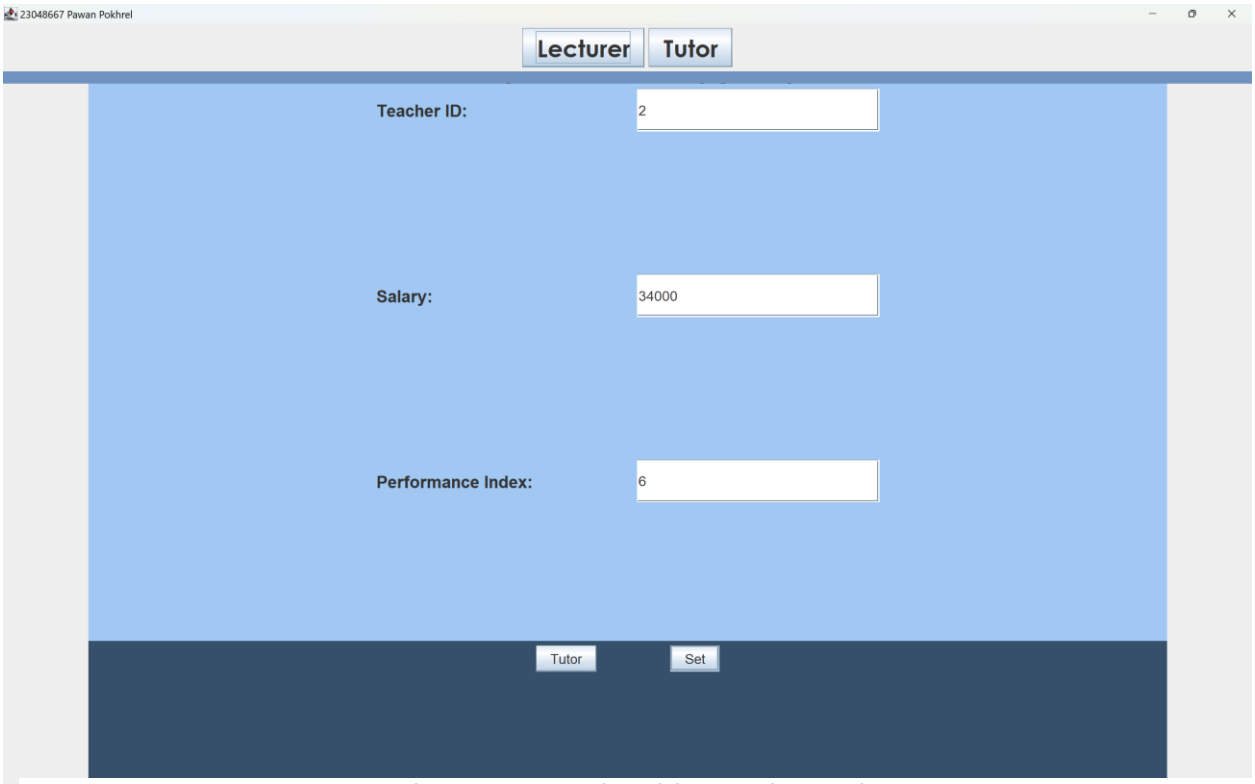


Figure 17: Screenshot of the setSalary section

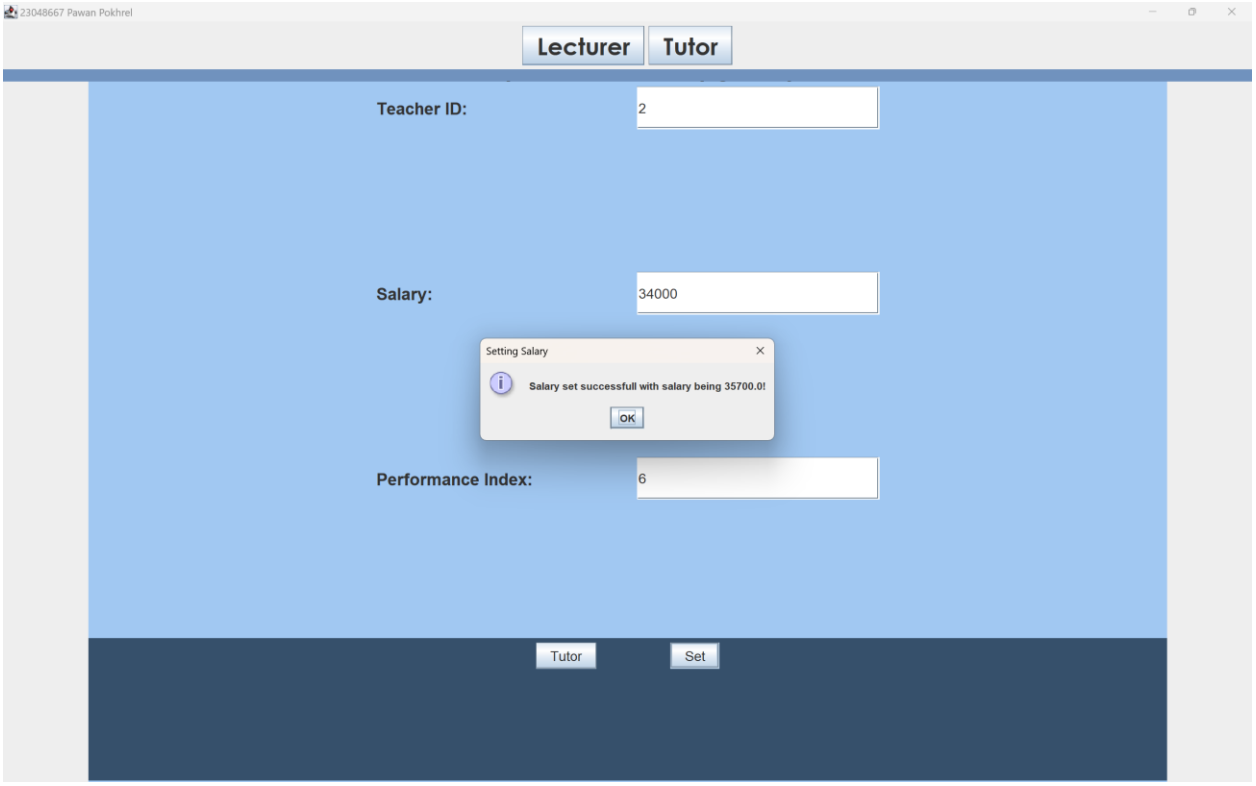


Figure 19: Screenshot after the successful setting of the salary

Step v)

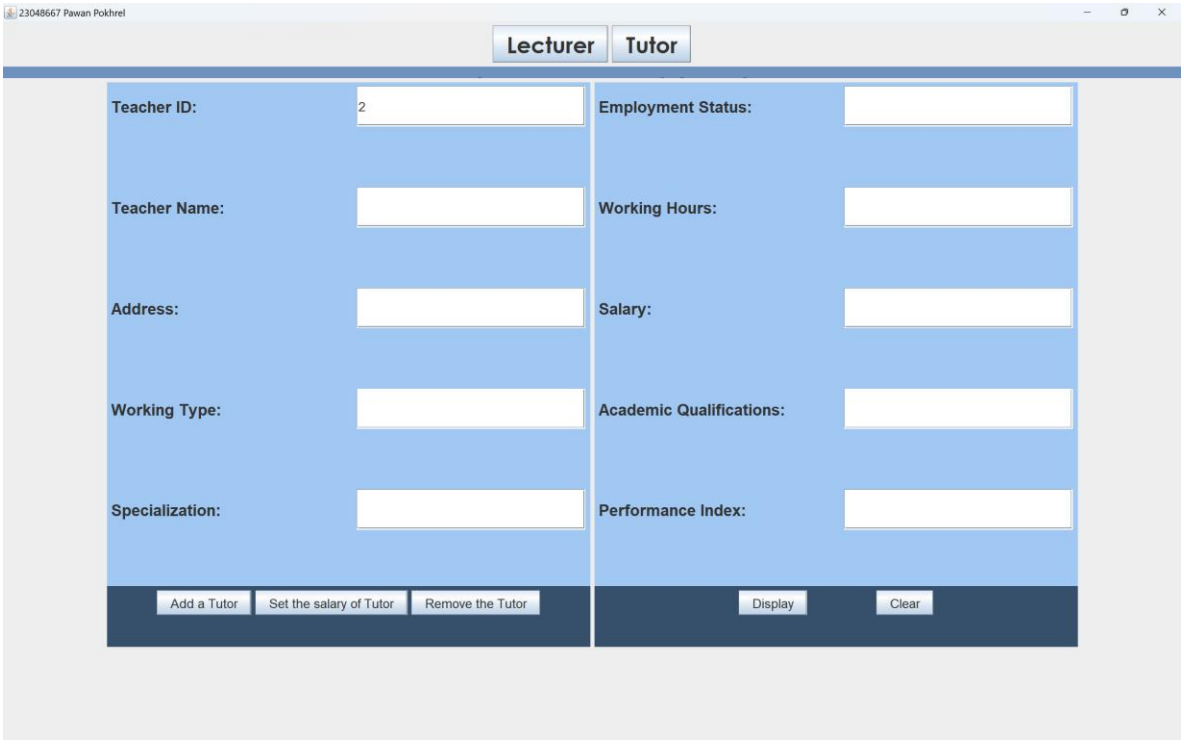


Figure 20: Screenshot of the process to remove Tutor

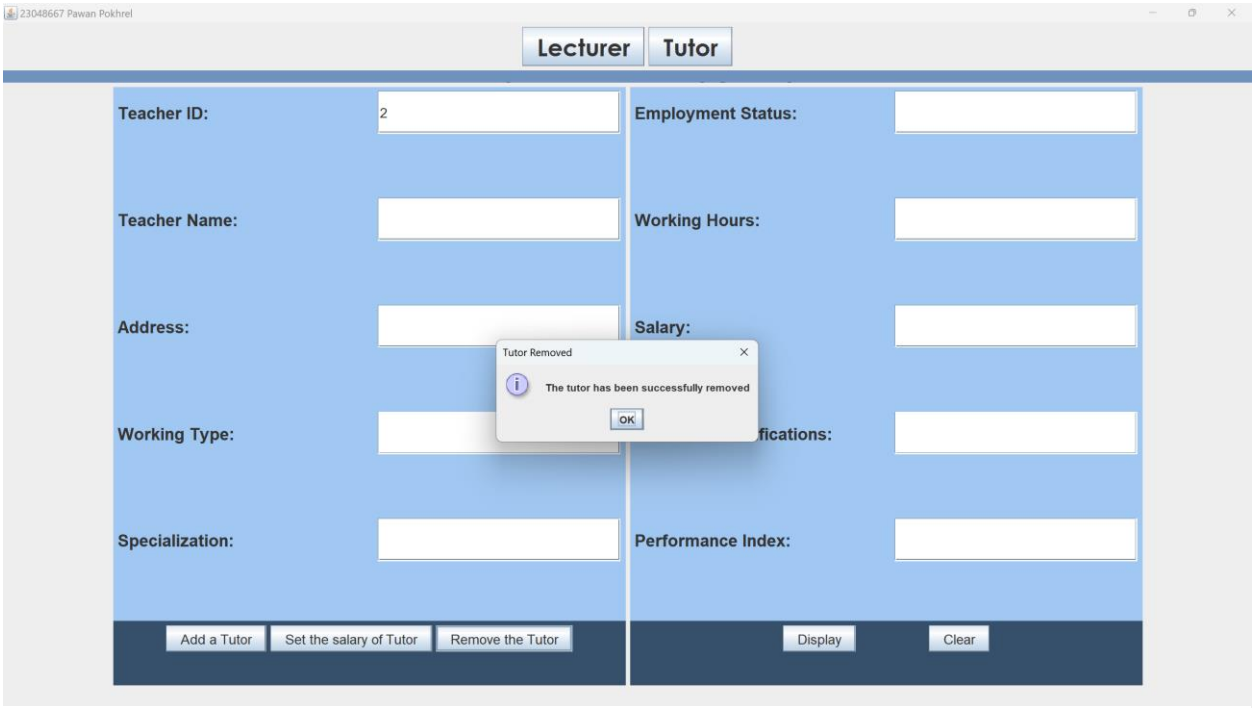


Figure 21: Screenshot of the sucessful removing of Tutor object

5.3. Test 3 – Test for the invalid inputs message display

Test No.:	3
Objective:	To test the appropriate display of the dialog boxes when wrong inputs are provided.
Action:	i) Enter a non-numeric value in the techerIdField.
Expected Result:	The TeacherGUI Class is expected to show a message dialog box with an error message.
Actual Result:	The TeacherGUI Class successfully displayed an error message in the dialog box.
Conclusion	The test was completed with the expected result successfully.

Table 4: Test 3

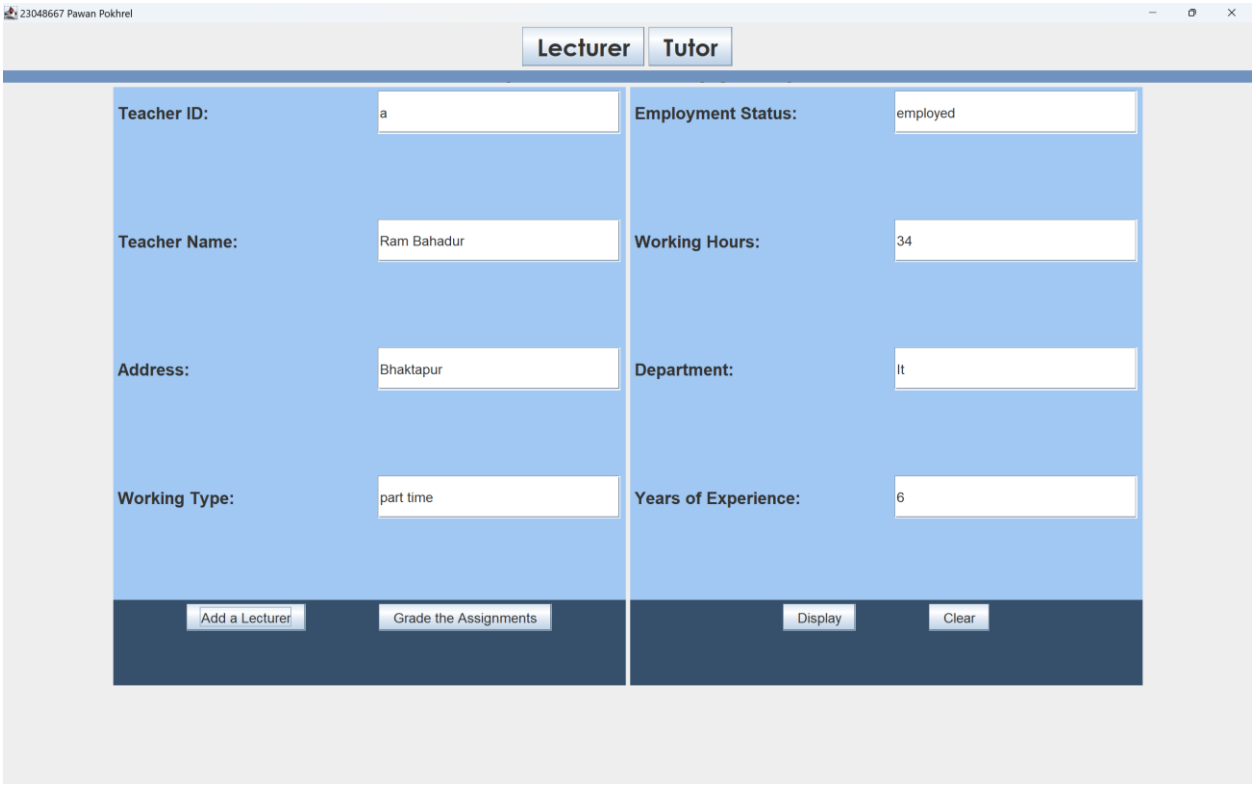


Figure 22: Screenshot of an invalid input into the teacherId

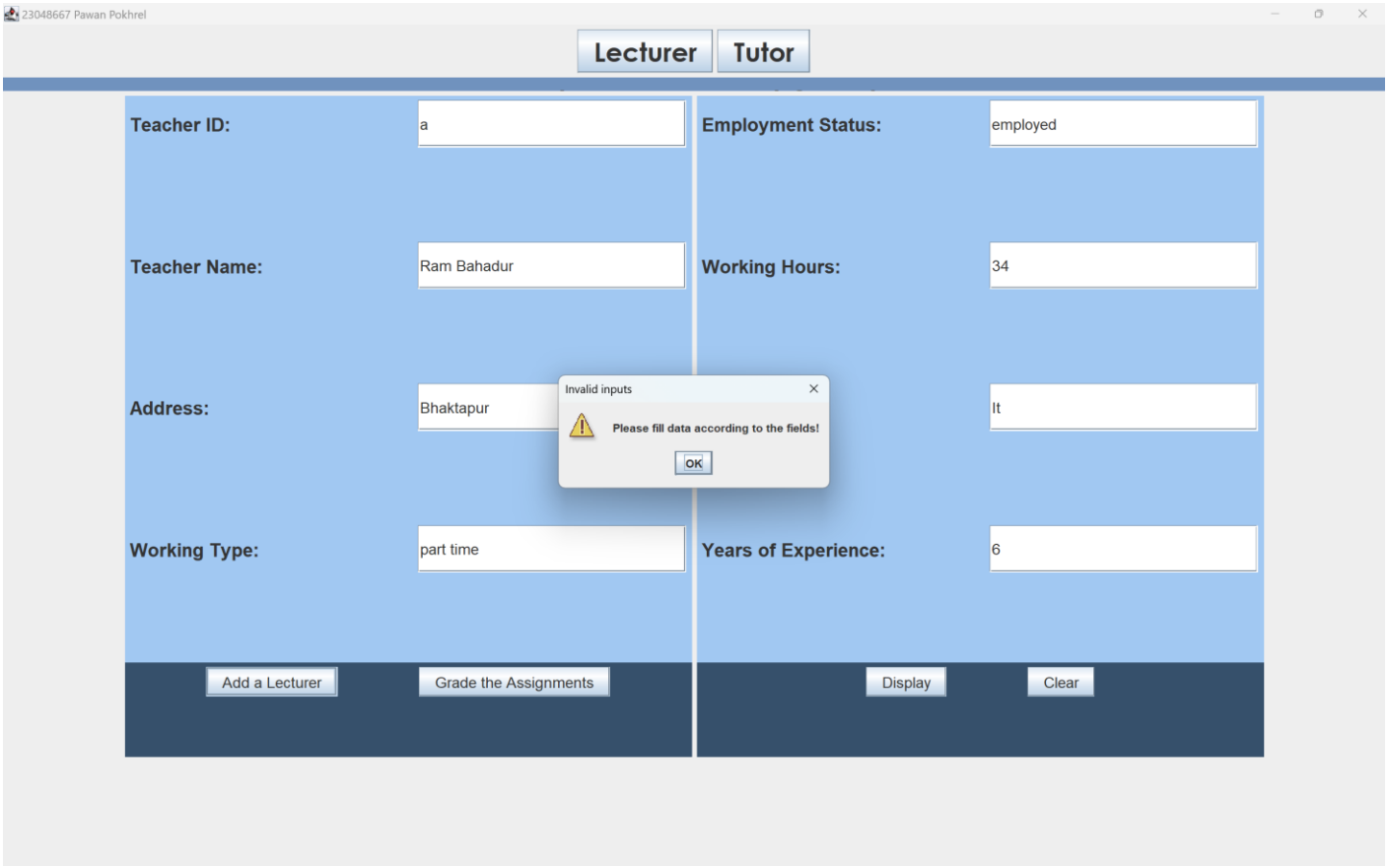
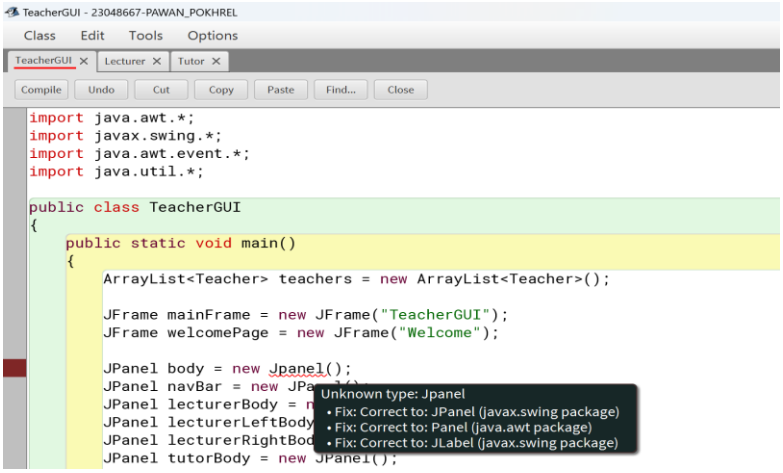


Figure 23: Screenshot of a message dialog regarding the error

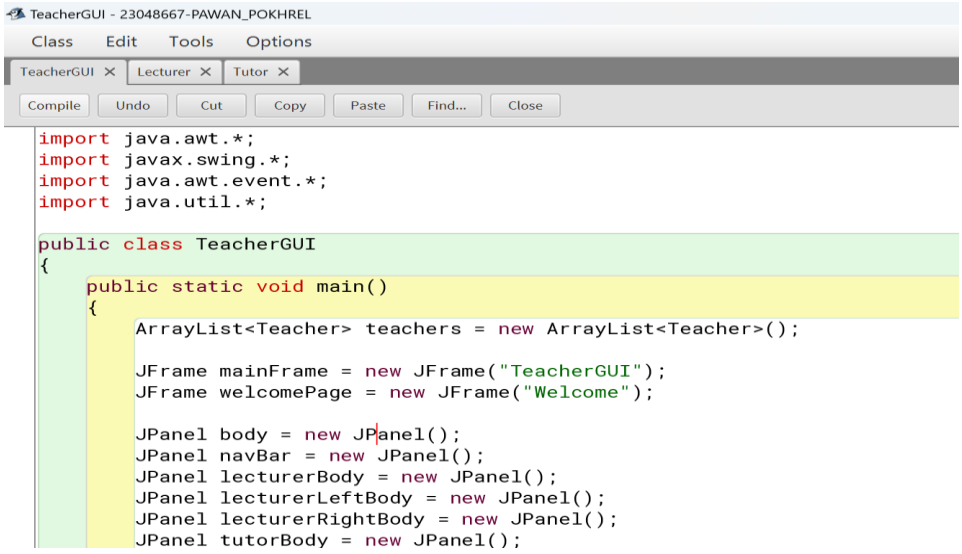
CHAPTER VI: ERROR DETECTION AND CORRECTION

6.1. Syntax Error

Syntax errors are the errors in the source code, like an incorrectly spelled instruction mnemonic or a label that isn't declared before it's used in the program (Bates, 2011). These errors are detected by the compilers of the programming language. On the process of completion of the Coursework, some errors were detected:



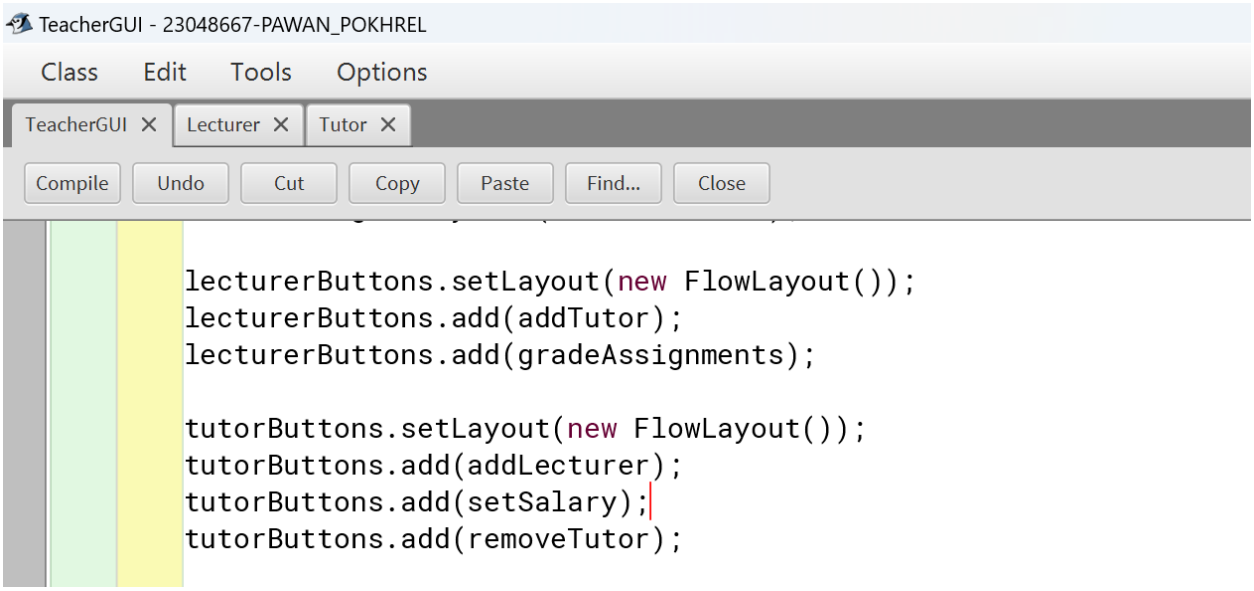
During the time of development, the name of the class JPanel was misspelled as Jpanel which caused an syntax error in the code.



This error was fixed simply by correcting the Jpanel to JPanel.

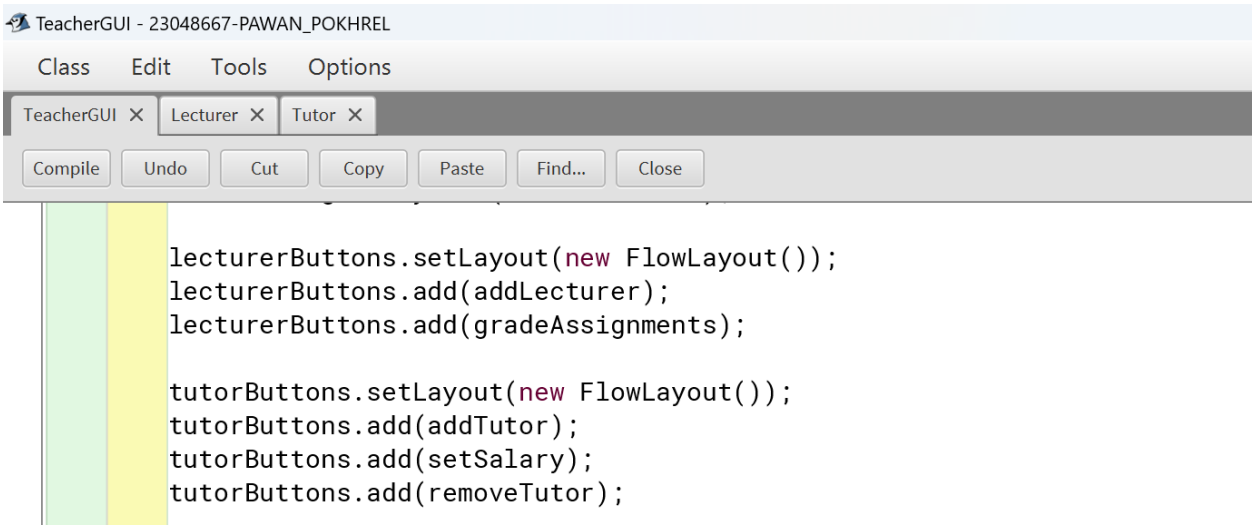
6.2. Semantic Error

Semantic Errors are the errors in the programming which are detected during run time. These errors arise when there is a flaw in the logic or meaning of the code, as opposed to syntax errors, which happen when the code breaks the rules of the programming language and usually prevents the code from compiling. Because of this, the programme might execute, but it won't behave or generate the intended results. The semantic error detected during the completion of my coursework is:



When I ran my TeacherGUI, the “Add a Tutor” button was mistakenly added to the lecturer section and vice versa.

This error was fixed by adding the buttons to their respective panels or sections.



6.3. Logical Error

A logic error is a programming error that occurs when a computer's programming code is programmed or compiled but is not immediately obvious (Woodruff, 2023). A logic error is a problem in a program that results in unexpected behaviour; unlike syntax errors, logic errors are caused by faulty reasoning. Even when the code executes without a hitch, it's the quiet culprit responsible for inaccurate outcomes. Deciphering these mistakes calls for an acute attention to detail and a thorough comprehension of the desired results (Woodruff, 2023). The logical error detected during the development of my coursework is:

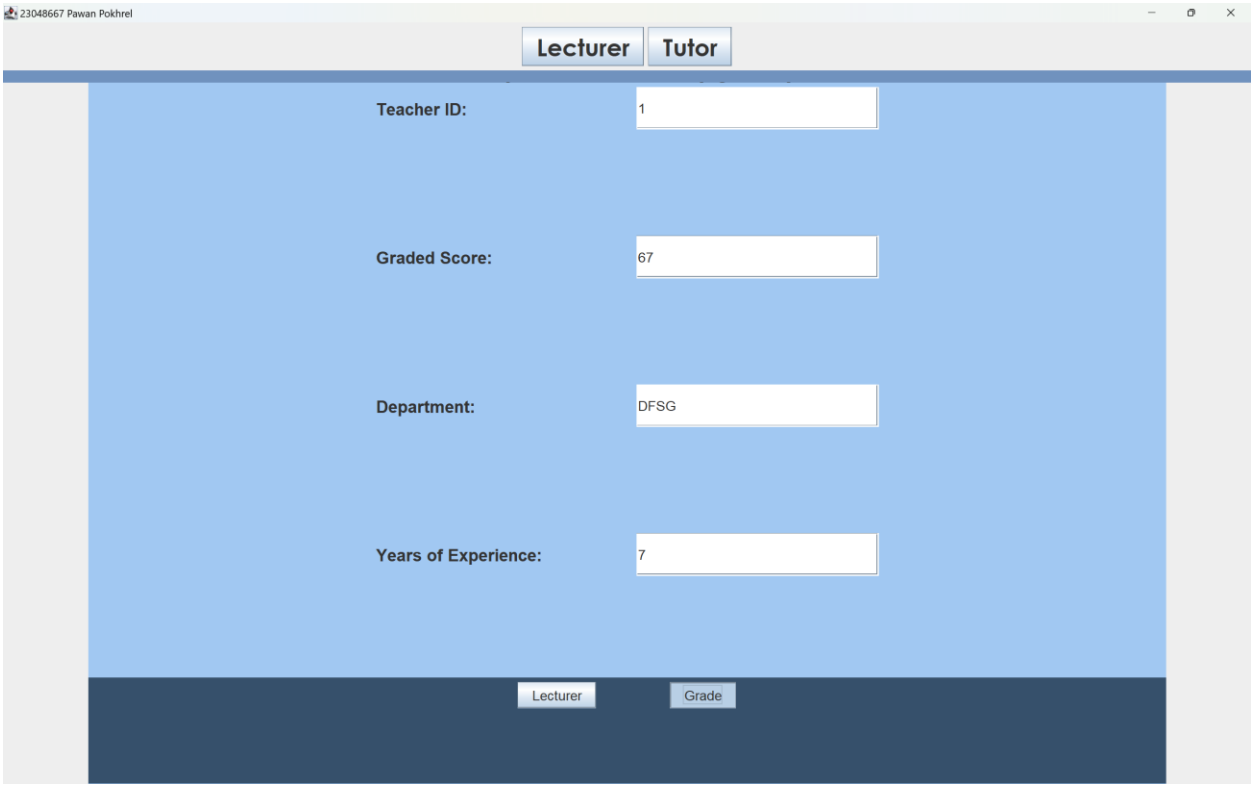


Figure 24: No output when clicking on the grade button

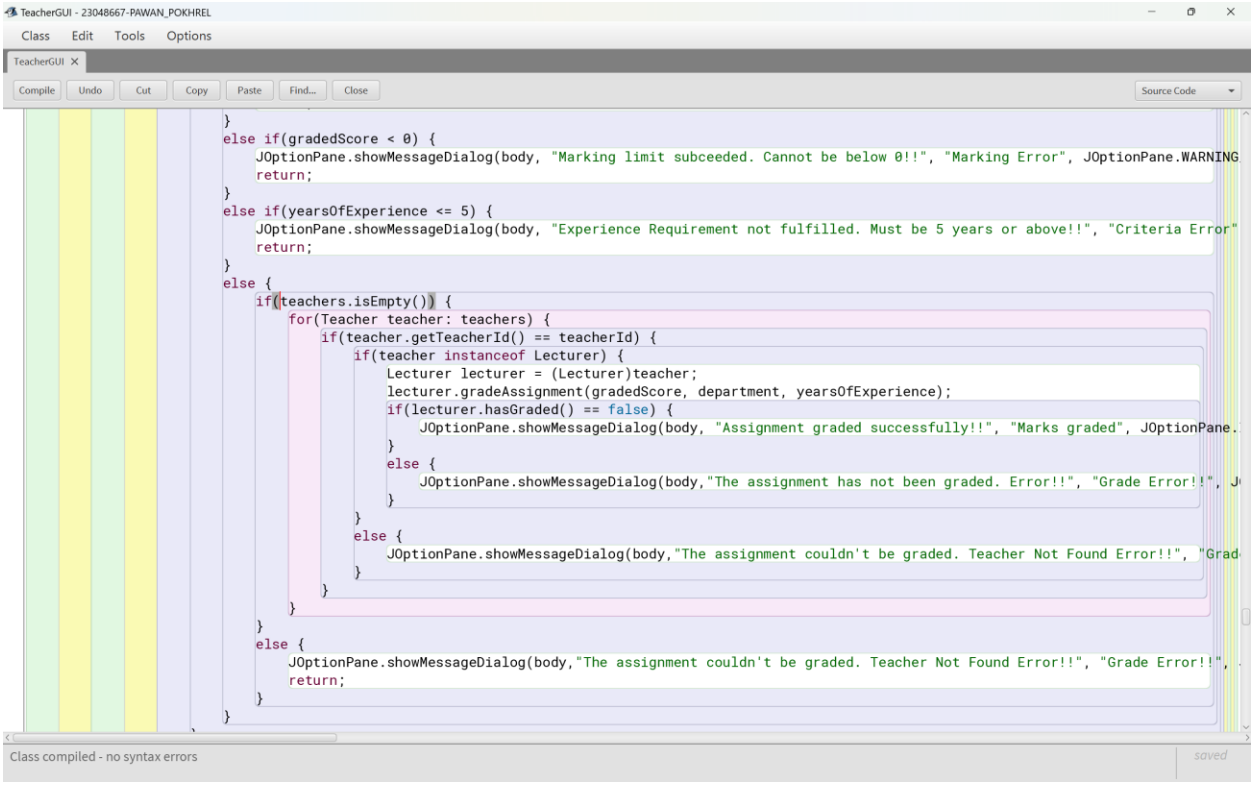


Figure 25: The program jumping into a wrong condition due to missing !

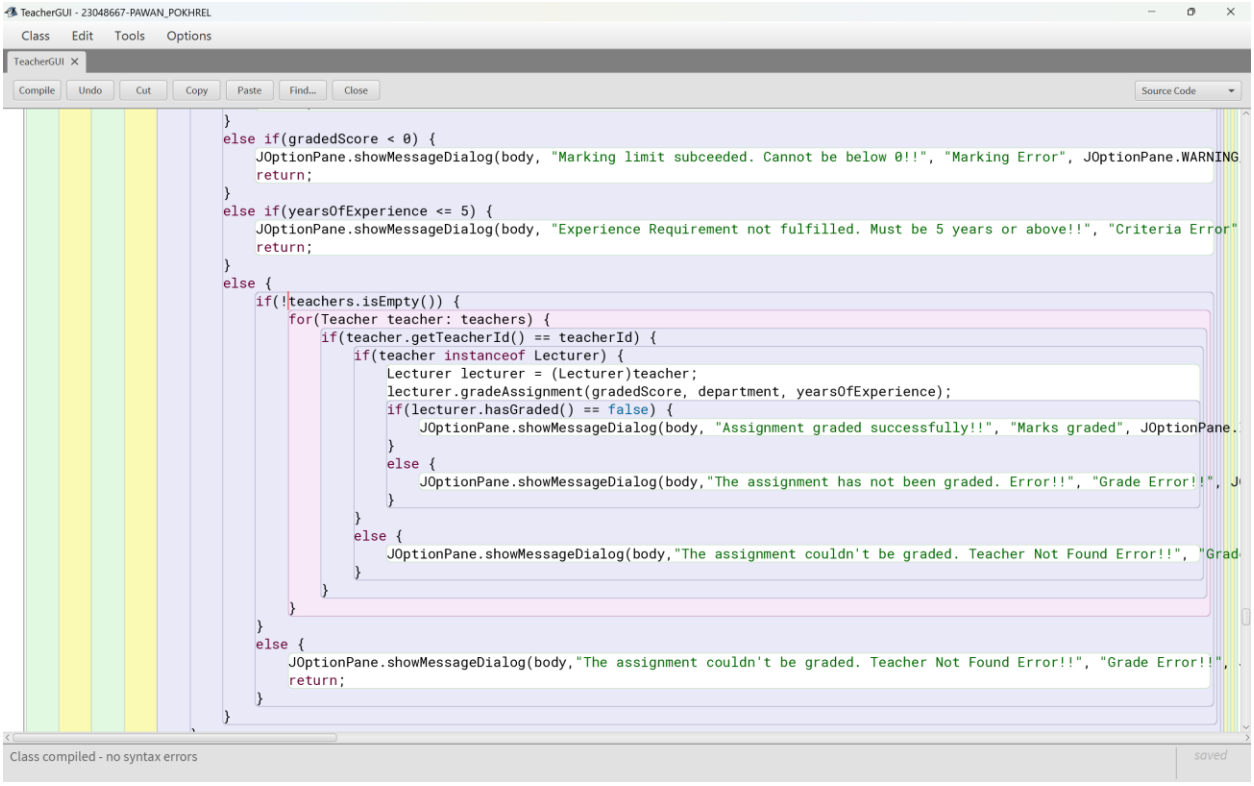


Figure 26: Fixing the issue

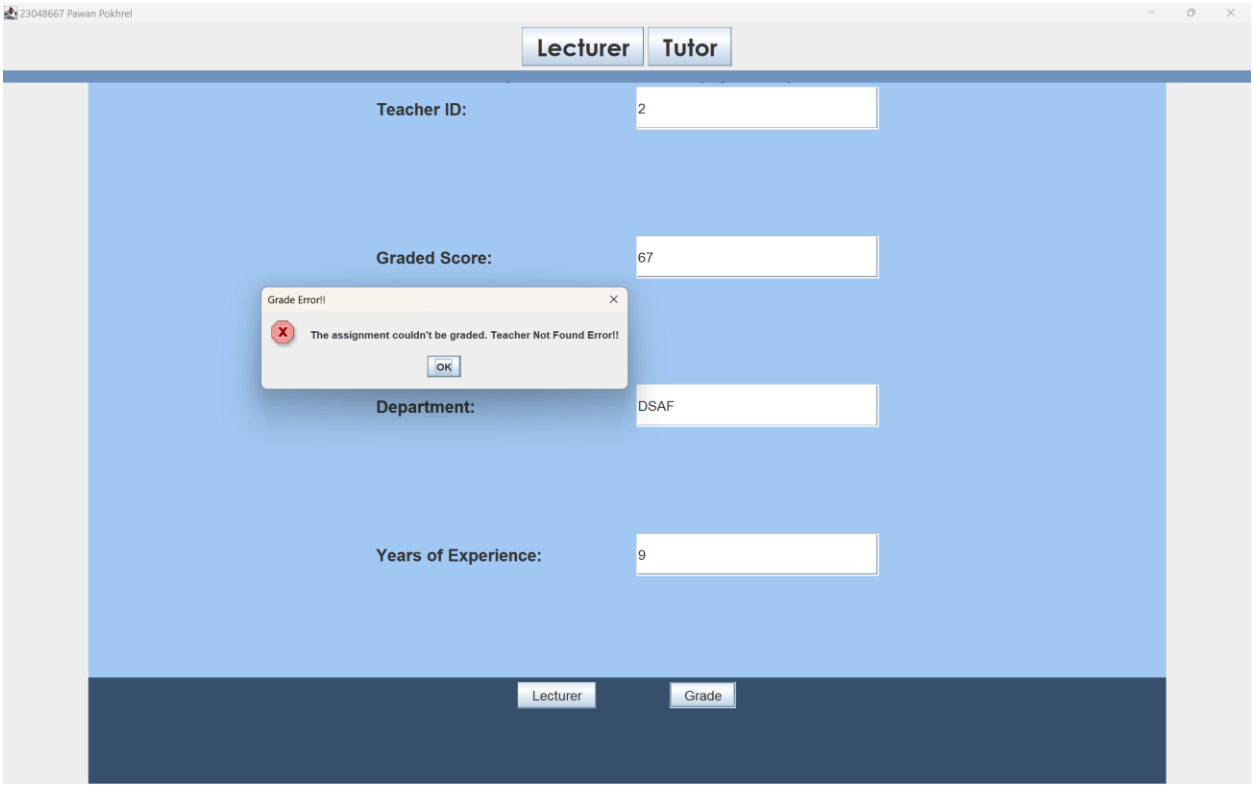


Figure 27: an expected output after fixing the issue

CHAPTER VII: CONCLUSION

7.1. Conclusion

In conclusion, the second phase of the CS4001NI Programming Module has been instrumental in broadening students' proficiency in Java programming, particularly with an emphasis on integrating graphical user interface (GUI) elements into Java applications. Through the comprehensive exploration and implementation of the TeacherGUI class and its associated functionalities, students have deepened their understanding of fundamental object-oriented programming (OOP) principles while gaining invaluable hands-on experience in designing and implementing sophisticated, user-centric software solutions. By leveraging Swing components, event handling mechanisms, and layout managers, students have not only mastered the technical intricacies of GUI development but have also cultivated a keen appreciation for the importance of intuitive and visually appealing interfaces in software usability and user satisfaction.

Furthermore, this phase has served as a pivotal platform for students to grasp the profound implications of GUI design in shaping the user experience and driving the success of modern software applications. By encapsulating complex functionalities within seamless and aesthetically pleasing interfaces, Java developers can optimize user interaction and streamline workflow processes, ultimately enhancing overall efficiency and productivity. Armed with these advanced skills and insights, students are well-positioned to navigate the dynamic landscape of the software industry, where user-centric design principles are increasingly recognized as essential drivers of product adoption and market competitiveness. Through this coursework, students have not only elevated their technical proficiency but have also cultivated a mindset rooted in innovation, empathy, and a relentless commitment to delivering impactful solutions that meet and exceed user expectations.

REFERENCES

Bates, M., 2011. *Science Direct.* [Online] Available at: https://www.sciencedirect.com/topics/engineering/syntax-error#:~:text=Syntax%20errors%20are%20mistakes%20in%20the%20source%20code%2C%20such%20as,corrected%20in%20the%20edit%20window.
[Accessed Jan 2024].

Hartman, J., 2023. *Guru99.* [Online] Available at: https://www.guru99.com/java-platform.html
[Accessed January 2024].

Pokhrel, P., 2024. *30% Individual Coursework,* Lalitpur, Nepal: Pawan Pokhrel.

Shan–shan, Z. Z. W., 2021. Formal Definition of Pseudo Code and Mapping Rules to Java Code. In: *2021 IEEE 4th International Conference on Computer and Communication Engineering Technology (CCET).* Beijing,China: s.n., pp. 175-179.

Uzunbayir, S., 2018. *Class Diagrams Note.* [Online] Available at: http://homes.ieu.edu.tr/culudagli/files/SE305/Labs/Week11/Class%20Diagrams%20Notes.pdf

Woodruff, A., 2023. *EasyTechJunkie.* [Online] Available at: https://www.easytechjunkie.com/what-is-a-logic-error.htm
[Accessed January 2024].

APPENDIX

Code of Teacher.java

```
public class Teacher
{

    //instance variables of Teacher Class with private access modifiers (concept of encapsulation)
    private int teacherId;
    private String teacherName;
    private String address;
    private String workingType;
    private String employmentStatus;
    private double workingHours;

    //constructor for the class Teacher with 5 paramaters
    public Teacher(int teacherId, String teacherName, String address, String workingType, String employmentStatus)
    {
        this.teacherId = teacherId;
        this.teacherName = teacherName;
        this.address = address;
        this.workingType = workingType;
        this.employmentStatus = employmentStatus;
```

```
        this.workingHours = -1;//workingHours is a non-negative value. So, it will be '-1' if no value is assigned.
    }

    //accessor methods getXXX() for each attributes
    //accessor method for teacherId
    public int getTeacherId()
    {
        return this.teacherId;
    }

    //accesor method for teacherName
    public String getTeacherName()
    {
        return this.teacherName;
    }

    //accessor method for address
    public String getAddress()
    {
        return this.address;
    }
```

```
    //accessor method for workingType
    public String getWorkingType()
    {
        return this.workingType;
    }

    //accessor method for employmentStatus
    public String getEmploymentStatus()
    {
        return this.employmentStatus;
    }

    //accessor method for workingHours
    public double getWorkingHours()
    {
        return this.workingHours;
    }


    //setter method setXXX() for setting workingHours
    public void setWorkingHours(double workingHours)
    {
        //working hour is a non-negative entry
```

```
        if(workingHours < 0) {
            System.out.println("Invalid entry for working hour");
        }
        else {
            this.workingHours = workingHours;
        }
    }

    //method for displaying the obtained attribute values through accessor methods
    public void display()
    {
        System.out.println("Teacher ID ----------> " + getTeacherId());
        System.out.println("Teacher Name ----------> " + getTeacherName());
        System.out.println("Address ----------> " + getAddress());
        System.out.println("Working Type ----------> " + getWorkingType());
        System.out.println("Employment Status -------> " + getEmploymentStatus());

        //since the workingHours has been set to -1 at the beginning, so if the value is not yet assigned
        if(getWorkingHours() == -1) {
            System.out.println("The working hours has not been assigned!!");
        }
        else {
```

```
            System.out.println("Working Hours ----------> " + getWorkingHours() + " hrs");
        }
    }
}
```

Code of Lecturer.java

```
//Lecturer class inherits the properties of Teacher class (concept of inheritance)
public class Lecturer extends Teacher
{

    //instance variables of Lecturer Class with private access modifiers (concept of encapsulation)
    private String department;
    private int yearsOfExperience;
    private int gradedScore;
    private boolean hasGraded;

    //a variable grade to store the grade according to the gradedScore and then later print if required
    String grade;

    //constructor for the class Tutor with 8 paramaters
    public Lecturer(int teacherId, String teacherName, String address, String workingType, String employmentStatus, double workingHours, String department, int yearsOfExperience)
    {
        //call to the super class constructor
        super(teacherId, teacherName, address, workingType, employmentStatus);
```

```
        //call to the setter method for workingHours in the super class
        super.setWorkingHours(workingHours);

        this.department = department;
        this.yearsOfExperience = yearsOfExperience;
        this.gradedScore = 0;
        this.hasGraded = false;
    }

    //accessor methods getXXX() for each attributes
    //accessor method for department
    public String getDepartment()
    {
        return this.department;
    }

    //accessor method for yearsOfExperience
    public int getYearsOfExperience()
    {
        return this.yearsOfExperience;
    }
```

```
    //accessor method for gradedScore
    public int getGradedScore()
    {
        return this.gradedScore;
    }

    //accessor method for hasGraded
    public boolean hasGraded()
    {
        return this.hasGraded;
    }

    //accessor method for grade
    public String getGrade()
    {
        return this.grade;
    }

    //mutator method setXXX() for gradedScore to set the value
    public void setGradedScore(int gradedScore)
    {
        this.gradedScore = gradedScore;
    }
```

```
    //mehtod for grading assignment
    public void gradeAssignment(int gradedScore, String department, int yearsOfExperience)
    {
        //for checking whether the assignment is previously graded or not
        if(hasGraded() == false) {
            if(yearsOfExperience >= 5 && department == this.department) {
                if(gradedScore >= 70) {
                    this.grade = "A";
                }
                else if(gradedScore >= 60) {
                    this.grade = "B";
                }
                else if(gradedScore >= 50) {
                    this.grade = "C";
                }
                else if(gradedScore >= 40) {
                    this.grade = "D";
                }
                else {
                    this.grade = "E";
                }
```

```
            setGradedScore(gradedScore);
            this.hasGraded = true;
            System.out.println("Assignment graded successfully!! \nGrade --> " +
getGrade());
        }

        else {
            System.out.println("You are not eligible for grading this assignment!!");
        }
    }

    else {
        System.out.println("This assignment has already been graded!!");
    }
}

//display method overriding
@Override
//method for displaying details of lecturer
public void display()
{
    super.display();
    System.out.println("Department ----------> " + getDepartment());
```

```
        System.out.println("Years of Experience -----> " + getYearsOfExperience());


        if(hasGraded() == false) {
            System.out.println("The grade has not been graded yet!!");
        }
        else {
            System.out.println("Graded Score ----------> " + getGradedScore() + "\nGrade --
----------------> "+ getGrade());
        }
    }
}
```

Code of Tutor.java

```
//Tutor class inherits the properties of Teacher class (concept of inheritance)
public class Tutor extends Teacher
{

    //instance variables of Tutor Class with private access modifier (concept of encapsulation)
    private double salary;
    private String specialization;
    private String academicQualifications;
    private int performanceIndex;
    private boolean isCertified;

    //constructor for the class Tutor with 10 paramaters
    public Tutor(int teacherId, String teacherName, String address, String workingType, String employmentStatus, double workingHours, double salary, String specialization, String academicQualifications, int performanceIndex)
    {
        super(teacherId, teacherName, address, workingType, employmentStatus);
        super.setWorkingHours(workingHours);
        this.salary = salary;
        this.specialization = specialization;
        this.academicQualifications = academicQualifications;
```

```
        this.performanceIndex = performanceIndex;
        this.isCertified = false;
    }

    //accessor methods getXXX() for each attributes
    //accessor method for salary
    public double getSalary()
    {
        return this.salary;
    }

    //accessor method for specialization
    public String getSpecialization()
    {
        return this.specialization;
    }

    //accessor method for academicQualifications
    public String getAcademicQualifications()
    {
        return this.academicQualifications;
    }
```

```
    //accessor method for performanceIndex
    public int getPerformanceIndex()
    {
        return this.performanceIndex;
    }

    //accessor method for isCertified
    public boolean isCertified()
    {
        return this.isCertified;
    }

    //setter method setXXX() for setting the salary of tutor
    public void setSalary(double salary, int performanceIndex)
    {
        //a variable to store the appraisal%
        double appraisal;

        //to check if the tutor is already certified or not
        if(isCertified() == false) {
            if(performanceIndex > 5 && getWorkingHours() > 20) {
                if(performanceIndex <= 7) {
                    appraisal = 5;
```

```
            }
            else if(performanceIndex <= 9) {
                appraisal = 10;
            }
            else {
                appraisal = 20;
            }
            this.salary = salary + (appraisal / 100) * salary;
            this.performanceIndex = performanceIndex;
            this.isCertified = true;
            System.out.println("Salary is set with " + appraisal + "% appraisal");
        }
        else {
            System.out.println("Salary cannot be approved as per the criteria");
        }
    }
    else {
        System.out.println("The tutor is already certified with an approved salary");
    }
}

//method to remove the tutors that are not certified
public void removeTutor()
```

```
    {
        if(isCertified() == false) {
            this.salary = 0;
            this.specialization = "";
            this.academicQualifications = "";
            this.performanceIndex = 0;
            this.isCertified = false;
        }
        else {
            System.out.println("Tutor is certified; cannot be removed.");
        }
    }

    //display method overriding
    @Override
    //method to display the details of the tutor
    public void display()
    {
        //since the display method of super class is called in both the conditions of tutor being certified and not being certified
        super.display();

        //displaying the information of the tutor only if the tutor is certified
```

```
        if(isCertified() == true) {
            System.out.println("Salary ------------------> " + getSalary());
            System.out.println("Specialization ----------> " + getSpecialization());
            System.out.println("Academic Qualification --> " + getAcademicQualifications());
            System.out.println("Performance Index -------> " + getPerformanceIndex());
        }
    }
}
```

Code of TeacherGUI.java

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.util.*;



/**
 * The TeacherGUI class provides a graphical user interface for managing teachers.
 * It includes methods for setting up the GUI components, handling user actions,
 * and managing teacher data.
 */

public class TeacherGUI
{
    public static void main(String[] args)
    {
        // Creating an ArrayList to store teacher objects
        ArrayList<Teacher> teachers = new ArrayList<Teacher>();


        // Creating the main frame for the GUI
        JFrame body = new JFrame("23048667 Pawan Pokhrel");
```

```
// Creating font styles for labels and text fields
Font titleFont = new Font("Century Gothic", Font.BOLD, 28);
Font textFont = new Font("Roboto",Font.PLAIN, 16);
Font labelFont = new Font("Roboto", Font.BOLD, 20);

JFrame welcomePage = new JFrame("23048667 Pawan Pokhrel");
JLabel welcomeTitle = new JLabel("Welcome To Teacher Management Console");
welcomeTitle.setFont(titleFont);
JPanel titlePanel = new JPanel();
JTextArea welcomeText = new JTextArea("\t\t\tWelcome to the Teacher Management Console! 🍎➖\n\n\tThis intuitive platform empowers you to effortlessly manage your educational faculty with precision and ease. From\n\ttracking teacher details to grading assignments and setting salaries, our system streamlines the entire process, \n\tensuring smooth operations for your institution.Experience the efficiency and effectiveness of our Teacher Management \n\tConsole today! \n\n\tGet started and unlock a world of educational administration made simple. 📚💻 ");
welcomeText.setFont(labelFont);
JPanel welcomePanel = new JPanel();
welcomeText.setEditable(false);

// Creating panels for different sections of the GUI
JPanel navBar = new JPanel();
JPanel lecturerBody = new JPanel();
```

```
JPanel lecturerLeftBody = new JPanel();

JPanel lecturerRightBody = new JPanel();

JPanel gradeAssignmentBody = new JPanel();

JPanel tutorBody = new JPanel();

JPanel tutorLeftBody = new JPanel();

JPanel tutorRightBody = new JPanel();

JPanel setSalaryBody = new JPanel();

JPanel eastGap = new JPanel();

JPanel westGap = new JPanel();

JPanel southGap = new JPanel();


// Creating labels for headers and text fields for user input

JLabel headerLabel = new JLabel("Teacher Management Console");

headerLabel.setFont(titleFont);

JLabel gradeAssignmentTitle = new JLabel("Grade the Assignments");

gradeAssignmentTitle.setFont(titleFont);

JLabel setSalaryTitle = new JLabel("Set Salary");

setSalaryTitle.setFont(titleFont);

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

JTextField teacherIdField = new JTextField(20);

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

JTextField teacherNameField = new JTextField(20);

JLabel addressLabel = new JLabel("Address: ");
```

```
JTextField addressField = new JTextField(20);

JLabel workingTypeLabel = new JLabel("Working Type: ");

JTextField workingTypeField = new JTextField(20);

JLabel employmentStatusLabel = new JLabel("Employment Status: ");

JTextField employmentStatusField = new JTextField(20);

JLabel workingHoursLabel = new JLabel("Working Hours: ");

JTextField workingHoursField = new JTextField(20);

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

JTextField departmentField = new JTextField(20);

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

JTextField yearsOfExperienceField = new JTextField(20);

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

JTextField gradedScoreField = new JTextField(20);

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

JTextField salaryField = new JTextField(20);

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

JTextField specializationField = new JTextField(20);

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

JTextField academicQualificationsField = new JTextField(20);

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

JTextField performanceIndexField = new JTextField(20);


// Setting font styles for labels and text fields
```

```
teacherIdLabel.setFont(labelFont);
teacherNameLabel.setFont(labelFont);
addressLabel.setFont(labelFont);
workingTypeLabel.setFont(labelFont);
employmentStatusLabel.setFont(labelFont);
workingHoursLabel.setFont(labelFont);
departmentLabel.setFont(labelFont);
yearsOfExperienceLabel.setFont(labelFont);
gradedScoreLabel.setFont(labelFont);
salaryLabel.setFont(labelFont);
specializationLabel.setFont(labelFont);
academicQualificationsLabel.setFont(labelFont);
performanceIndexLabel.setFont(labelFont);
welcomeText.setFont(textFont);

teacherIdField.setFont(textFont);
teacherNameField.setFont(textFont);
addressField.setFont(textFont);
workingTypeField.setFont(textFont);
employmentStatusField.setFont(textFont);
workingHoursField.setFont(textFont);
departmentField.setFont(textFont);
yearsOfExperienceField.setFont(textFont);
```

```
        gradedScoreField.setFont(textFont);
        salaryField.setFont(textFont);
        specializationField.setFont(textFont);
        academicQualificationsField.setFont(textFont);
        performanceIndexField.setFont(textFont);

        // Creating panels for each section of the GUI
        JPanel headerPanel = new JPanel();
        JPanel teacherIdPanel = new JPanel();
        JPanel teacherNamePanel = new JPanel();
        JPanel addressPanel = new JPanel();
        JPanel workingTypePanel = new JPanel();
        JPanel employmentStatusPanel = new JPanel();
        JPanel workingHoursPanel = new JPanel();
        JPanel departmentPanel = new JPanel();
        JPanel yearsOfExperiencePanel = new JPanel();
        JPanel gradedScorePanel = new JPanel();
        JPanel salaryPanel = new JPanel();
        JPanel specializationPanel = new JPanel();
        JPanel academicQualificationsPanel = new JPanel();
        JPanel performanceIndexPanel = new JPanel();
        JPanel welcomeNav = new JPanel();
        JPanel welcomeButtons = new JPanel();
```

```
// Creating buttons for navigation and actions
JButton directToLecturerPage = new JButton("Lecturer");
JButton directToTutorPage = new JButton("Tutor");
JButton addLecturer = new JButton("Add a Lecturer");
JButton addTutor = new JButton("Add a Tutor");
JButton goToGradeAssignments = new JButton("Grade the Assignments");
JButton goToSetSalary = new JButton("Set the salary of Tutor");
JButton removeTutor = new JButton("Remove the Tutor");
JButton displayLecturer = new JButton("Display");
JButton displayTutor = new JButton("Display");
JButton clear = new JButton("Clear");
JButton welcomeToLecturer = new JButton("Lecturer");
JButton welcomeToTutor = new JButton("Tutor");
JButton gradeAssignments = new JButton("Grade");
JButton setSalary = new JButton("Set");
JButton backToLecturerPage = new JButton("Lecturer");
JButton backToTutorPage = new JButton("Tutor");

// Setting font styles for buttons
directToLecturerPage.setFont(titleFont);
directToTutorPage.setFont(titleFont);
addLecturer.setFont(textFont);
```

```
addTutor.setFont(textFont);
goToGradeAssignments.setFont(textFont);
goToSetSalary.setFont(textFont);
gradeAssignments.setFont(textFont);
setSalary.setFont(textFont);
removeTutor.setFont(textFont);
displayLecturer.setFont(textFont);
displayTutor.setFont(textFont);
clear.setFont(textFont);

// Creating panels for buttons
JPanel lecturerButtons = new JPanel();
JPanel tutorButtons = new JPanel();
JPanel commonButtons = new JPanel();
JPanel gradeButtons = new JPanel();
JPanel setButtons = new JPanel();

// Making the welcome frame visible and maximizing it
welcomePage.setVisible(true);
welcomePage.setExtendedState(JFrame.MAXIMIZED_BOTH);
welcomePage.setLayout(null);

// Adding the title panel to the navigation bar
```

```
welcomeNav.setBackground(new Color(54, 80, 107));


// Creating and setting up the title panel
titlePanel.setBackground(new Color(54, 80, 107));


// Setting up the welcome panel
welcomePanel.setBackground(new Color(112, 154, 191));
welcomeText.setPreferredSize(new Dimension(800, 150)); // Adjust dimensions as needed
welcomeText.setLineWrap(true); // Enable line wrapping
welcomeText.setWrapStyleWord(true); // Wrap at word boundaries


// Adding components to the welcome frame
welcomePage.add(welcomeNav);
welcomeNav.setBounds(0,0,1463,120);
welcomeNav.setLayout(new FlowLayout(FlowLayout.CENTER));
welcomeNav.add(titlePanel);
titlePanel.add(welcomeTitle);
welcomePage.add(welcomePanel);
welcomePanel.setBounds(250,140,900,400);
welcomePanel.add(welcomeText);
welcomePanel.setLayout(new GridLayout(2,1,0,50));
welcomePanel.add(welcomeButtons);
```

```
welcomeButtons.setPreferredSize(new Dimension(1000,100));
welcomeButtons.add(welcomeToLecturer);
welcomeButtons.add(welcomeToTutor);
welcomeToLecturer.setPreferredSize(new Dimension(100,50));
welcomeToTutor.setPreferredSize(new Dimension(100,50));


// Setting the layout of the main frame and adding panels to it
body.setLayout(new BorderLayout());
body.add(navBar, BorderLayout.NORTH);
body.add(eastGap, BorderLayout.EAST);
body.add(westGap, BorderLayout.WEST);
navBar.setLayout(new FlowLayout(FlowLayout.CENTER));
navBar.add(directToLecturerPage);
navBar.add(directToTutorPage);

goToGradeAssignments.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e)
    {
        body.setVisible(true);
        welcomePage.setVisible(false);
        body.setExtendedState(JFrame.MAXIMIZED_BOTH);
        body.add(gradeAssignmentBody, BorderLayout.CENTER);
```

```
navBar.add(headerPanel);
headerPanel.add(headerLabel);
headerPanel.setBackground(new Color(112,146,190));
headerLabel.setHorizontalAlignment(SwingConstants.CENTER);
headerPanel.setPreferredSize(new Dimension(1463,60));

gradeAssignmentBody.setVisible(true);
setSalaryBody.setVisible(false);
lecturerBody.setVisible(false);
tutorBody.setVisible(false);

gradeAssignmentBody.setLayout(new GridLayout(5,1,0,50));
gradeAssignmentBody.add(teacherIdPanel);
gradeAssignmentBody.add(gradedScorePanel);
gradeAssignmentBody.add(departmentPanel);
gradeAssignmentBody.add(yearsOfExperiencePanel);
gradeAssignmentBody.add(gradeButtons);

FlowLayout GapFlowLayout = new FlowLayout();
GapFlowLayout.setHgap(86);

gradeButtons.setLayout(GapFlowLayout);
```

```
        gradeButtons.add(backToLecturerPage);
        backToLecturerPage.setFont(textFont);
        gradeButtons.add(gradeAssignments);
        gradeButtons.setBackground(new Color(54,80,107));

        gradeAssignmentBody.setPreferredSize(new Dimension(500, 700));
        gradeAssignmentBody.setBackground(new Color(161,200,242));
    }
});

goToSetSalary.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e)
    {
        body.setVisible(true);
        welcomePage.setVisible(false);
        body.setExtendedState(JFrame.MAXIMIZED_BOTH);
        body.add(setSalaryBody, BorderLayout.CENTER);

        navBar.add(headerPanel);
        headerPanel.add(headerLabel);
        headerPanel.setBackground(new Color(112,146,190));
        headerLabel.setHorizontalAlignment(SwingConstants.CENTER);
        headerPanel.setPreferredSize(new Dimension(1463,60));
```

```
gradeAssignmentBody.setVisible(false);
setSalaryBody.setVisible(true);
lecturerBody.setVisible(false);
tutorBody.setVisible(false);

setSalaryBody.setLayout(new GridLayout(4,1,0,50));
setSalaryBody.add(teacherIdPanel);
setSalaryBody.add(salaryPanel);
setSalaryBody.add(performanceIndexPanel);
setSalaryBody.add(setButtons);

FlowLayout GapFlowLayout = new FlowLayout();
GapFlowLayout.setHgap(86);

setButtons.setLayout(GapFlowLayout);
setButtons.add(backToTutorPage);
backToTutorPage.setFont(textFont);
setButtons.add(setSalary);
setButtons.setBackground(new Color(54,80,107));

setSalaryBody.setPreferredSize(new Dimension(500, 700));
setSalaryBody.setBackground(new Color(161,200,242));
```

```
        }
    });

    backToLecturerPage.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e)
        {
            body.setVisible(true);
            welcomePage.setVisible(false);
            body.setExtendedState(JFrame.MAXIMIZED_BOTH);
            body.add(lecturerBody, BorderLayout.CENTER);

            navBar.add(headerPanel);
            headerPanel.add(headerLabel);
            headerPanel.setBackground(new Color(112,146,190));
            headerLabel.setHorizontalAlignment(SwingConstants.CENTER);
            headerPanel.setPreferredSize(new Dimension(1463,60));

            gradeAssignmentBody.setVisible(false);
            setSalaryBody.setVisible(false);
            lecturerBody.setVisible(true);
            tutorBody.setVisible(false);

            lecturerBody.setLayout(new FlowLayout());
```

```
lecturerBody.add(lecturerLeftBody);
lecturerBody.add(lecturerRightBody);

lecturerLeftBody.setLayout(new GridLayout(5,1,0,50));
lecturerLeftBody.add(teacherIdPanel);
lecturerLeftBody.add(teacherNamePanel);
lecturerLeftBody.add(addressPanel);
lecturerLeftBody.add(workingTypePanel);
lecturerLeftBody.add(lecturerButtons);

lecturerRightBody.setLayout(new GridLayout(5,1,0,50));
lecturerRightBody.add(employmentStatusPanel);
lecturerRightBody.add(workingHoursPanel);
lecturerRightBody.add(departmentPanel);
lecturerRightBody.add(yearsOfExperiencePanel);
lecturerRightBody.add(commonButtons);

FlowLayout GapFlowLayout = new FlowLayout();
GapFlowLayout.setHgap(86);

lecturerButtons.setLayout(GapFlowLayout);
lecturerButtons.add(addLecturer);
lecturerButtons.add(goToGradeAssignments);
```

```
        commonButtons.setLayout(GapFlowLayout);
        commonButtons.add(displayLecturer);
        displayTutor.setVisible(false);
        commonButtons.add(clear);
        lecturerButtons.setBackground(new Color(54,80,107));
        commonButtons.setBackground(new Color(54,80,107));


        lecturerBody.setPreferredSize(new Dimension(1200, 700));
        lecturerBody.setBackground(new Color(112,154,191));
        lecturerLeftBody.setPreferredSize(new Dimension(600, 700));
        lecturerRightBody.setPreferredSize(new Dimension(600, 700));
        lecturerLeftBody.setBackground(new Color(161,200,242));
        lecturerRightBody.setBackground(new Color(161,200,242));
    }
});

welcomeToLecturer.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e)
    {
        body.setVisible(true);
        welcomePage.setVisible(false);
```

```
body.setExtendedState(JFrame.MAXIMIZED_BOTH);
body.add(lecturerBody, BorderLayout.CENTER);

navBar.add(headerPanel);
headerPanel.add(headerLabel);
headerPanel.setBackground(new Color(112,146,190));
headerLabel.setHorizontalAlignment(SwingConstants.CENTER);
headerPanel.setPreferredSize(new Dimension(1463,60));

gradeAssignmentBody.setVisible(false);
setSalaryBody.setVisible(false);
lecturerBody.setVisible(true);
tutorBody.setVisible(false);

lecturerBody.setLayout(new FlowLayout());
lecturerBody.add(lecturerLeftBody);
lecturerBody.add(lecturerRightBody);

lecturerLeftBody.setLayout(new GridLayout(5,1,0,50));
lecturerLeftBody.add(teacherIdPanel);
lecturerLeftBody.add(teacherNamePanel);
lecturerLeftBody.add(addressPanel);
lecturerLeftBody.add(workingTypePanel);
```

```
lecturerLeftBody.add(lecturerButtons);


lecturerRightBody.setLayout(new GridLayout(5,1,0,50));
lecturerRightBody.add(employmentStatusPanel);
lecturerRightBody.add(workingHoursPanel);
lecturerRightBody.add(departmentPanel);
lecturerRightBody.add(yearsOfExperiencePanel);
lecturerRightBody.add(commonButtons);

FlowLayout GapFlowLayout = new FlowLayout();
GapFlowLayout.setHgap(86);

lecturerButtons.setLayout(GapFlowLayout);
lecturerButtons.add(addLecturer);
lecturerButtons.add(goToGradeAssignments);

commonButtons.setLayout(GapFlowLayout);
commonButtons.add(displayLecturer);
displayTutor.setVisible(false);
commonButtons.add(clear);
lecturerButtons.setBackground(new Color(54,80,107));
commonButtons.setBackground(new Color(54,80,107));
```

```
        lecturerBody.setPreferredSize(new Dimension(1200, 700));
        lecturerBody.setBackground(new Color(112,154,191));
        lecturerLeftBody.setPreferredSize(new Dimension(600, 700));
        lecturerRightBody.setPreferredSize(new Dimension(600, 700));
        lecturerLeftBody.setBackground(new Color(161,200,242));
        lecturerRightBody.setBackground(new Color(161,200,242));
    }
});

backToTutorPage.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e)
    {
        body.setVisible(true);
        welcomePage.setVisible(false);
        body.setExtendedState(JFrame.MAXIMIZED_BOTH);
        body.add(tutorBody, BorderLayout.CENTER);

        navBar.add(headerPanel);
        headerPanel.add(headerLabel);
        headerPanel.setBackground(new Color(112,146,190));
        headerLabel.setHorizontalAlignment(SwingConstants.CENTER);
        headerPanel.setPreferredSize(new Dimension(1463,60));
```

```
gradeAssignmentBody.setVisible(false);
setSalaryBody.setVisible(false);
lecturerBody.setVisible(false);
tutorBody.setVisible(true);

tutorBody.setLayout(new FlowLayout());
tutorBody.add(tutorLeftBody);
tutorBody.add(tutorRightBody);

tutorLeftBody.setLayout(new GridLayout(6,1,0,50));
tutorLeftBody.add(teacherIdPanel);
tutorLeftBody.add(teacherNamePanel);
tutorLeftBody.add(addressPanel);
tutorLeftBody.add(workingTypePanel);
tutorLeftBody.add(specializationPanel);
tutorLeftBody.add(tutorButtons);

tutorRightBody.setLayout(new GridLayout(6,1,0,50));
tutorRightBody.add(employmentStatusPanel);
tutorRightBody.add(workingHoursPanel);
tutorRightBody.add(salaryPanel);
tutorRightBody.add(academicQualificationsPanel);
```

```
tutorRightBody.add(performanceIndexPanel);
tutorRightBody.add(commonButtons);

tutorButtons.setLayout(new FlowLayout());
tutorButtons.add(addTutor);
tutorButtons.add(goToSetSalary);
tutorButtons.add(removeTutor);

FlowLayout GapFlowLayout = new FlowLayout();
GapFlowLayout.setHgap(86);

commonButtons.setLayout(GapFlowLayout);
commonButtons.add(displayTutor);
displayLecturer.setVisible(false);
commonButtons.add(clear);

tutorButtons.setBackground(new Color(54,80,107));
commonButtons.setBackground(new Color(54,80,107));

tutorBody.setPreferredSize(new Dimension(1200, 700));
tutorBody.setBackground(new Color(112,154,191));
tutorLeftBody.setPreferredSize(new Dimension(600, 700));
tutorRightBody.setPreferredSize(new Dimension(600, 700));
```

```
            tutorLeftBody.setBackground(new Color(161,200,242));
            tutorRightBody.setBackground(new Color(161,200,242));
        }
    });

    welcomeToTutor.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e)
        {
            body.setVisible(true);
            welcomePage.setVisible(false);
            body.setExtendedState(JFrame.MAXIMIZED_BOTH);
            body.add(tutorBody, BorderLayout.CENTER);

            navBar.add(headerPanel);
            headerPanel.add(headerLabel);
            headerPanel.setBackground(new Color(112,146,190));
            headerLabel.setHorizontalAlignment(SwingConstants.CENTER);
            headerPanel.setPreferredSize(new Dimension(1463,60));

            gradeAssignmentBody.setVisible(false);
            setSalaryBody.setVisible(false);
            lecturerBody.setVisible(false);
            tutorBody.setVisible(true);
```

```
tutorBody.setLayout(new FlowLayout());
tutorBody.add(tutorLeftBody);
tutorBody.add(tutorRightBody);

tutorLeftBody.setLayout(new GridLayout(6,1,0,50));
tutorLeftBody.add(teacherIdPanel);
tutorLeftBody.add(teacherNamePanel);
tutorLeftBody.add(addressPanel);
tutorLeftBody.add(workingTypePanel);
tutorLeftBody.add(specializationPanel);
tutorLeftBody.add(tutorButtons);

tutorRightBody.setLayout(new GridLayout(6,1,0,50));
tutorRightBody.add(employmentStatusPanel);
tutorRightBody.add(workingHoursPanel);
tutorRightBody.add(salaryPanel);
tutorRightBody.add(academicQualificationsPanel);
tutorRightBody.add(performanceIndexPanel);
tutorRightBody.add(commonButtons);

tutorButtons.setLayout(new FlowLayout());
tutorButtons.add(addTutor);
```

```
        tutorButtons.add(goToSetSalary);
        tutorButtons.add(removeTutor);

        FlowLayout GapFlowLayout = new FlowLayout();
        GapFlowLayout.setHgap(86);

        commonButtons.setLayout(GapFlowLayout);
        commonButtons.add(displayTutor);
        displayLecturer.setVisible(false);
        commonButtons.add(clear);

        tutorButtons.setBackground(new Color(54,80,107));
        commonButtons.setBackground(new Color(54,80,107));

        tutorBody.setPreferredSize(new Dimension(1200, 700));
        tutorBody.setBackground(new Color(112,154,191));
        tutorLeftBody.setPreferredSize(new Dimension(600, 700));
        tutorRightBody.setPreferredSize(new Dimension(600, 700));
        tutorLeftBody.setBackground(new Color(161,200,242));
        tutorRightBody.setBackground(new Color(161,200,242));
    }
});
```

```
directToLecturerPage.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e)
    {
        body.setVisible(true);
        welcomePage.setVisible(false);
        body.setExtendedState(JFrame.MAXIMIZED_BOTH);
        body.add(lecturerBody, BorderLayout.CENTER);

        navBar.add(headerPanel);
        headerPanel.add(headerLabel);
        headerPanel.setBackground(new Color(112,146,190));
        headerLabel.setHorizontalAlignment(SwingConstants.CENTER);
        headerPanel.setPreferredSize(new Dimension(1463,60));

        gradeAssignmentBody.setVisible(false);
        setSalaryBody.setVisible(false);
        lecturerBody.setVisible(true);
        tutorBody.setVisible(false);

        lecturerBody.setLayout(new FlowLayout());
        lecturerBody.add(lecturerLeftBody);
        lecturerBody.add(lecturerRightBody);
```

```
lecturerLeftBody.setLayout(new GridLayout(5,1,0,50));
lecturerLeftBody.add(teacherIdPanel);
lecturerLeftBody.add(teacherNamePanel);
lecturerLeftBody.add(addressPanel);
lecturerLeftBody.add(workingTypePanel);
lecturerLeftBody.add(lecturerButtons);

lecturerRightBody.setLayout(new GridLayout(5,1,0,50));
lecturerRightBody.add(employmentStatusPanel);
lecturerRightBody.add(workingHoursPanel);
lecturerRightBody.add(departmentPanel);
lecturerRightBody.add(yearsOfExperiencePanel);
lecturerRightBody.add(commonButtons);

FlowLayout GapFlowLayout = new FlowLayout();
GapFlowLayout.setHgap(86);

lecturerButtons.setLayout(GapFlowLayout);
lecturerButtons.add(addLecturer);
lecturerButtons.add(goToGradeAssignments);

commonButtons.setLayout(GapFlowLayout);
commonButtons.add(displayLecturer);
```

```
            displayTutor.setVisible(false);
            commonButtons.add(clear);
            lecturerButtons.setBackground(new Color(54,80,107));
            commonButtons.setBackground(new Color(54,80,107));



            lecturerBody.setPreferredSize(new Dimension(1200, 700));
            lecturerBody.setBackground(new Color(112,154,191));
            lecturerLeftBody.setPreferredSize(new Dimension(600, 700));
            lecturerRightBody.setPreferredSize(new Dimension(600, 700));
            lecturerLeftBody.setBackground(new Color(161,200,242));
            lecturerRightBody.setBackground(new Color(161,200,242));
        }
    });

    directToTutorPage.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e)
        {
            body.setVisible(true);
            welcomePage.setVisible(false);
            body.setExtendedState(JFrame.MAXIMIZED_BOTH);
            body.add(tutorBody, BorderLayout.CENTER);
```

```
navBar.add(headerPanel);
headerPanel.add(headerLabel);
headerPanel.setBackground(new Color(112,146,190));
headerLabel.setHorizontalAlignment(SwingConstants.CENTER);
headerPanel.setPreferredSize(new Dimension(1463,60));

gradeAssignmentBody.setVisible(false);
setSalaryBody.setVisible(false);
lecturerBody.setVisible(false);
tutorBody.setVisible(true);

tutorBody.setLayout(new FlowLayout());
tutorBody.add(tutorLeftBody);
tutorBody.add(tutorRightBody);

tutorLeftBody.setLayout(new GridLayout(6,1,0,50));
tutorLeftBody.add(teacherIdPanel);
tutorLeftBody.add(teacherNamePanel);
tutorLeftBody.add(addressPanel);
tutorLeftBody.add(workingTypePanel);
tutorLeftBody.add(specializationPanel);
tutorLeftBody.add(tutorButtons);
```

```
tutorRightBody.setLayout(new GridLayout(6,1,0,50));
tutorRightBody.add(employmentStatusPanel);
tutorRightBody.add(workingHoursPanel);
tutorRightBody.add(salaryPanel);
tutorRightBody.add(academicQualificationsPanel);
tutorRightBody.add(performanceIndexPanel);
tutorRightBody.add(commonButtons);

tutorButtons.setLayout(new FlowLayout());
tutorButtons.add(addTutor);
tutorButtons.add(goToSetSalary);
tutorButtons.add(removeTutor);

FlowLayout GapFlowLayout = new FlowLayout();
GapFlowLayout.setHgap(86);

commonButtons.setLayout(GapFlowLayout);
commonButtons.add(displayTutor);
commonButtons.add(clear);

tutorButtons.setBackground(new Color(54,80,107));
commonButtons.setBackground(new Color(54,80,107));
```

```
        tutorBody.setPreferredSize(new Dimension(1200, 700));
        tutorBody.setBackground(new Color(112,154,191));
        tutorLeftBody.setPreferredSize(new Dimension(600, 700));
        tutorRightBody.setPreferredSize(new Dimension(600, 700));
        tutorLeftBody.setBackground(new Color(161,200,242));
        tutorRightBody.setBackground(new Color(161,200,242));
    }
});



eastGap.setPreferredSize(new Dimension(100,914));
westGap.setPreferredSize(new Dimension(100,914));
southGap.setPreferredSize(new Dimension(1463,100));



teacherIdPanel.add(teacherIdLabel);
teacherIdPanel.add(teacherIdField);
teacherNamePanel.add(teacherNameLabel);
teacherNamePanel.add(teacherNameField);
addressPanel.add(addressLabel);
```

```
addressPanel.add(addressField);
workingTypePanel.add(workingTypeLabel);
workingTypePanel.add(workingTypeField);
employmentStatusPanel.add(employmentStatusLabel);
employmentStatusPanel.add(employmentStatusField);
workingHoursPanel.add(workingHoursLabel);
workingHoursPanel.add(workingHoursField);
departmentPanel.add(departmentLabel);
departmentPanel.add(departmentField);
yearsOfExperiencePanel.add(yearsOfExperienceLabel);
yearsOfExperiencePanel.add(yearsOfExperienceField);
gradedScorePanel.add(gradedScoreLabel);
gradedScorePanel.add(gradedScoreField);
salaryPanel.add(salaryLabel);
salaryPanel.add(salaryField);
specializationPanel.add(specializationLabel);
specializationPanel.add(specializationField);
academicQualificationsPanel.add(academicQualificationsLabel);
academicQualificationsPanel.add(academicQualificationsField);
performanceIndexPanel.add(performanceIndexLabel);
performanceIndexPanel.add(performanceIndexField);

teacherIdLabel.setPreferredSize(new Dimension(300,50));
```

```
teacherIdField.setPreferredSize(new Dimension(300,50));
teacherNameLabel.setPreferredSize(new Dimension(300,50));
teacherNameField.setPreferredSize(new Dimension(300,50));
addressLabel.setPreferredSize(new Dimension(300,50));
addressField.setPreferredSize(new Dimension(300,50));
workingTypeLabel.setPreferredSize(new Dimension(300,50));
workingTypeField.setPreferredSize(new Dimension(300,50));
employmentStatusLabel.setPreferredSize(new Dimension(300,50));
employmentStatusField.setPreferredSize(new Dimension(300,50));
workingHoursLabel.setPreferredSize(new Dimension(300,50));
workingHoursField.setPreferredSize(new Dimension(300,50));
departmentLabel.setPreferredSize(new Dimension(300,50));
departmentField.setPreferredSize(new Dimension(300,50));
yearsOfExperienceLabel.setPreferredSize(new Dimension(300,50));
yearsOfExperienceField.setPreferredSize(new Dimension(300,50));
gradedScoreLabel.setPreferredSize(new Dimension(300,50));
gradedScoreField.setPreferredSize(new Dimension(300,50));
salaryLabel.setPreferredSize(new Dimension(300,50));
salaryField.setPreferredSize(new Dimension(300,50));
specializationLabel.setPreferredSize(new Dimension(300,50));
specializationField.setPreferredSize(new Dimension(300,50));
academicQualificationsLabel.setPreferredSize(new Dimension(300,50));
academicQualificationsField.setPreferredSize(new Dimension(300,50));
```

```
        performanceIndexLabel.setPreferredSize(new Dimension(300,50));
        performanceIndexField.setPreferredSize(new Dimension(300,50));

        teacherIdPanel.setOpaque(false);
        teacherNamePanel.setOpaque(false);
        addressPanel.setOpaque(false);
        workingTypePanel.setOpaque(false);
        employmentStatusPanel.setOpaque(false);
        workingHoursPanel.setOpaque(false);
        departmentPanel.setOpaque(false);
        yearsOfExperiencePanel.setOpaque(false);
        gradedScorePanel.setOpaque(false);
        salaryPanel.setOpaque(false);
        specializationPanel.setOpaque(false);
        academicQualificationsPanel.setOpaque(false);
        performanceIndexPanel.setOpaque(false);
        eastGap.setOpaque(false);
        westGap.setOpaque(false);
        southGap.setOpaque(false);
        lecturerBody.setOpaque(false);
        tutorBody.setOpaque(false);

        body.setPreferredSize(new Dimension(1463, 914));
```

```
addLecturer.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e)
    {
        try
        {
            if (teacherIdField.getText() == "" || teacherNameField.getText() == "" || addressField.getText() == "" || workingTypeField.getText() == "" || employmentStatusField.getText() == "" || departmentField.getText() == "" || workingHoursField.getText() == "" || yearsOfExperienceField.getText() == "") {
                JOptionPane.showMessageDialog(body, "Please, fill out the empty fields!!!", "Empty Fields !!", JOptionPane.WARNING_MESSAGE);
                return;
            }
            else {
                int teacherId = Integer.parseInt(teacherIdField.getText());
                String teacherName = teacherNameField.getText();
                String address = addressField.getText();
                String workingType = workingTypeField.getText();
                String employmentStatus = employmentStatusField.getText();
                double workingHours = Double.parseDouble(workingHoursField.getText());
                String department = departmentField.getText();
```

```
                int yearsOfExperience = Integer.parseInt(yearsOfExperienceField.getText());


                if(teachers.isEmpty()) {
                    Lecturer lecturer = new Lecturer(teacherId, teacherName, address, workingType, employmentStatus, workingHours, department, yearsOfExperience);
                    teachers.add(lecturer);
                    JOptionPane.showMessageDialog(body, "Lecturer has been added successfully", "Lecture Added", JOptionPane.INFORMATION_MESSAGE);
                }
                else {
                    for (Teacher teacher: teachers) {
                        if (teacher.getTeacherId() == teacherId) {
                            if (teacher instanceof Lecturer) {
                                JOptionPane.showMessageDialog(body, "A lecturer with same id has been detected!", "Duplication Error", JOptionPane.WARNING_MESSAGE);
                                return;
                            }
                        }
                        else {
                            Lecturer lecturer = new Lecturer(teacherId, teacherName, address, workingType, employmentStatus, workingHours, department, yearsOfExperience);
                            teachers.add(lecturer);
```

```
                    JOptionPane.showMessageDialog(body, "Lecturer has been added successfully", "Lecture Added", JOptionPane.INFORMATION_MESSAGE);
                    }
                }
            }
        }
        catch(NumberFormatException x)
        {
            JOptionPane.showMessageDialog(body, "Please fill data according to the fields!", "Invalid inputs", JOptionPane.WARNING_MESSAGE);
        }
    }
});

addTutor.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e)
    {
        try
        {
            if (teacherIdField.getText() == "" || teacherNameField.getText() == "" || addressField.getText() == "" || workingTypeField.getText() == "" || employmentStatusField.getText() == "" || departmentField.getText() == "" || workingHoursField.getText() == "" || yearsOfExperienceField.getText() == "" ||
```

```
salaryField.getText() == "" || specializationField.getText() == "" || academicQualificationsField.getText() == "" || performanceIndexField.getText() == "") {
                JOptionPane.showMessageDialog(body, "Please, fill out the empty fields!!!", "Empty Fields !!", JOptionPane.WARNING_MESSAGE);
                return;
            }
            else {
                int teacherId = Integer.parseInt(teacherIdField.getText());
                String teacherName = teacherNameField.getText();
                String address = addressField.getText();
                String workingType = workingTypeField.getText();
                String employmentStatus = employmentStatusField.getText();
                double workingHours = Double.parseDouble(workingHoursField.getText());
                double salary = Double.parseDouble(salaryField.getText());
                String specialization = specializationField.getText();
                String academicQualifications = academicQualificationsField.getText();
                int performanceIndex = Integer.parseInt(performanceIndexField.getText());

                if(teachers.isEmpty()) {
                    Tutor tutor = new Tutor(teacherId, teacherName, address, workingType, employmentStatus, workingHours, salary, specialization, academicQualifications, performanceIndex);
```

```
                teachers.add(tutor);
                JOptionPane.showMessageDialog(body, "Tutor has been added successfully", "Tutor Added", JOptionPane.INFORMATION_MESSAGE);
            }
            else {
                for (Teacher teacher: teachers) {
                    if (teacher.getTeacherId() == teacherId) {
                        if (teacher instanceof Tutor) {
                            JOptionPane.showMessageDialog(body, "A tutor with same id has been detected!", "Duplication Error", JOptionPane.WARNING_MESSAGE);
                            return;
                        }
                    }
                    else {
                        Tutor tutor = new Tutor(teacherId, teacherName, address, workingType, employmentStatus, workingHours, salary, specialization, academicQualifications, performanceIndex);
                            teachers.add(tutor);
                            JOptionPane.showMessageDialog(body, "Tutor has been added successfully", "Tutor added", JOptionPane.INFORMATION_MESSAGE);
                        }
                    }
                }
            }
```

```
            }
            catch(NumberFormatException x)
            {
                JOptionPane.showMessageDialog(body, "Please fill data according to the fields!", "Invalid Inputs !!", JOptionPane.WARNING_MESSAGE);
            }
        }
    });

    gradeAssignments.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e)
        {
            try
            {

                if (teacherIdField.getText() == "" || teacherNameField.getText() == "" || addressField.getText() == "" || workingTypeField.getText() == "" || employmentStatusField.getText() == "" || departmentField.getText() == "" || workingHoursField.getText() == "" || yearsOfExperienceField.getText() == "" || salaryField.getText() == "" || specializationField.getText() == "" || academicQualificationsField.getText() == "" || performanceIndexField.getText() == "") {
                    JOptionPane.showMessageDialog(body, "Please, fill out the empty fields!!!", "Empty Fields !!", JOptionPane.WARNING_MESSAGE);
                    return;
                }
```

```
            else
            {
                int teacherId = Integer.parseInt(teacherIdField.getText());
                int gradedScore = Integer.parseInt(gradedScoreField.getText());
                String department = departmentField.getText();
                int yearsOfExperience = Integer.parseInt(yearsOfExperienceField.getText());
                boolean validityCheck = false;
                if(gradedScore > 100) {
                    JOptionPane.showMessageDialog(body, "Marking limit exceeded. Cannot be above 100!!", "Marking Error", JOptionPane.WARNING_MESSAGE);
                    return;
                }
                else if(gradedScore < 0) {
                    JOptionPane.showMessageDialog(body, "Marking limit subceeded. Cannot be below 0!!", "Marking Error", JOptionPane.WARNING_MESSAGE);
                    return;
                }
                else if(yearsOfExperience <= 5) {
                    JOptionPane.showMessageDialog(body, "Experience Requirement not fulfilled. Must be 5 years or above!!", "Criteria Error", JOptionPane.WARNING_MESSAGE);
                    return;
                }
```

```
                else {
                    if(!teachers.isEmpty()) {
                        for(Teacher teacher: teachers) {
                            if(teacher.getTeacherId() == teacherId) {
                                if(teacher instanceof Lecturer) {
                                    Lecturer lecturer = (Lecturer)teacher;
                                    lecturer.gradeAssignment(gradedScore, department, yearsOfExperience);
                                    if(lecturer.hasGraded() == false) {
                                        JOptionPane.showMessageDialog(body, "Assignment graded successfully!!", "Marks graded", JOptionPane.INFORMATION_MESSAGE);
                                    }
                                    else {
                                        JOptionPane.showMessageDialog(body,"The assignment has not been graded. Error!!", "Grade Error!!", JOptionPane.ERROR_MESSAGE);
                                    }
                                }
                                else {
                                    JOptionPane.showMessageDialog(body,"The assignment couldn't be graded. Teacher Not Found Error!!", "Grade Error!!", JOptionPane.ERROR_MESSAGE);
                                }
                            }
                        }
```

```
                    }
                    else {
                        JOptionPane.showMessageDialog(body,"The assignment couldn't be graded. Teacher Not Found Error!!", "Grade Error!!", JOptionPane.ERROR_MESSAGE);
                        return;
                    }
                }
            }
        }
        catch(NumberFormatException x)
        {
            JOptionPane.showMessageDialog(body, "Please fill data according to the fields!", "Empty Fields !!", JOptionPane.WARNING_MESSAGE);
        }

    }
});

setSalary.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e)
    {
        try
```

```
        {
            if(teacherIdField.getText() == "" || salaryField.getText() == "" || performanceIndexField.getText() == "") {
                JOptionPane.showMessageDialog(body, "Please, fill out the empty fields!!!", "Empty Fields!!", JOptionPane.WARNING_MESSAGE);
                return;
            }
            else {
                int teacherId = Integer.parseInt(teacherIdField.getText());
                double salary = Double.parseDouble(salaryField.getText());
                int performanceIndex = Integer.parseInt(performanceIndexField.getText());

                if(performanceIndex > 10) {
                    JOptionPane.showMessageDialog(body, "Performance Index limit exceeded. Cannot be above 10!!", "Index Error", JOptionPane.WARNING_MESSAGE);
                    return;
                }
                else if(performanceIndex < 0) {
                    JOptionPane.showMessageDialog(body," Performance Index limit subceeded. Cannot be below 0!!", "Index Error", JOptionPane.WARNING_MESSAGE);
                    return;
                }
                else {
```

```
                if(!teachers.isEmpty()){
                    for(Teacher teacher: teachers) {
                        if(teacher.getTeacherId() == teacherId) {
                            if(teacher instanceof Tutor) {


                                Tutor tutor = (Tutor) teacher;
                                tutor.setSalary(salary,performanceIndex);
                                salary = tutor.getSalary();
                                if(performanceIndex < 5 && tutor.getWorkingHours() < 20) {
                                    JOptionPane.showMessageDialog(body, "New Salary Requirements not fulfilled.\nPerformance Index must be above 5 or Working Hours must be abpve 20 hours!!", "Salary Not Approved", JOptionPane.INFORMATION_MESSAGE);
                                }
                                else {
                                    JOptionPane.showMessageDialog(body,"Salary set successfull with salary being " + salary + "!", "Setting Salary", JOptionPane.INFORMATION_MESSAGE);
                                }
                            }
                        }
                    }
                }
                else {
```

```
                    JOptionPane.showMessageDialog(body,"The salary couldn't be set. Teacher Not Found Error!!", "Grade Error!!", JOptionPane.ERROR_MESSAGE);
                    return;
                }
            }
        }
        catch(NumberFormatException x)
        {
            JOptionPane.showMessageDialog(body, "Please fill data according to the fields!", "Empty Fields !!", JOptionPane.WARNING_MESSAGE);
        }
    }
});

removeTutor.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e)
    {
        try
        {
            if(teacherIdField.getText() == "") {
                JOptionPane.showMessageDialog(body,"Please, fill out the empty Teacher Id field!!!", "Empty Fields!!",JOptionPane.WARNING_MESSAGE);
```

```
            return;
        }
        else {
            int teacherId = Integer.parseInt(teacherIdField.getText());

            for(Teacher teacher: teachers) {
                if(teacher.getTeacherId() == teacherId) {
                    if(teacher instanceof Tutor) {

                        Tutor tutor = (Tutor) teacher;
                        boolean isCertified = tutor.isCertified();
                        if(isCertified == false) {
                            tutor.removeTutor();
                            JOptionPane.showMessageDialog(body, "The tutor has been successfully removed", "Tutor Removed", JOptionPane.INFORMATION_MESSAGE);
                        }
                        else {
                            JOptionPane.showMessageDialog(body, "Certified tutors cannot be removed","Remove Tutor Error",JOptionPane.INFORMATION_MESSAGE);
                        }
                    }
                }
            }
```

```
            }
        }

        catch(NumberFormatException x)
        {
            JOptionPane.showMessageDialog(body, "Please fill data according to the fields!", "Invalid ID",JOptionPane.WARNING_MESSAGE);
        }
    }
});

displayLecturer.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e)
    {
        try
        {
            int teacherId = Integer.parseInt(teacherIdField.getText());
            String teacherName = teacherNameField.getText();
            String address = addressField.getText();
            String workingType = workingTypeField.getText();
            String employmentStatus = employmentStatusField.getText();
            int workingHours = Integer.parseInt(workingHoursField.getText());
            String department = departmentField.getText();
```

```
                int yearsOfExperience = Integer.parseInt(yearsOfExperienceField.getText());


                JOptionPane.showMessageDialog(body,"Teacher ID: "+ teacherId + "\nTeacher Name:" + teacherName + "\nAddress:" + address + "\nWorking Type: "+ workingType + "\nEmployment Status: " + employmentStatus + "\nWorking Hours: " + workingHours + "\nDepartment: " + department + "\nYears of Experience: " + yearsOfExperience);
            }
            catch(NumberFormatException a)
            {
                JOptionPane.showMessageDialog(body, "Please fill data according to the fields!", "Invalid ID",JOptionPane.WARNING_MESSAGE);
            }
        }
    });


    displayTutor.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e)
        {
            try
            {
                int teacherId = Integer.parseInt(teacherIdField.getText());
                String teacherName = teacherNameField.getText();
```

```
                String address = addressField.getText();
                String workingType = workingTypeField.getText();
                String employmentStatus = employmentStatusField.getText();
                int workingHours = Integer.parseInt(workingHoursField.getText());
                double salary = Double.parseDouble(salaryField.getText());
                String specialization = specializationField.getText();
                String academicQualifications = academicQualificationsField.getText();
                int performanceIndex = Integer.parseInt(performanceIndexField.getText());

                JOptionPane.showMessageDialog(body,"Teacher ID: "+ teacherId + "\nTeacher Name:" + teacherName + "\nAddress:" + address + "\nWorking Type: "+ workingType + "\nEmployment Status: " + employmentStatus + "\nWorking Hours: " + "\nSalary: "+ salary + "\nSpecialization: "+ specialization + "\nAcademic Qualification: " + academicQualifications + "\nPerformance Index: " + performanceIndex);
            }
            catch(NumberFormatException a)
            {
                JOptionPane.showMessageDialog(body, "Please fill data according to the fields!", "Invalid ID",JOptionPane.WARNING_MESSAGE);
            }
        }
    });

    // ActionListener for the clear button
```

```
        clear.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Clearing the text fields
                teacherIdField.setText("");
                teacherNameField.setText("");
                addressField.setText("");
                workingTypeField.setText("");
                employmentStatusField.setText("");
                workingHoursField.setText("");
                departmentField.setText("");
                yearsOfExperienceField.setText("");
                gradedScoreField.setText("");
                salaryField.setText("");
                specializationField.setText("");
                academicQualificationsField.setText("");
                performanceIndexField.setText("");
            }
        });
    }


}
```