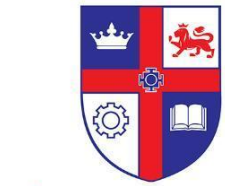
LONDON METROPOLITAN UNIVERSITY



Islington college
(इस्लिङ्टन कलेज)

CS4001NI Programming

30% Individual Coursework

2023-24 Autumn

Student Name: MD samim rain

London Met ID: 23050299

College ID: NP01CP4A230163

Group: C7

Assignment Due Date: Friday, May 10, 2024

Assignment Submission Date: Friday, May 10, 2024

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

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1. About the coursework

In this assignment developing a real-world problem program that can simulate a class to represent a teacher with two subclasses to represent a lecturer and a tutor. This program will allow users to for school, college, and university. However, this assignment is about creating a real-world problem scenario, and the main class, the Teacher (parent class) will have attributes and two subclasses Lecturer and Tutor (child class) will have attributes and each child class extends to the parent class.

The objective of this assignment is to extend a previous project by incorporating object-oriented principles in Java to model real-world scenarios involving teachers. In this assignment, we aim to create a class structure that represents different types of teachers, such as Lecturers and Tutors, using inheritance. Additionally, we will develop a graphical user interface (GUI) for a system that manages teacher details, leveraging an Array List to store the information efficiently. The assignment not only focuses on the implementation of the program but also requires a comprehensive report detailing the design choices, challenges faced during development, and the solutions employed to address them. Through this project, we seek to demonstrate proficiency in object-oriented programming concepts, GUI development, and problem-solving skills in Java. This report will provide insights into the design, implementation, and testing phases of the program, highlighting key features and their contributions to solving the specified problem scenario.

1.2. Introduction to Java

Java is a popular programming language, easy-to-learn. Java is a class-based high-level object-oriented programming (OOP) language. Java is widely used in various applications in the real world. It was developed by 'James Gosling' Sun Microsystem and released in 1995. It is a platform-independent structured programming language that is widely used to write applications and operating systems like Windows, and like a Oracle database. Java was originally designed for interactive television but is currently, used in mobile devices, games, etc.

2. Tools used.

2.1. Blue J

Blue J is a software platform because the Blue J tool is a practical software platform for Java code practice. It is easy to use and helps to learn Java programming. This software application helps to provide a more precise interface for creating projects and coding in Java and this software is used for free no pay money and no pay for download. It uses friendly weather and a Java environment.

2.2. Ms word.

MS Word is a popular platform word processing program that allows the creation and editing of professional documents. MS Word created by Microsoft and released in 1983, is a widely used word processing software in the English language and many other languages, offering features like spell check, formatting, and image insertion. It is popular among professionals, students, and individuals for creating various written materials, compatible with both Windows and Mac operating systems. It is used for free no pay-for-money.

2.3. Draw.io

Draw.io tools used to static structural components of class diagrams make signs and meanings of attributes of a class. We can use it to create diagrams such as flowcharts, wireframes, network diagrams, and class diagrams. Draw.io is an online tool and no need to pay money.

3. Class diagrams.

A class diagram static structure diagram that describes the structure of a system's classes, their attributes, and methods the class diagram in Java like a map that shows different parts of a program is connected to object relationships because it helps us

organize our code and makes it easier to understand. It is a graphical tool for creating and visualizing object-oriented systems.

3.1. Teacher class diagram.

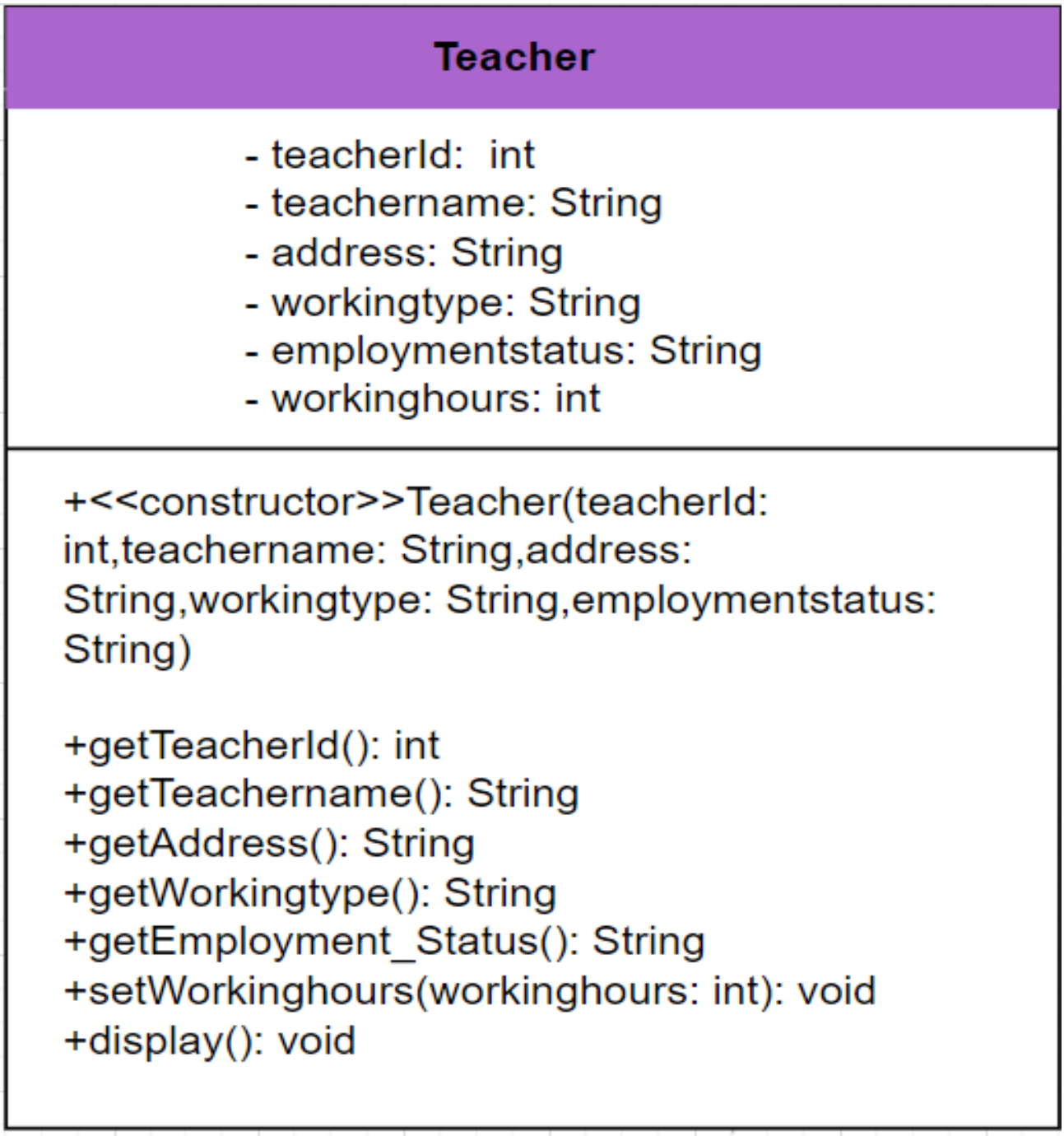


Figure 1 : Teacher class.

3.2. Lecturer class diagram.

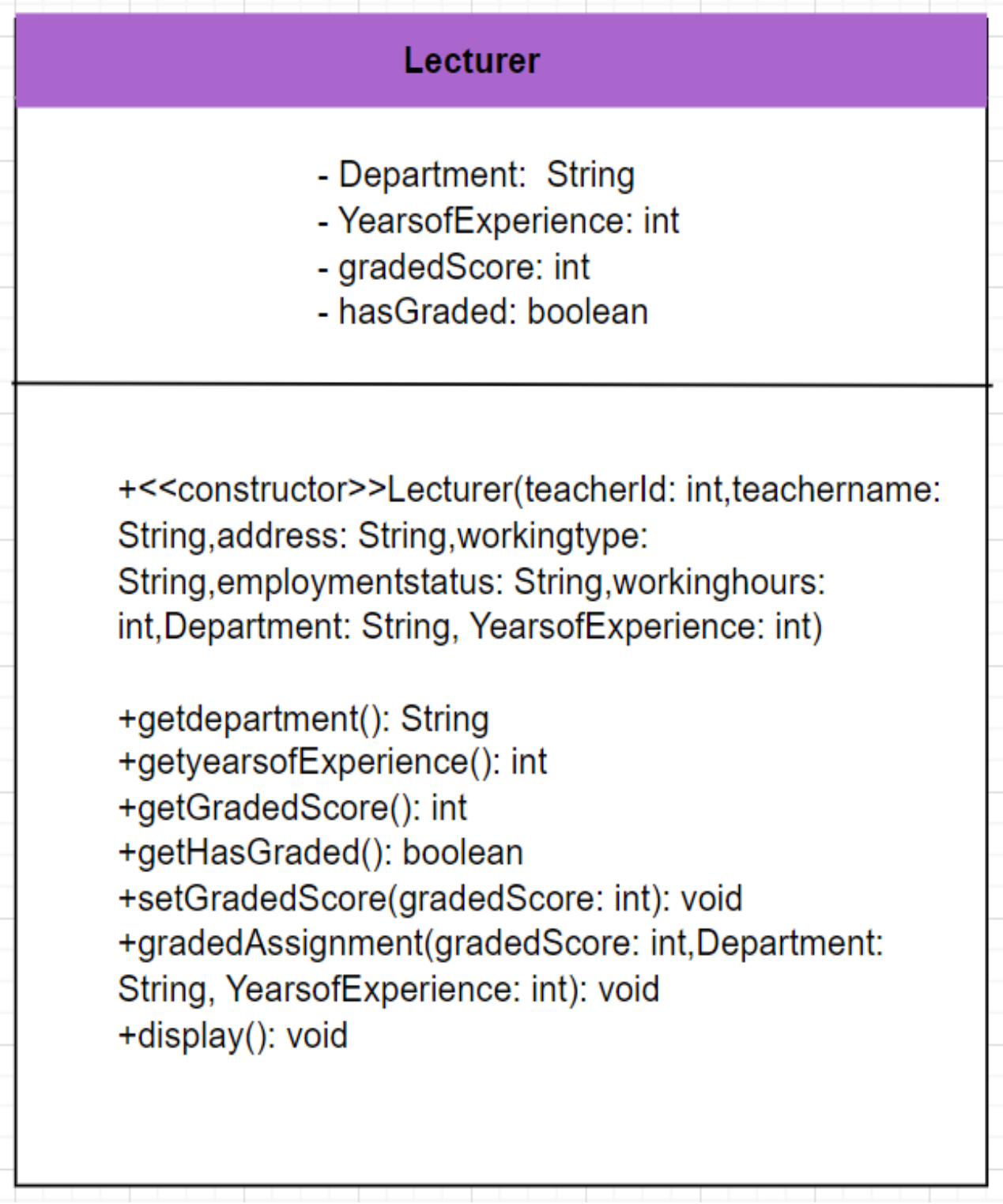


Figure 2 : Lecturer class.

3.3. Tutor class diagram

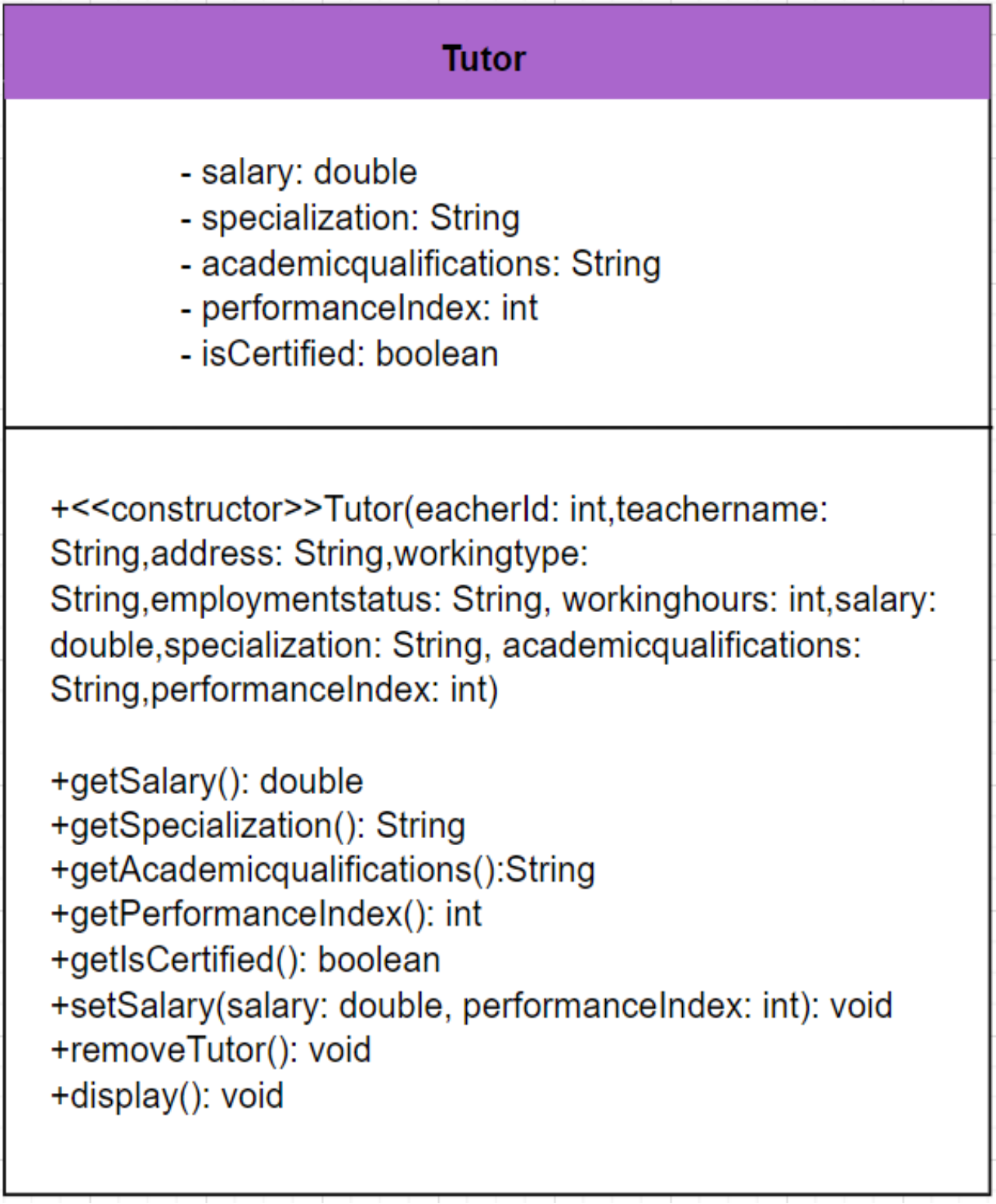


Figure 3 : Tutor class.

3.4. Teacher GUI class diagram

Figure 4 : Teacher GUI.

3.5. Relationship of class diagram.

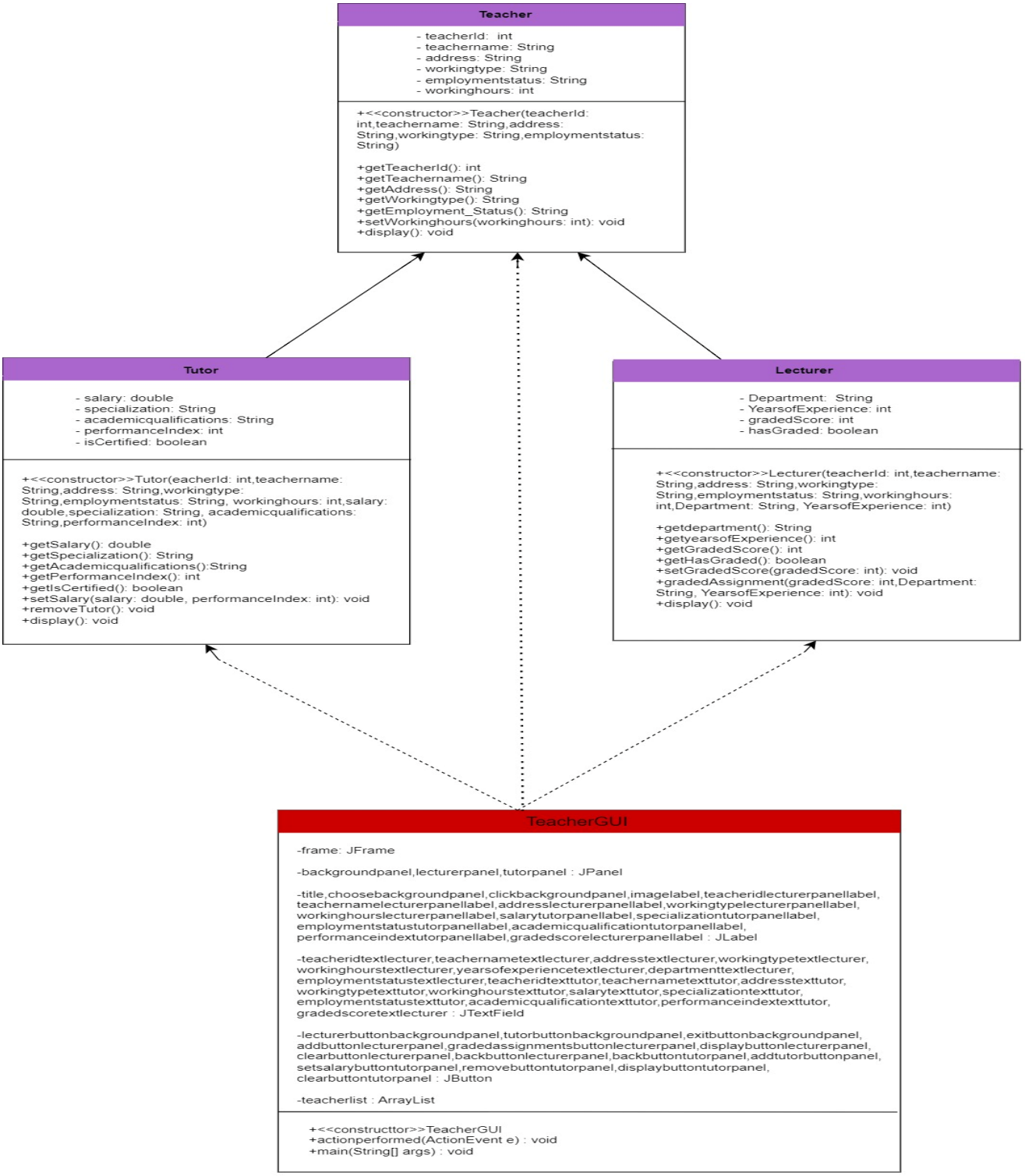


Figure 5 : Relationship of class diagram.

4. Pseudocode of Teacher GUI.

IMPORT javax.swing.JFrame

IMPORT javax.swing.JLabel

IMPORT javax.swing.JTextField

IMPORT javax.swing.JPanel

IMPORT javax.swing.border.Border

IMPORT java.awt.Color

IMPORT java.awt.Font

IMPORT javax.swing.JOptionPane

IMPORT javax.swing.BorderFactory

IMPORT javax.swing.ImageIcon

IMPORT javax.swing.JButton

IMPORT java.awt.event.ActionListener

IMPORT java.awt.event.ActionEvent

IMPORT java.util.ArrayList

CREATE CLASS name TeacherGUI that implements ActionListener

DO

DECLARE frame, backgroundpanel, lecturerpanel, tutorpanel AS JFrame, JPanel

DECLARE title, choosebackgroundpanel, managementlabel, clickbackgroundpanel, imagelabel, teacheridlecturerpanellabel, ... AS JLabel

DECLARE teacheridtextlecturer, teachernametextlecturer, ... AS JTextField

DECLARE lecturerbuttonbackgroundpanel, tutorbuttonbackgroundpanel, exitbuttonbackgroundpanel, ... AS JButton

DECLARE teacherlist AS ArrayList<Teacher>

CONSTRUCTOR TeacherGUI

DO

INITIALIZE teacherlist AS new ArrayList<Teacher>

CREATE all required components

SET add to the bounds and their components all required added

END DO

METHOD actionPerformed(e: ActionEvent):

DO

IF e.getSource() EQUALS lecturerbuttonbackgroundpanel THEN

Remove backgroundpanel from frame

Add lecturerpanel to frame

Revalidate frame

Repaint frame

END IF

IF e.getSource() EQUALS tutorbuttonbackgroundpanel THEN

Remove backgroundpanel from frame

Add tutorpanel to frame

Revalidate frame

Repaint frame

END IF

IF e.getSource() EQUALS backbuttonlecturerpanel THEN

Add backgroundpanel to frame

Remove lecturerpanel from frame

Revalidate frame

Repaint frame

END IF

IF e.getSource() EQUALS backbuttontutorpanel THEN

Add backgroundpanel to frame

Remove tutorpanel from frame

Revalidate frame

Repaint frame

END IF

IF e.getSource() EQUALS exitbuttonbackgroundpanel THEN

Terminate the program

END IF

IF e.getSource() EQUALS addbuttonlecturerpanel THEN

TRY

Parse input values to integer or string variables

IF any of the required fields are empty THEN

Display error message indicating empty fields

```
        ELSE
            Check if teacher ID already exists in teacherlist
            IF teacher ID already exists THEN
                Display error message indicating duplicate ID
            ELSE
                Create a new Lecturer object with input values
                Add the new lecturer to the teacherlist
                Display success message
                Clear all text fields
            END IF
        END IF
    CATCH NumberFormatException
        Display error message indicating invalid input values
    END TRY
END IF

IF e.getSource() EQUALS gradedassignmentsbuttonlecturerpanel THEN
    TRY
        Parse input values to integer variables
        IF any of the required fields are empty THEN
            Display error message indicating empty fields
        ELSE
            Search for matching lecturer using teacher ID
```

```
            IF lecturer found THEN
                Update graded score for lecturer
                Display success message
                Clear all text fields
            ELSE
                Display error message indicating lecturer not found
            END IF
        END IF
    CATCH NumberFormatException
        Display error message indicating invalid input values
    END TRY
END IF

IF e.getSource() EQUALS displaybuttonlecturerpanel THEN
    TRY
        Parse input value to integer variable
        IF teacher ID field is empty THEN
            Display error message indicating empty field
        ELSE
            Search for matching lecturer using teacher ID
            IF lecturer found THEN
                IF graded score is zero THEN
                    Display lecturer details without graded score
```

```
                ELSE
                    Display lecturer details with graded score
                END IF
            ELSE
                Display message indicating no lecturer details found
            END IF
            Clear teacher ID text field
        END IF
    CATCH NumberFormatException
        Display error message indicating invalid input value in Teacher ID field
    END TRY
END IF

IF e.getSource() EQUALS clearbuttonlecturerpanel THEN
    Clear all text fields
END IF

DO
    IF e.getSource() EQUALS addtutorbuttonpanel THEN
    TRY
        Parse input values to integer or string variables
```

```
            IF teacherId IS EMPTY OR teacherName IS EMPTY OR address IS
EMPTY OR workingType IS EMPTY OR
              workingHours IS EMPTY OR salary <= 0.0 OR specialization IS EMPTY
OR employmentStatus IS EMPTY
              OR academicQualifications IS EMPTY OR performanceIndex IS EMPTY
THEN
                DISPLAY Error message indicating empty fields
              ELSE
                CREATE Boolean teacherIdExists SET TO FALSE
                FOR EACH Tutor IN teacherlist DO
                  IF Tutor.getTeacherId() EQUALS teacherId THEN
                    SET teacherIdExists TO TRUE
                    BREAK FROM LOOP
                  END IF
                END FOR

                IF teacherIdExists THEN
                  DISPLAY Error message indicating duplicate teacher ID
                ELSE
                  CREATE Tutor tutor ASSIGN NEW Tutor(teacherId, teacherName,
address,
                      workingType, employmentStatus, workingHours, salary,
specialization,
```

academicQualifications, performanceIndex)

ADD tutor TO teacherlist

DISPLAY Success message indicating tutor added successfully

CLEAR all text fields related to tutor information

END IF

END IF

CATCH NumberFormatException

DISPLAY Error message indicating invalid input values

END TRY

END IF

IF e.getSource() EQUALS setsalarybuttontutorpanel THEN

TRY

Parse input values to integer

IF teacherId IS EMPTY OR salary <= 0.0 OR performanceIndex IS EMPTY THEN

DISPLAY Error message indicating empty fields

ELSE

CREATE Boolean found SET TO FALSE

FOR EACH Teacher IN teacherlist DO

IF Teacher IS INSTANCE OF Tutor AND Teacher.getTeacherId() EQUALS teacherId THEN

CREATE Tutor tutor ASSIGN TO (Tutor) Teacher

SET found TO TRUE

tutor.setSalary(salary, performanceIndex)

DISPLAY Success message indicating salary set successfully

BREAK FROM LOOP

END IF

END FOR

IF NOT found THEN

DISPLAY Error message indicating tutor not found

END IF

CLEAR text fields

END IF

CATCH NumberFormatException

DISPLAY Error message indicating invalid input values

END TRY

END IF

IF e.getSource() EQUALS removebuttontutorpanel THEN

```
TRY
  CREATE Integer teacherId FROM
      Parse input values to integer

  IF teacherId IS EMPTY THEN
    DISPLAY Error message indicating empty teacher ID
  ELSE
    CREATE Boolean teacherFound SET TO FALSE
    CREATE Tutor tutorToRemove SET TO NULL
    FOR EACH Teacher IN teacherlist DO
      IF Teacher.getTeacherId() EQUALS teacherId THEN
        IF Teacher IS INSTANCE OF Tutor THEN
          CREATE Tutor tutor ASSIGN TO (Tutor) Teacher

          IF tutor.getIsCertified() THEN
            DISPLAY Error message indicating certified tutors cannot be
removed
          ELSE
            SET teacherFound TO TRUE
            SET tutorToRemove TO tutor
            BREAK FROM LOOP
          END IF
        END IF
```

```
            END IF
        END FOR

        IF teacherFound THEN
            tutorToRemove.removeTutor()
            REMOVE tutorToRemove FROM teacherlist
            DISPLAY Success message indicating tutor removed successfully
        ELSE
            DISPLAY Error message indicating tutor not found
        END IF
    END IF
  CATCH NumberFormatException
    DISPLAY Error message indicating invalid input value for teacher ID
  END TRY
END IF

IF e.getSource() EQUALS displaybuttontutorpanel THEN
  TRY
    CREATE Integer teacherId FROM
          Parse input values to integer

    IF teacherId IS EMPTY THEN
      DISPLAY Error message indicating empty teacher ID
```

```
            ELSE
                CREATE Boolean tutorFound SET TO FALSE
                FOR EACH Teacher IN teacherlist DO
                    IF Teacher IS INSTANCE OF Tutor AND Teacher.getTeacherId()
EQUALS teacherId THEN
                        CREATE Tutor tutor ASSIGN TO (Tutor) Teacher
                        SET tutorFound TO TRUE


                        DISPLAY Tutor details in a dialog box


                        BREAK FROM LOOP
                    END IF
                END FOR

                IF NOT tutorFound THEN
                    DISPLAY Error message indicating tutor not found
                END IF


                CLEAR text fields
            END IF
        CATCH NumberFormatException
            DISPLAY Error message indicating invalid teacher ID
        END TRY
```

END IF

IF e.getSource() EQUALS clearbuttontutorpanel THEN

CLEAR all text fields related to tutor information

END IF

END DO

END DO

5. Description of Button.

5.1. Lecturer Button background panel.

when the lecturerbuttonbackgroundpanel button is clicked, this method dynamically updates the content displayed in the frame, switching from a background panel to a lecturer panel, and ensures the GUI reflects these changes accurately.

5.2. Tutor button background panel.

when the tutorbuttonbackgroundpanel button is clicked, this method dynamically updates the content displayed in the frame, transitioning from a background panel to a tutor panel. This ensures that the graphical user interface accurately reflects the user's interactions and displays relevant content accordingly.

5.3. Exit button background panel.

when the exitbuttonbackgroundpanel button is clicked, this code immediately shuts down the application, effectively exiting the program. This button serves as a means for users to gracefully close the application when they no longer require its services.

5.4. Add button lecturer panel.

This button is likely used to add a new lecturer to the system. When clicked, it attempts to gather data from text fields (like teacher ID, name, address, etc.). It validates whether all necessary fields are filled. It checks if the teacher ID already exists in the system to ensure uniqueness. If all validations pass, it creates a new Lecturer object and adds it to a list of teachers (teacherlist). It displays appropriate feedback messages based on the success or failure of the operation. Finally, it clears the input text fields.

5.5. Graded assignment button lecturer panel.

This button seems related to assigning grades to assignments handled by a lecturer.

Upon clicking, it retrieves data from relevant text fields (teacher ID, graded score, department, years of experience). It checks if all necessary fields are filled. It then iterates over the list of teachers to find the one matching the provided teacher ID. If found (and if the teacher is an instance of Lecturer), it calls a method (gradedAssignment) on the Lecturer object to record the graded assignment along with department and years of

experience. It displays appropriate feedback messages based on the success or failure of finding the teacher. Finally, it clears the input text fields.

Add lecturer and graded assignment buttons seem to perform input validation, error handling, and interaction with a list of teachers (teacherlist) containing instances of the Teacher class, possibly including subtypes like Lecturer. They also provide feedback to the user through dialog boxes (JOptionPane) indicating the success or failure of the performed actions.

5.7. Display lecturer button panel.

this button is responsible for displaying details of a lecturer based on their ID. If a matching lecturer is found, it presents their information including any graded score they may have. Otherwise, it notifies the user that no details are available for the provided ID.

5.8. Clear lecturer button panel.

When the user clicks the "Clear" button is responsible for resetting all text fields in the lecturer panel, allowing users to easily clear previously entered data when needed.

5.9. Back button lecturer panel.

when the user "Back" button is clicked, it navigates the user back to the background panel, removing the current lecturer panel from the frame and showing the background panel instead. This action likely allows users to navigate between different sections or functionalities within the GUI application.

5.10. Add tutor button panel.

When the user clicks the “Add Tutor" button is responsible for collecting information from input fields, creating a new tutor object, adding it to the list of teachers, and providing feedback to the user about the success or failure of the operation. It also handles input validation and clears the input fields after adding a tutor.

5.11. Set salary tutor button panel.

When the user clicks the "Set Salary" button is responsible for collecting information from input fields, finding the corresponding tutor in the list of teachers, setting their salary based on provided inputs, and providing feedback to the user about the success or failure of the operation. It also handles input validation and clears the input fields after setting the salary.

5.12. Remove button tutor panel.

When the user clicks the "Remove Tutor" button is responsible for collecting the teacher ID from the input field, finding the corresponding tutor in the list of teachers, removing them from the system, and providing appropriate feedback to the user about the success or failure of the operation. It also handles input validation to ensure that the teacher ID is a numeric value.

5.13. Display button tutor panel.

When the user clicks the "Display" button is responsible for collecting the teacher ID from the input field, finding the corresponding tutor in the list of teachers, displaying their details, and providing appropriate feedback to the user about the success or failure of the operation. It also handles input validation to ensure that the teacher ID is a numeric value.

5.14. Clear button tutor panel.

When the user clicks the "Clear" button is responsible for resetting all text fields in the tutor panel, allowing users to easily clear previously entered data when needed. It provides a simple way to reset the form for new entries or when users want to start over.

5.15. Back button tutor panel.

when the "Back" button is clicked, it navigates the user back to the background panel, removing the current tutor panel from the frame and showing the background panel instead. This action likely allows users to navigate between different sections or functionalities within the GUI application.

Overall, these buttons handle various functionalities such as navigation between panels, adding, displaying, grading, clearing, adding tutor details, setting salary, displaying, removing tutor, and clearing. They ensure the smooth interaction of users with the GUI application for managing lecturer and tutor information.

6. Testing.

6.1. Test 1.

Test that the program can be compiled and run using the command prompt, including a screenshot like Figure 1 from the command prompt learning aid.

Objective	To inspect that the program can be compiled and run using the command prompt.
Action	Open Command Prompt. Use javac java to compile and run the program and reveal the background panel.
Expected Results	The TeacherGUI was expected to be opened.
Actual Result	The TeacherGUI was revealed as expected.
Conclusion	The test was successful.

Table 1: Table of Test 1

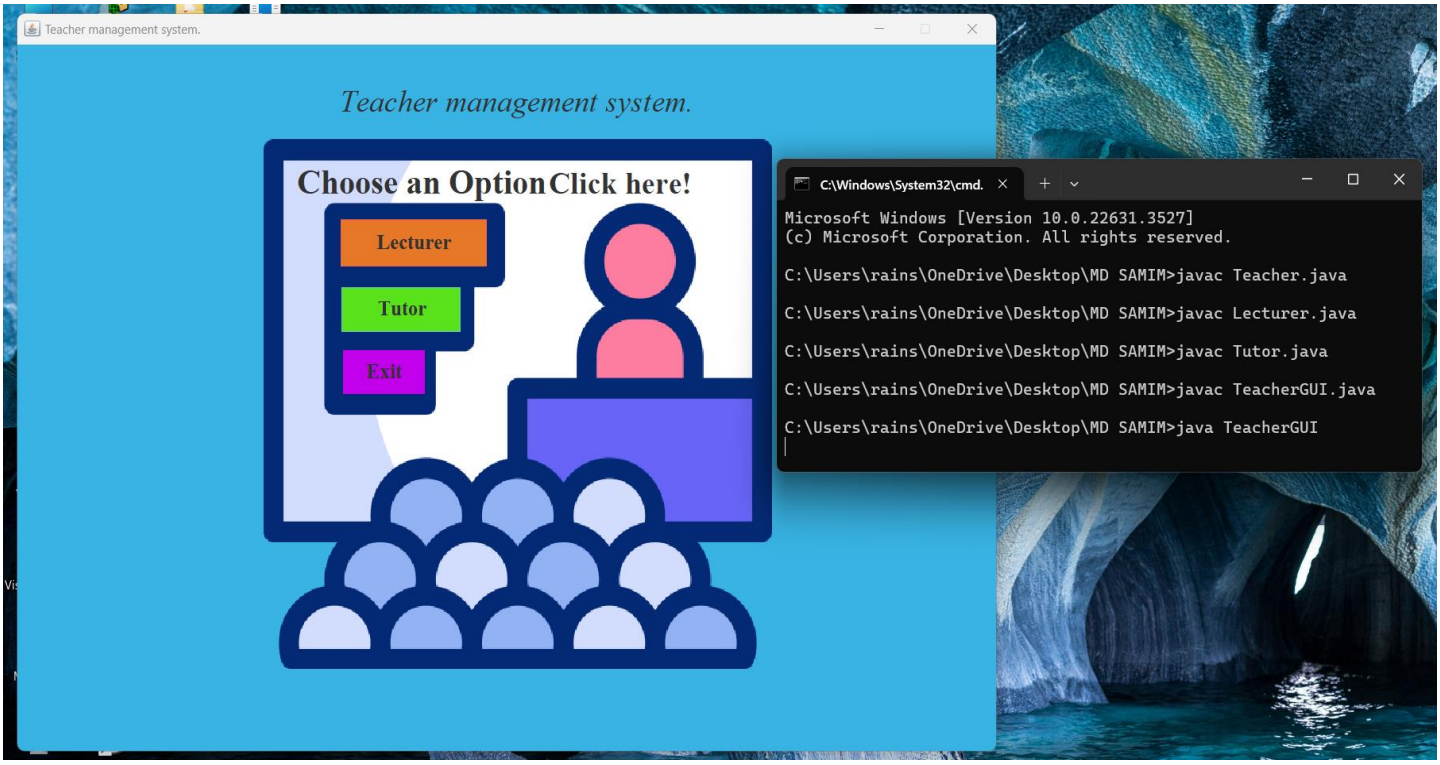


Figure 6: Teacher GUI opened in Command Prompt.

6.2. Test 2.

Evidences should be shown of:

a. Add the Lecturer.

b. Add the Tutor.

c. Grade Assignments from Lecturer.

d. Set the salary.

e. Remove the tutor.

Objectives	Evidences should be shown of: a. Add Lecturer b. Add Tutor c. Grade Assignment d. Set Salary e. Remove Tutor
Action	Step 1: All the fields were filled and Add Lecturer was pressed. Step 2: All the fields were filled, and Add Tutor was pressed. Step 3: All the fields were filled, and Grade assignment button was pressed. Step 4: All the fields were filled, and Set Salary was pressed. Step 5: Remove Tutor button was pressed.
Expected Result	All the evidences should to be shown to work Properly.
Actual Result	All the evidences should to be shown worked Properly.
Conclusion	The test was successful.

Table 2: Table of Test 2.

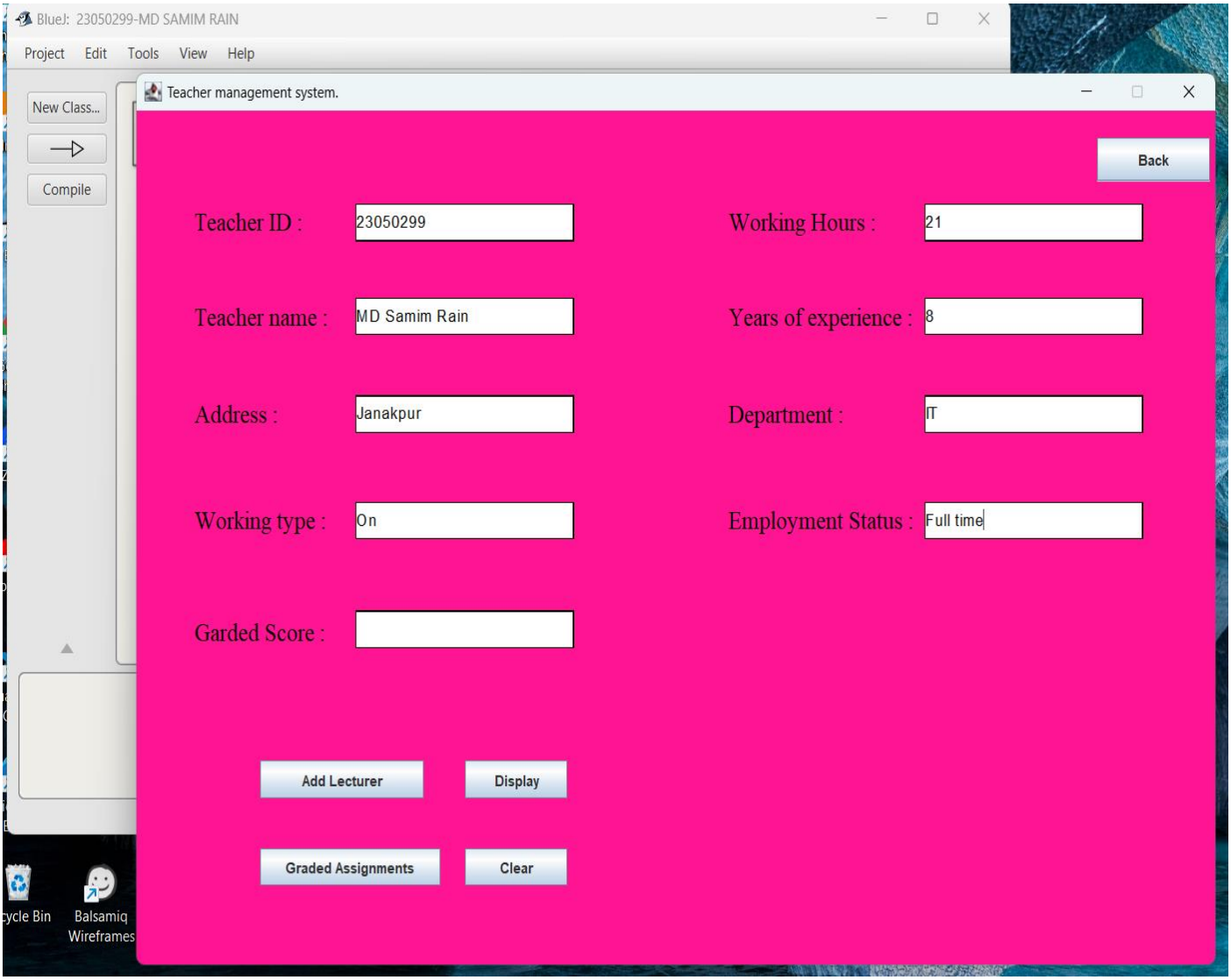


Figure 7: Teacher text fields of lecturer.

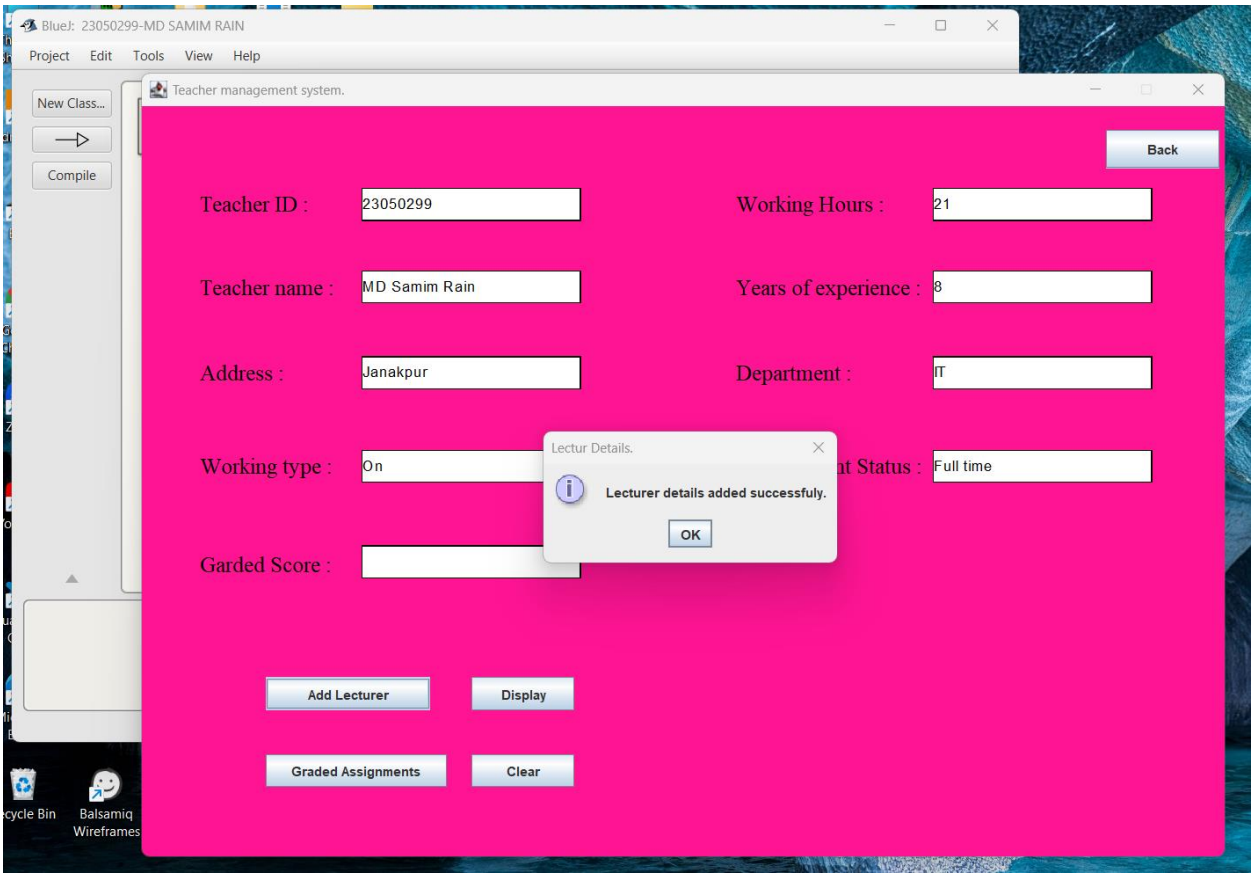


Figure 8 : Lecturer text fields added.

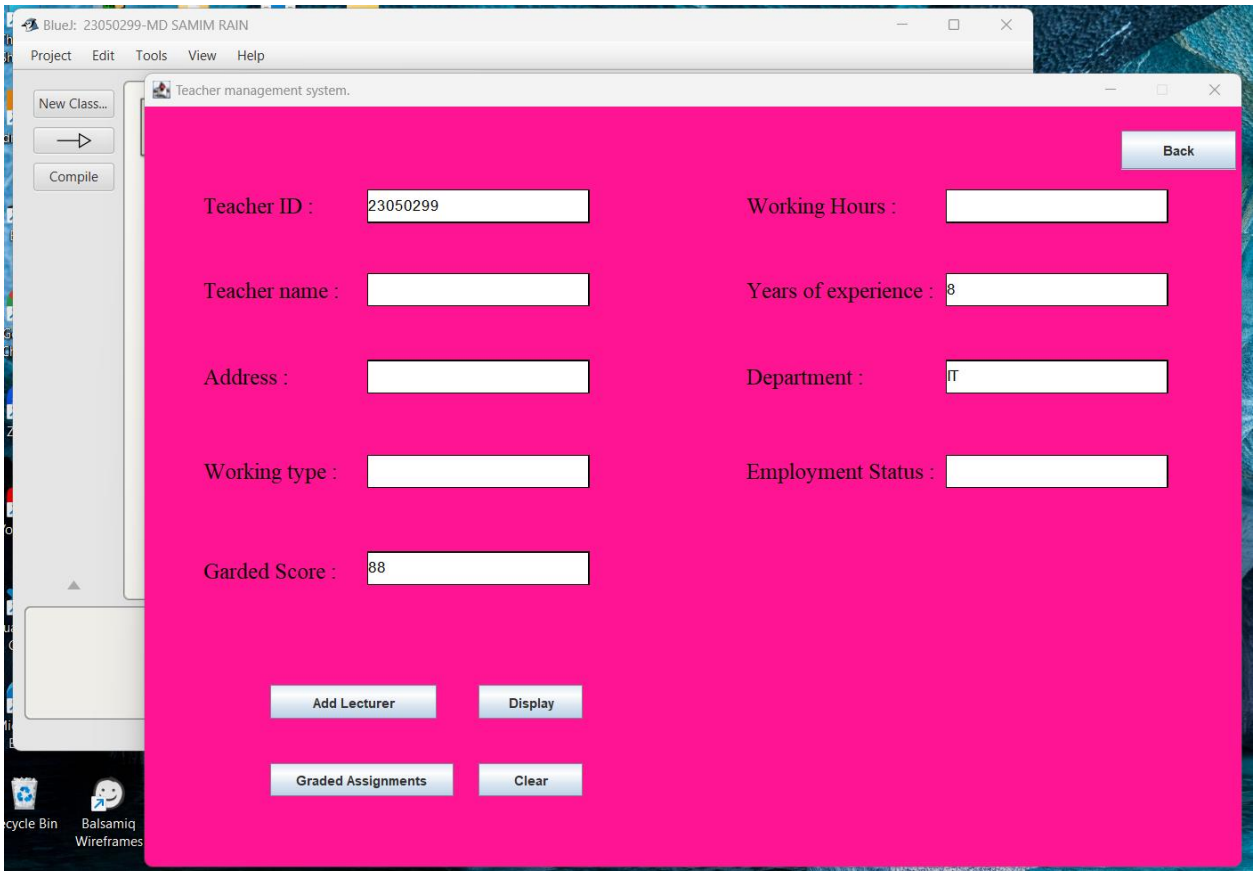


Figure 9 : Lecturer of graded assignment of text fields.

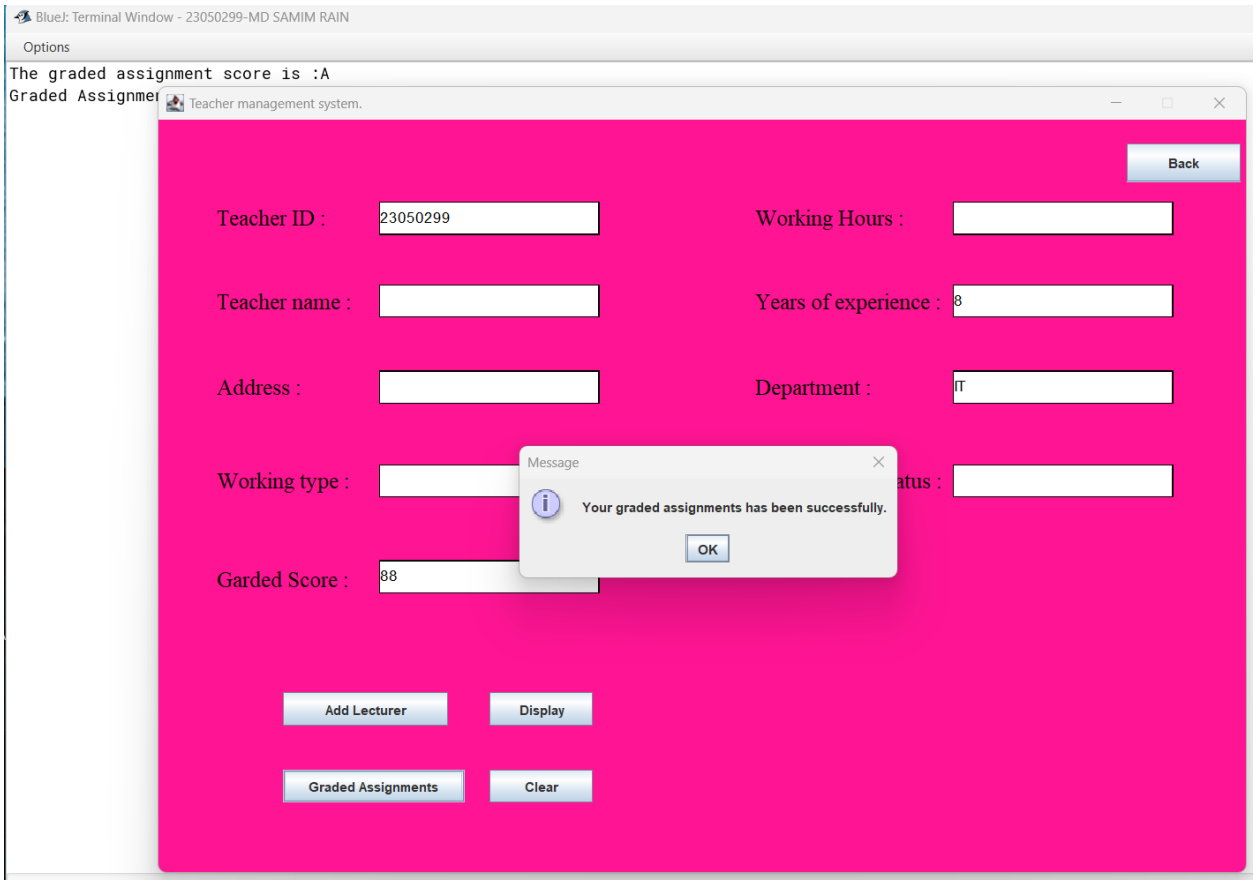


Figure 10: Graded assignment of added successfully.

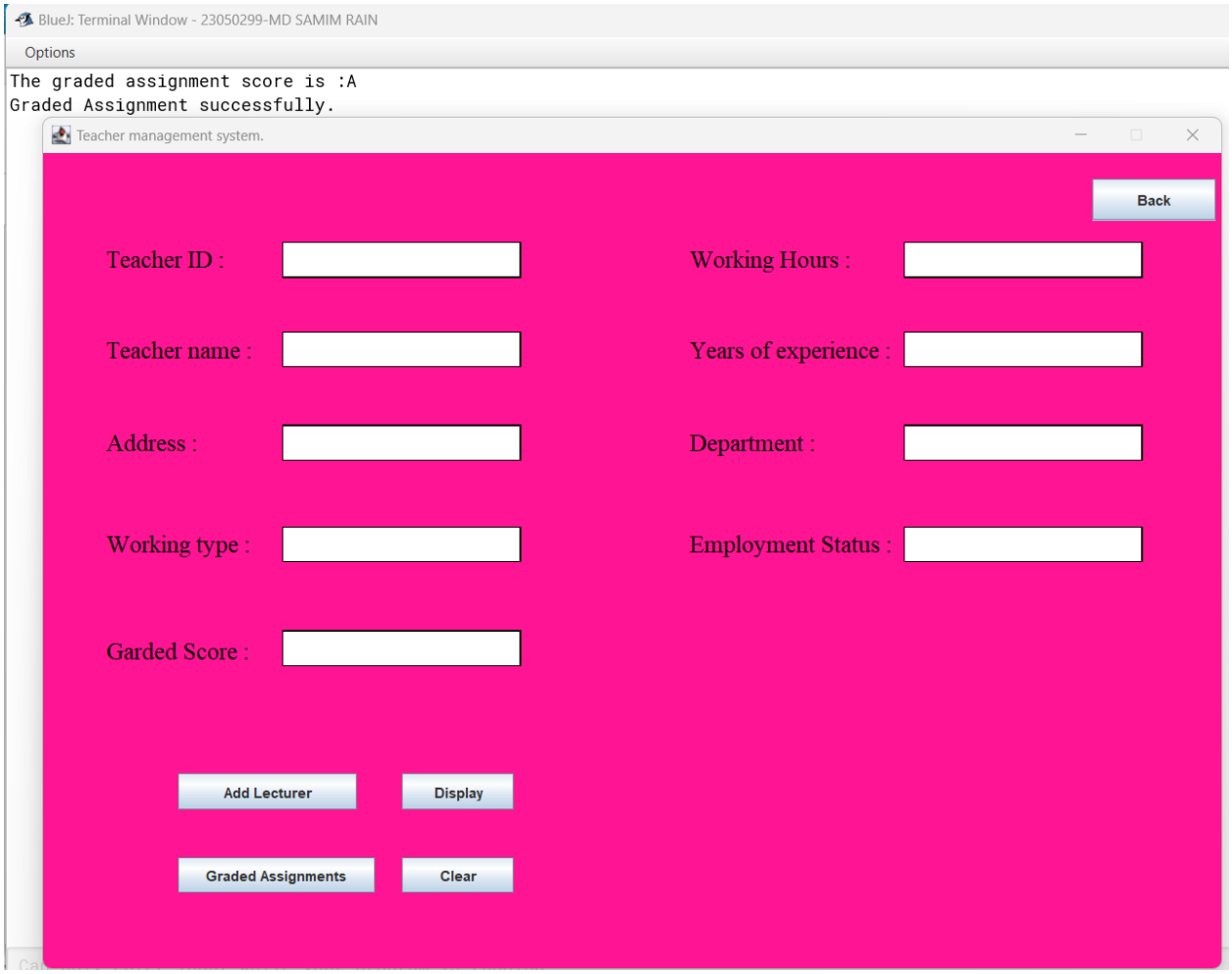


Figure 11: Graded assignment score shown.

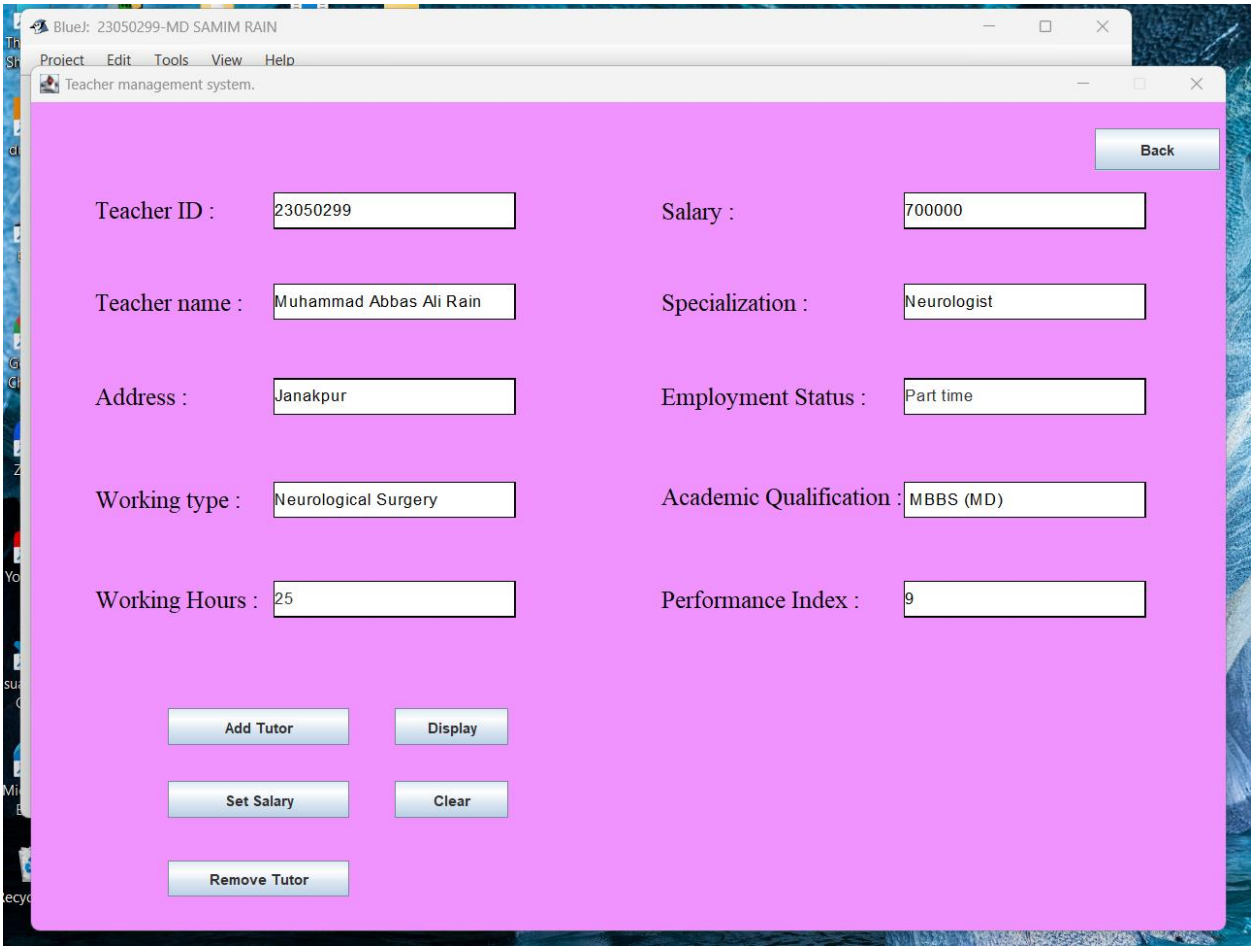


Figure 12: Add tutor text fields.

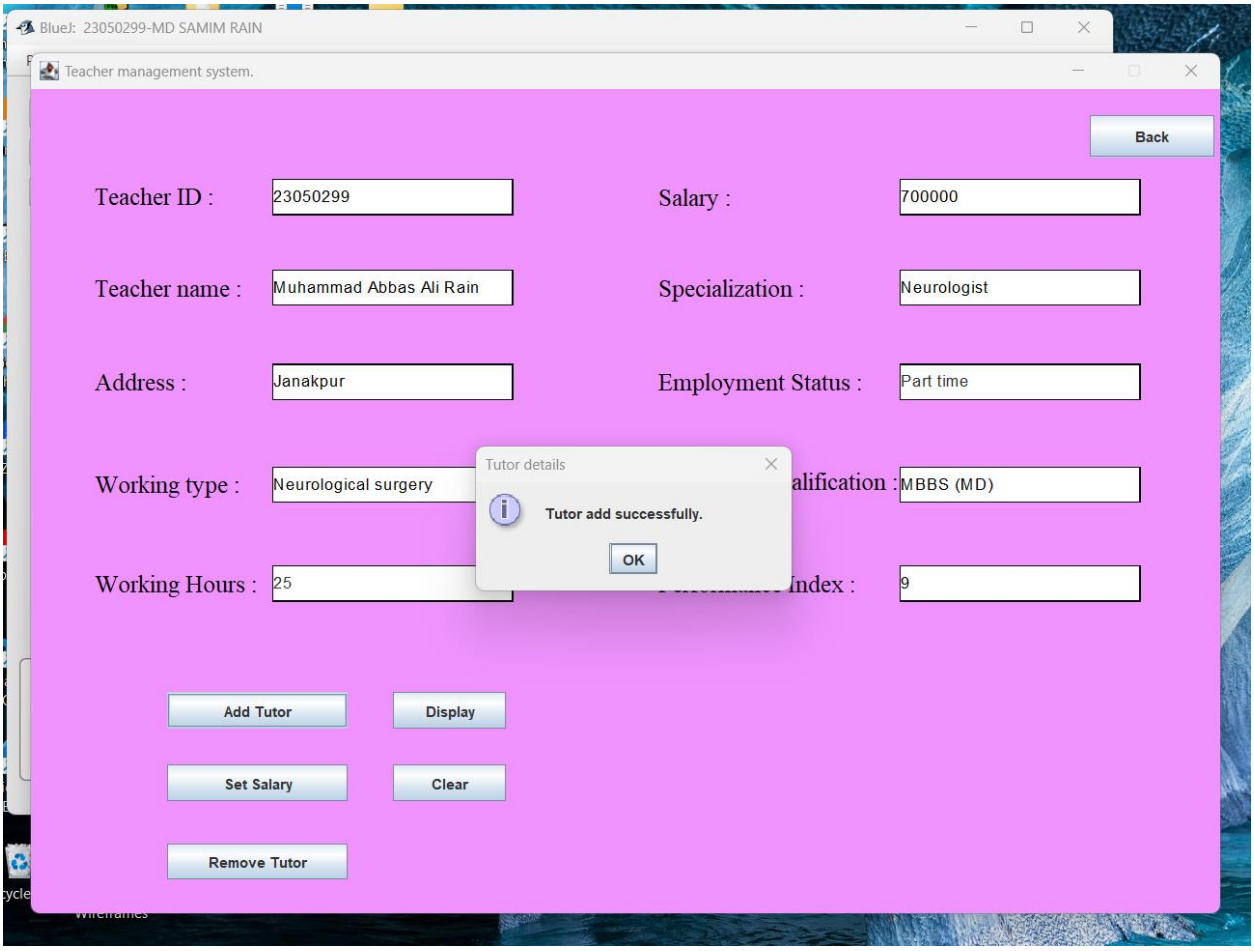


Figure 13: Add tutor details.

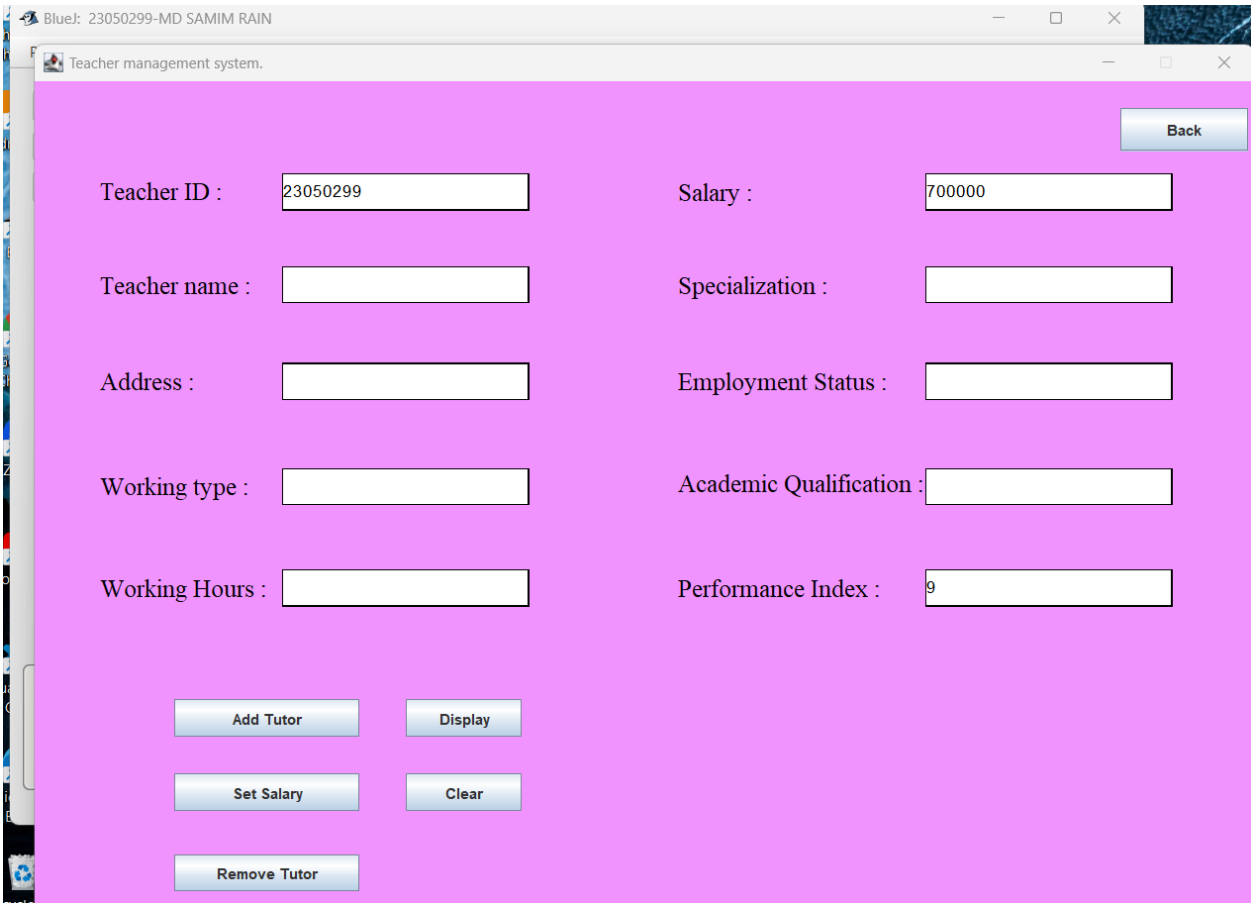


Figure 14: fields of set salary.

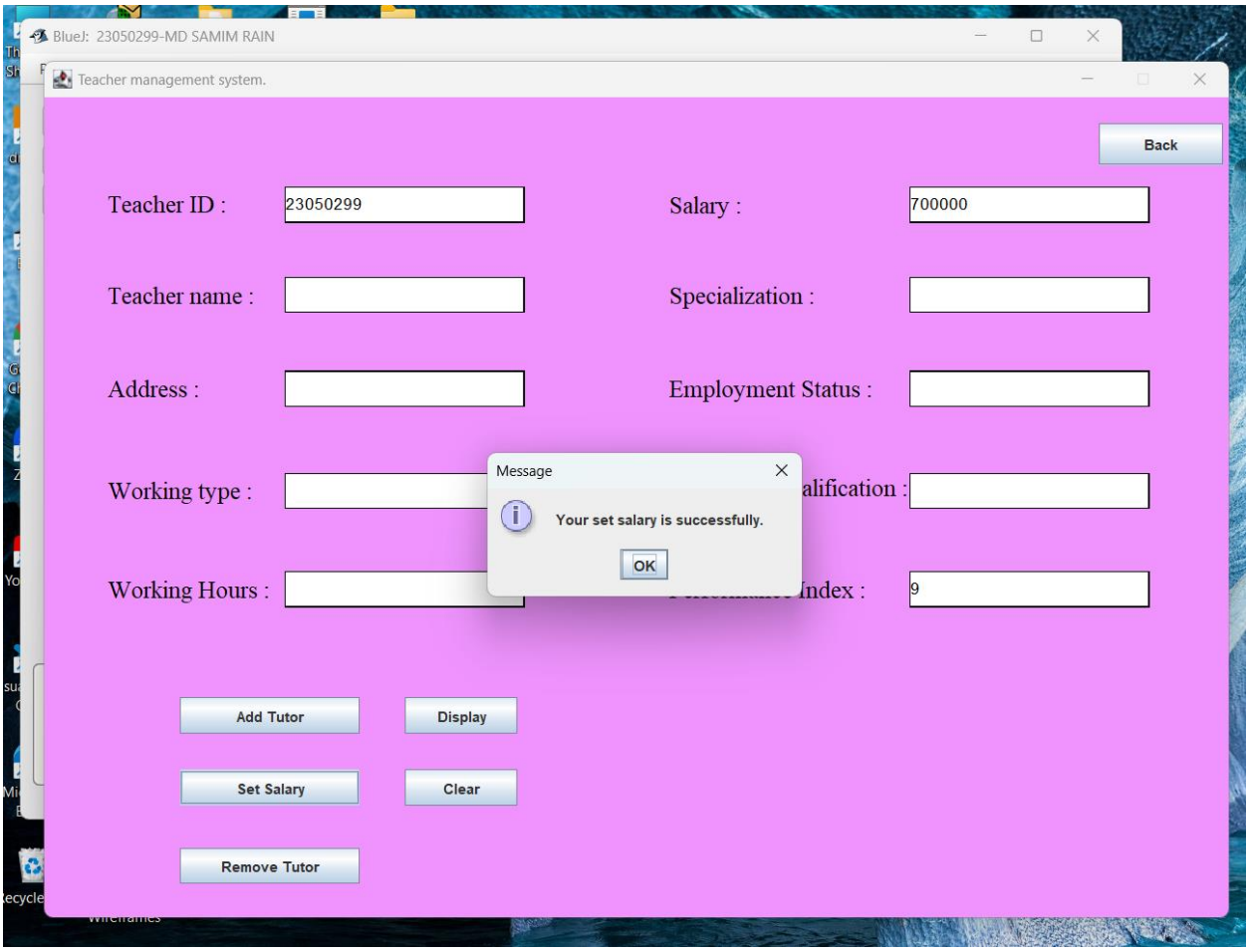


Figure 15: Set salary added.

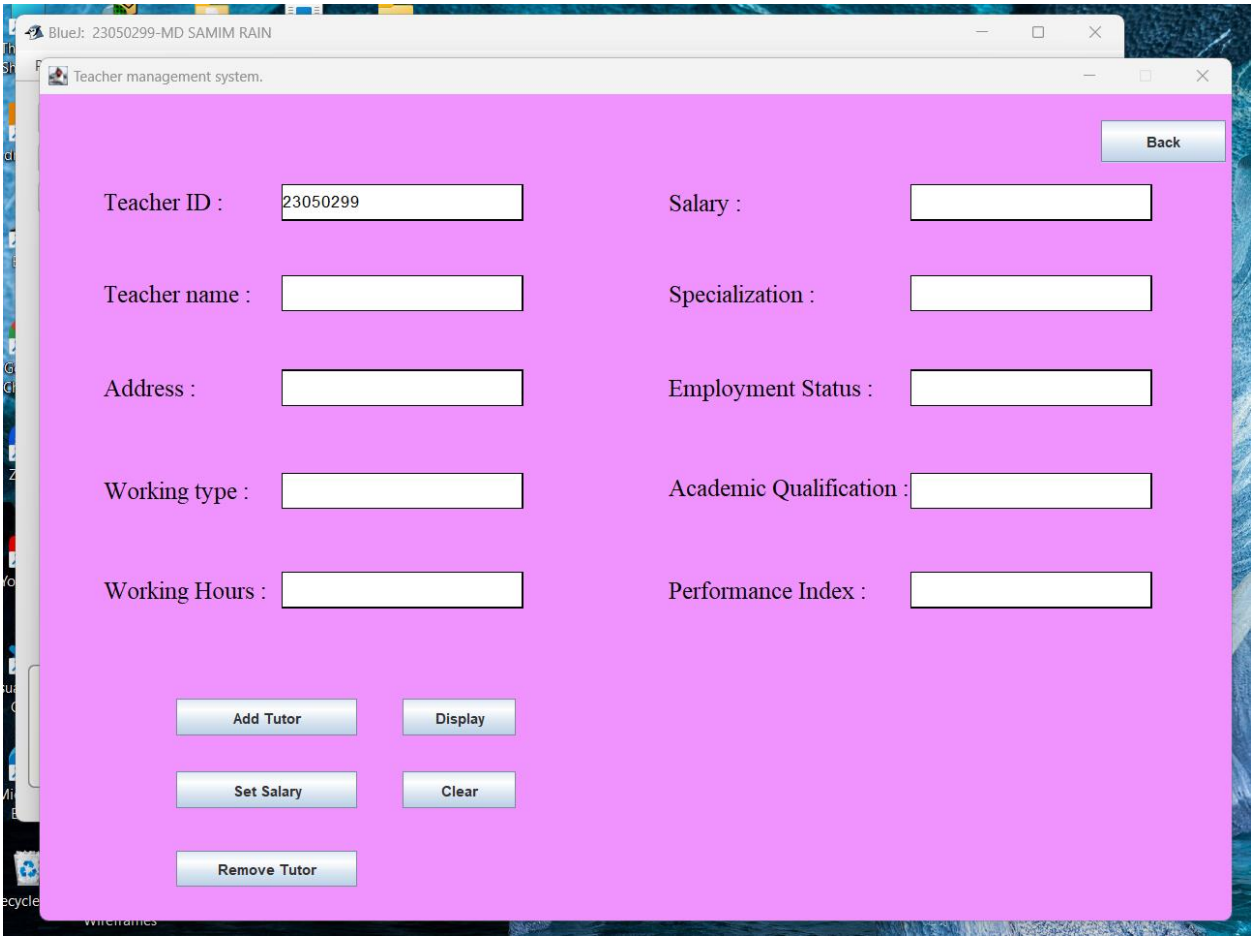


Figure 16: Field of remove tutor.

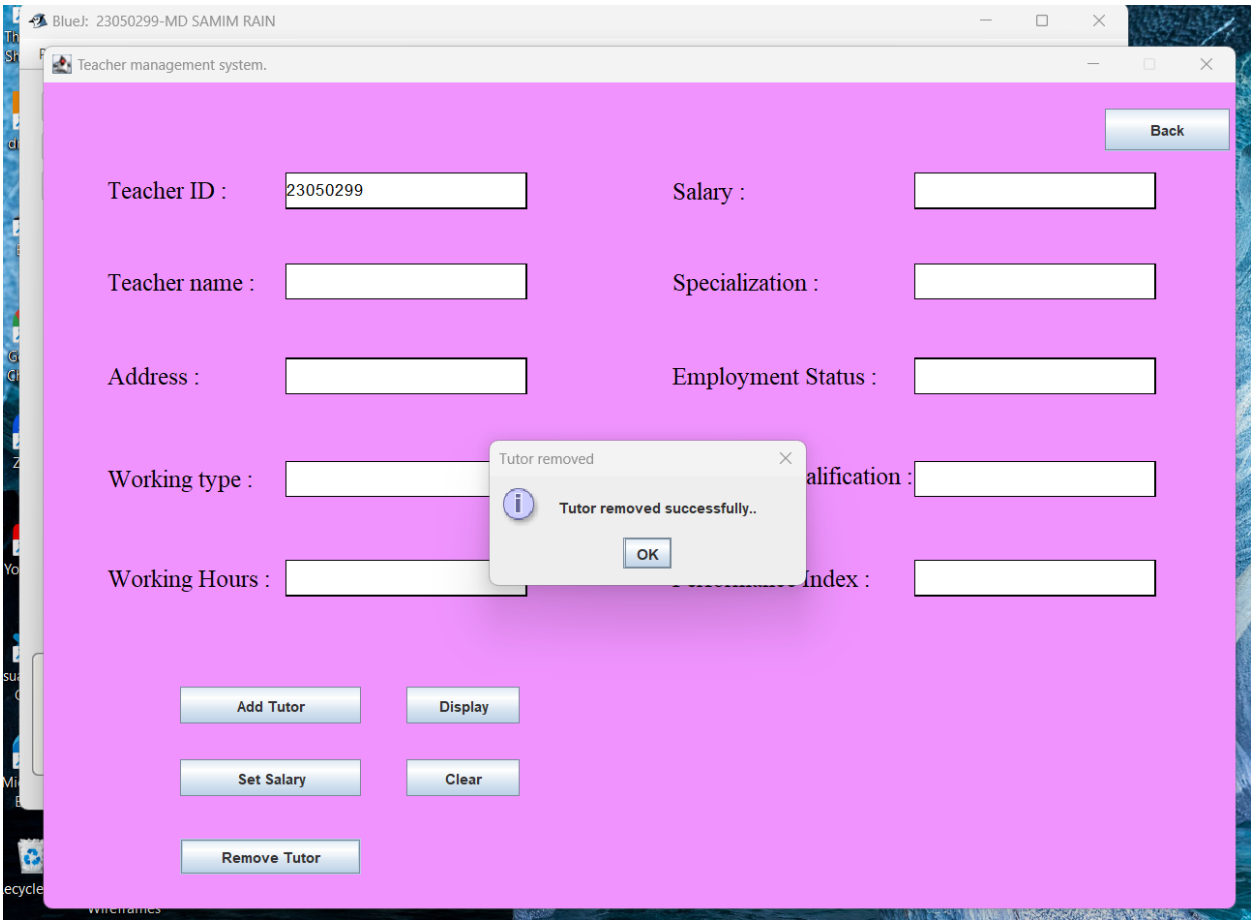


Figure 17: Result of tutor removal.

6.3. Test-3

Objectives	To test that appropriate dialog boxes, appear when unsuitable values are entered for the adding Lecturer.
Action	Step 1: Add Lecturer button was pressed without entering the data. Step 2: Add Lecturer was pressed with the wrong data. Step 3: Press display with the wrong id. Step 4: Press Add Lecturer again after adding it
Expected result	JOPtionPane Dialog to be shown with its relevant problem in the pane.
Actual Result	JOPtionPane Dialog was shown with its relevant problem in the pane.
Conclusion	The test was successful.

Table 3: Table of test 3.

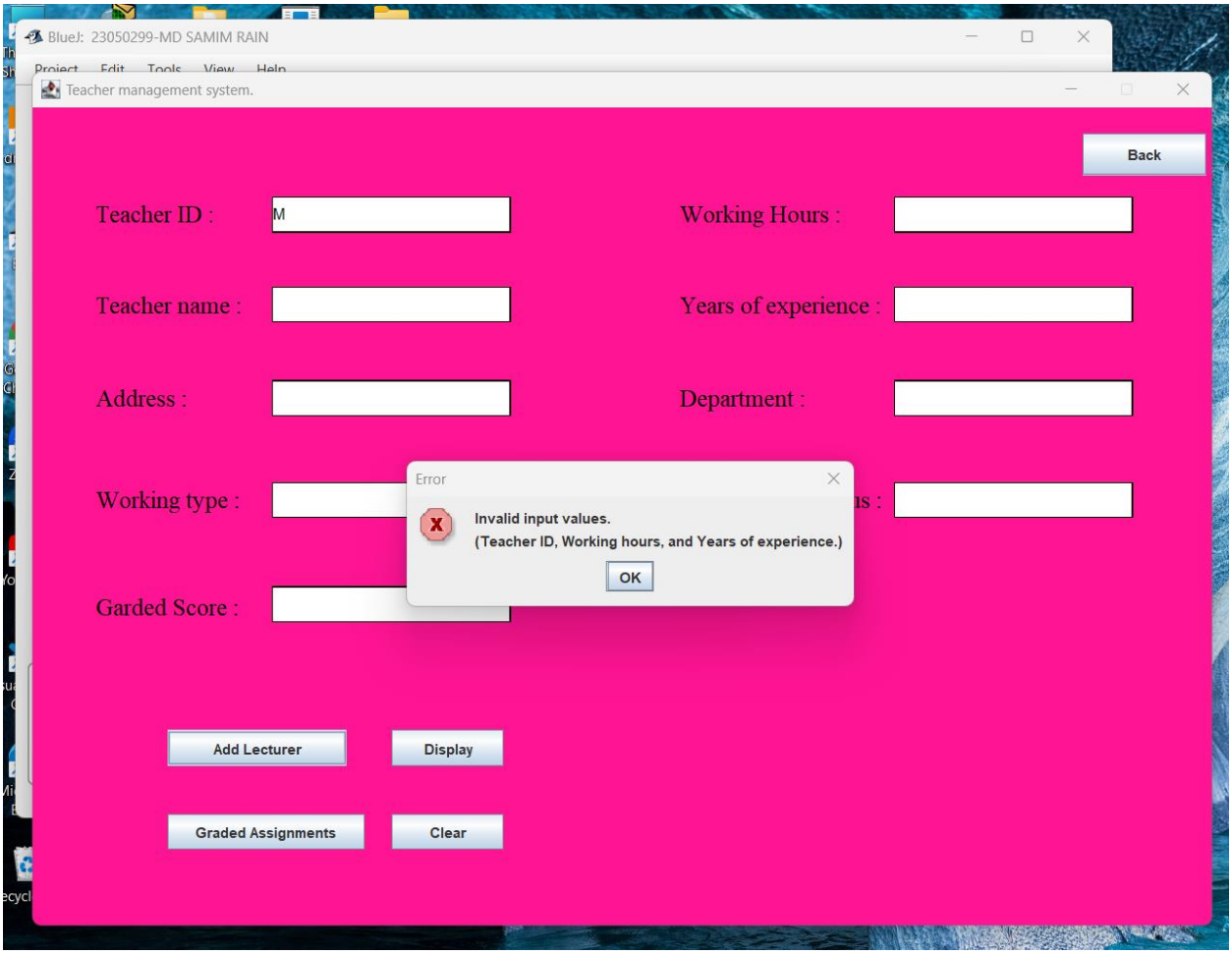


Figure 18: Add a lecturer button click.

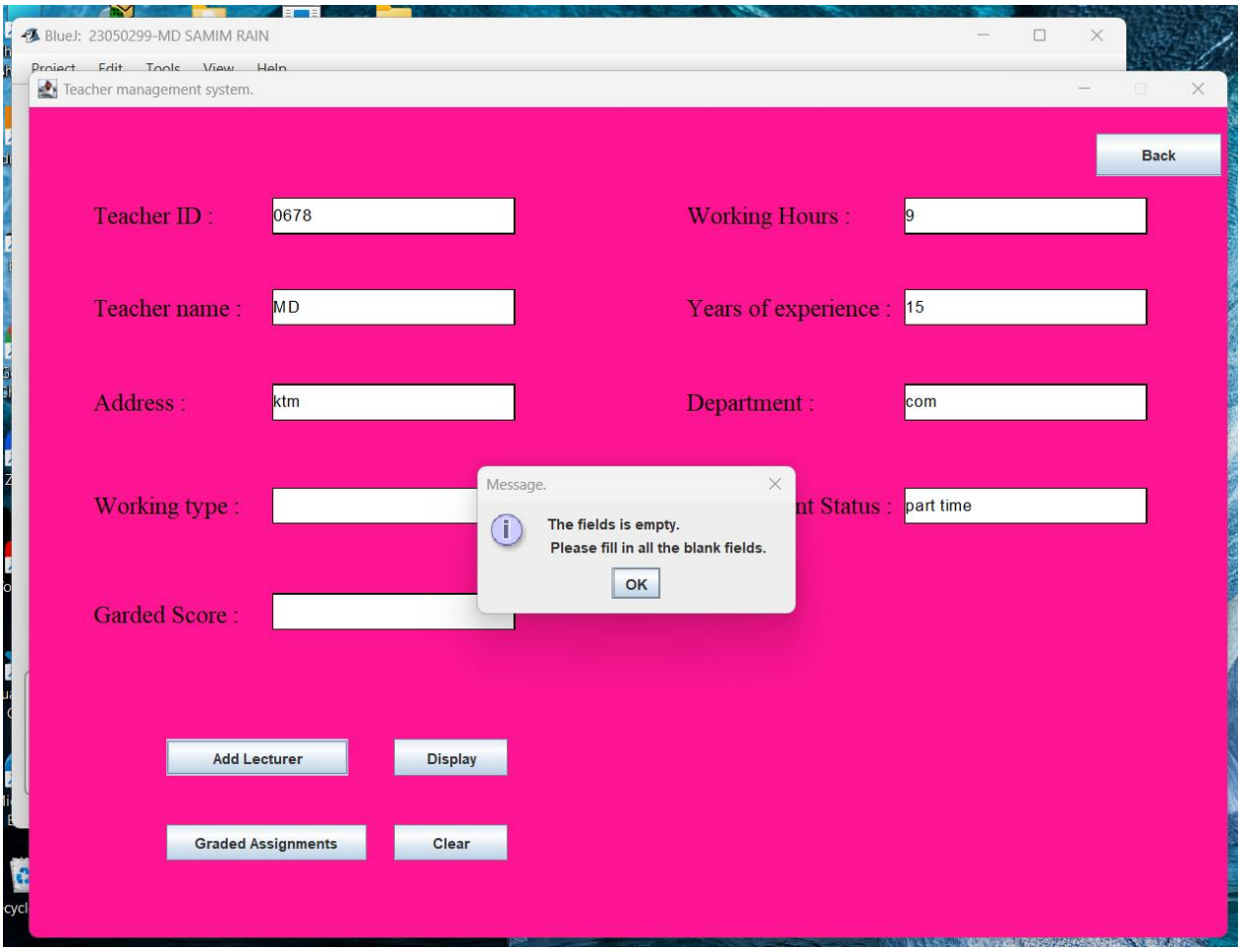


Figure 19: Add lecturer was clicked with wrong fields.

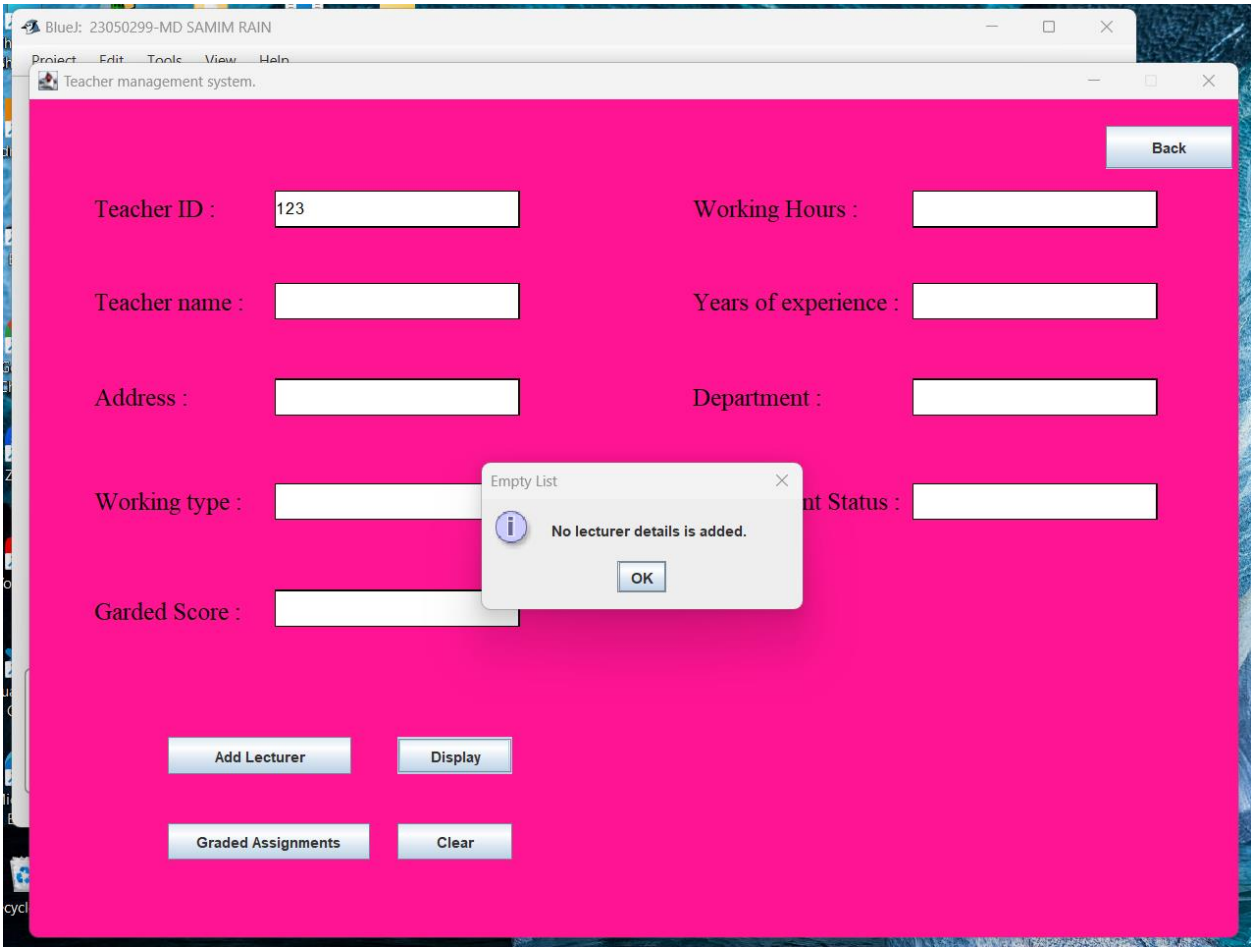


Figure 20: Lecturer details has not been added.

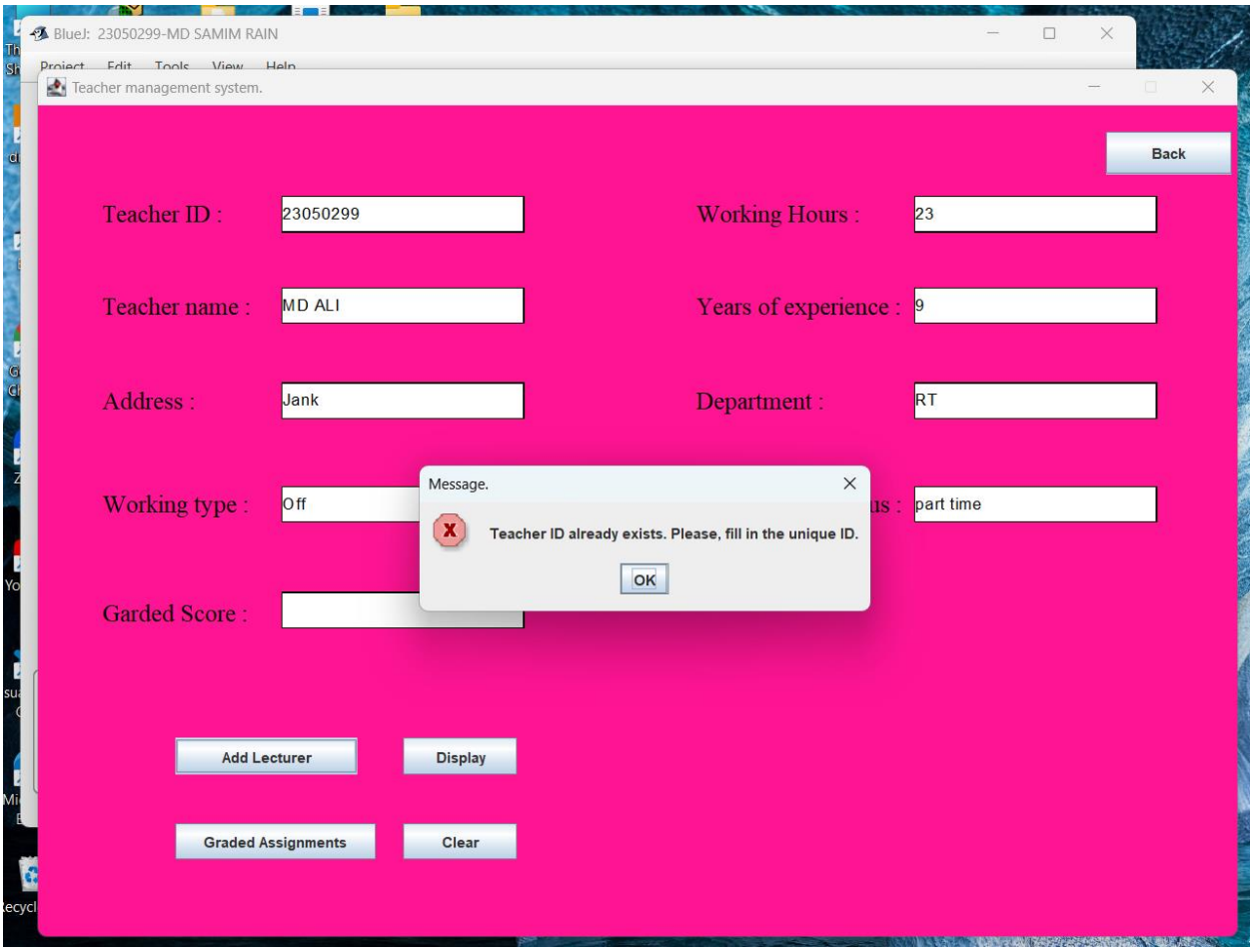


Figure 21: click add lecturer again after adding.

7. Error Detection

7.1 Semantics Error Detection

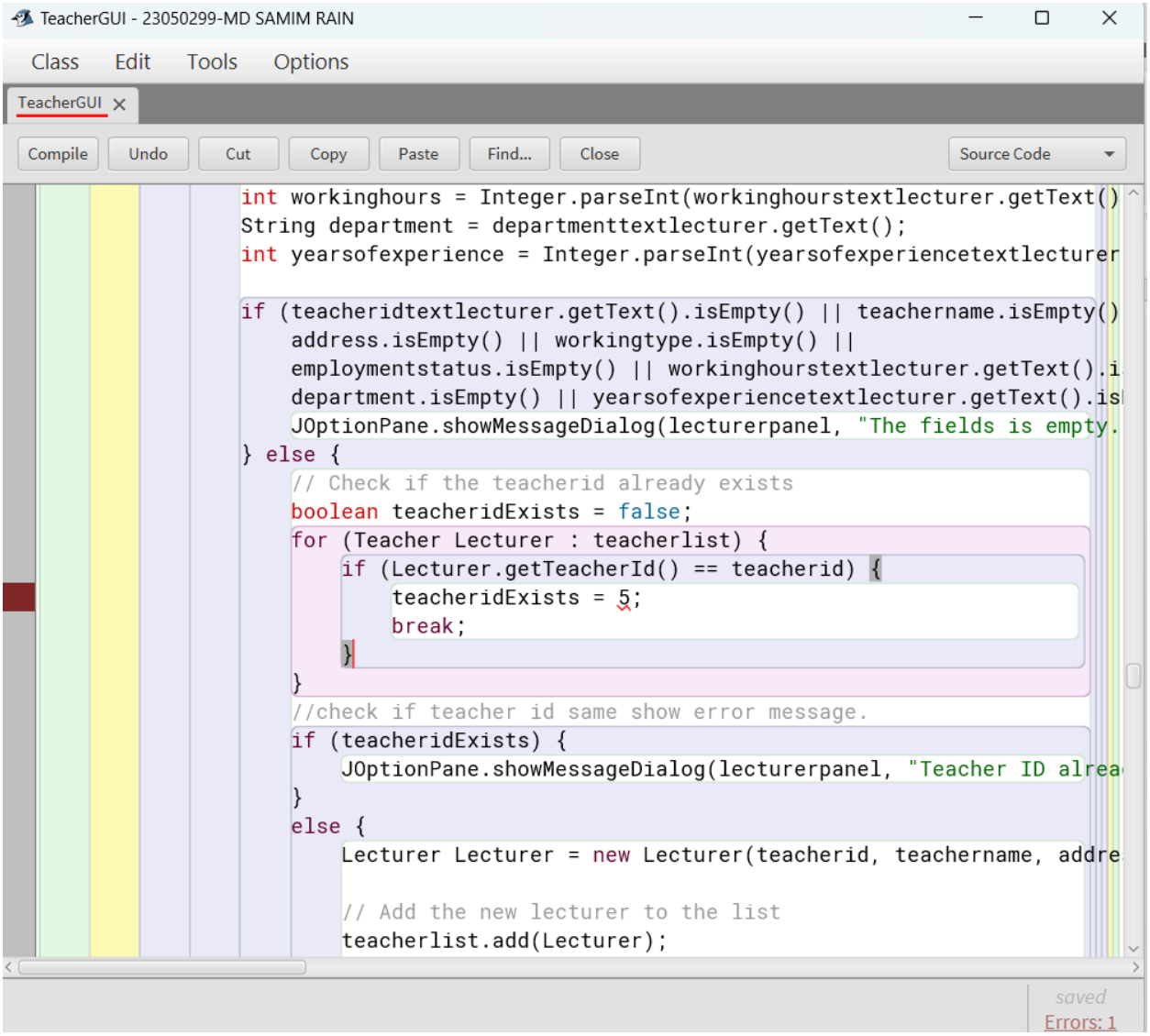


Figure 22: Semantic error.

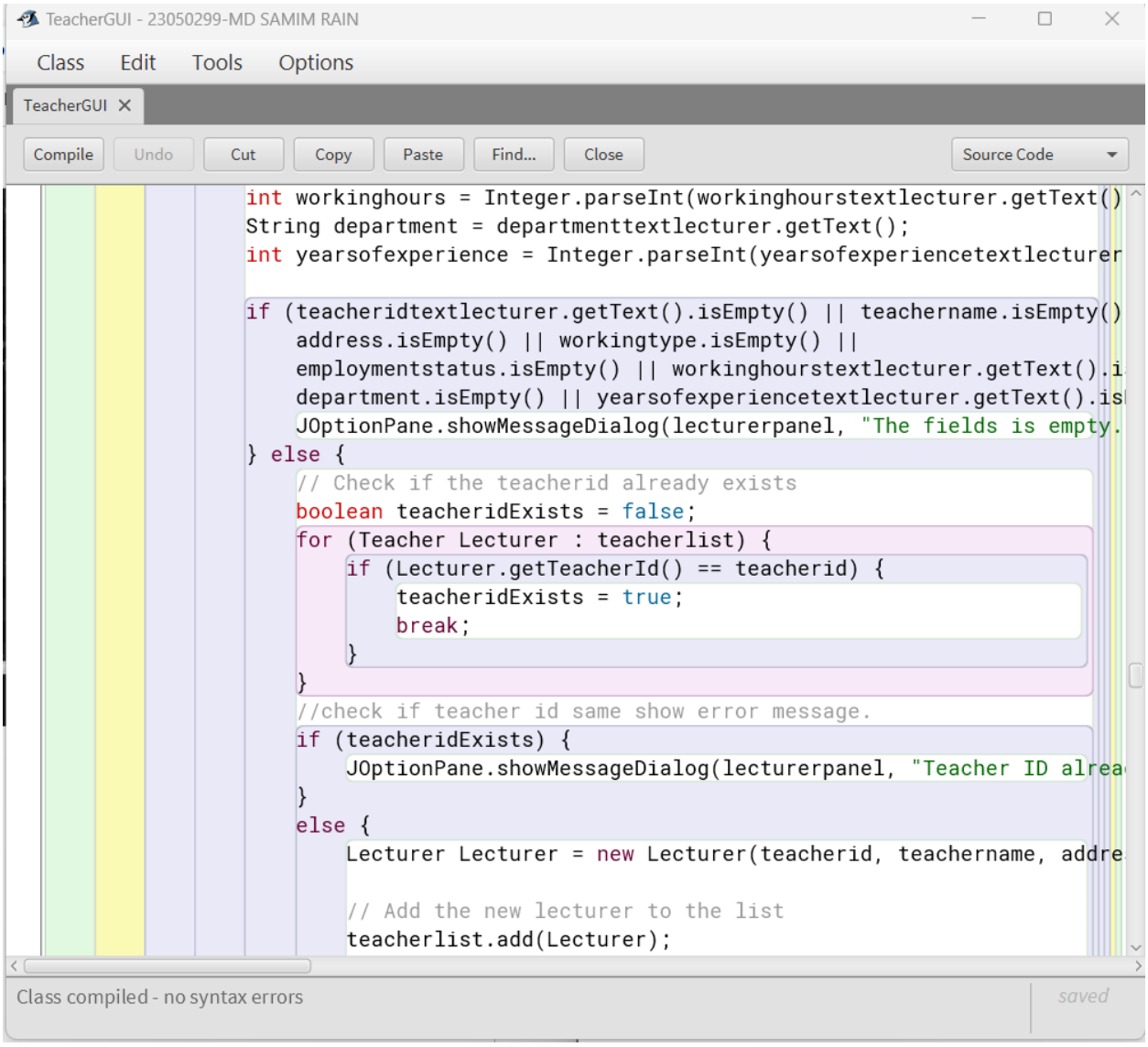


Figure 23: Solved Semantic time error.

7.2 Logical Error

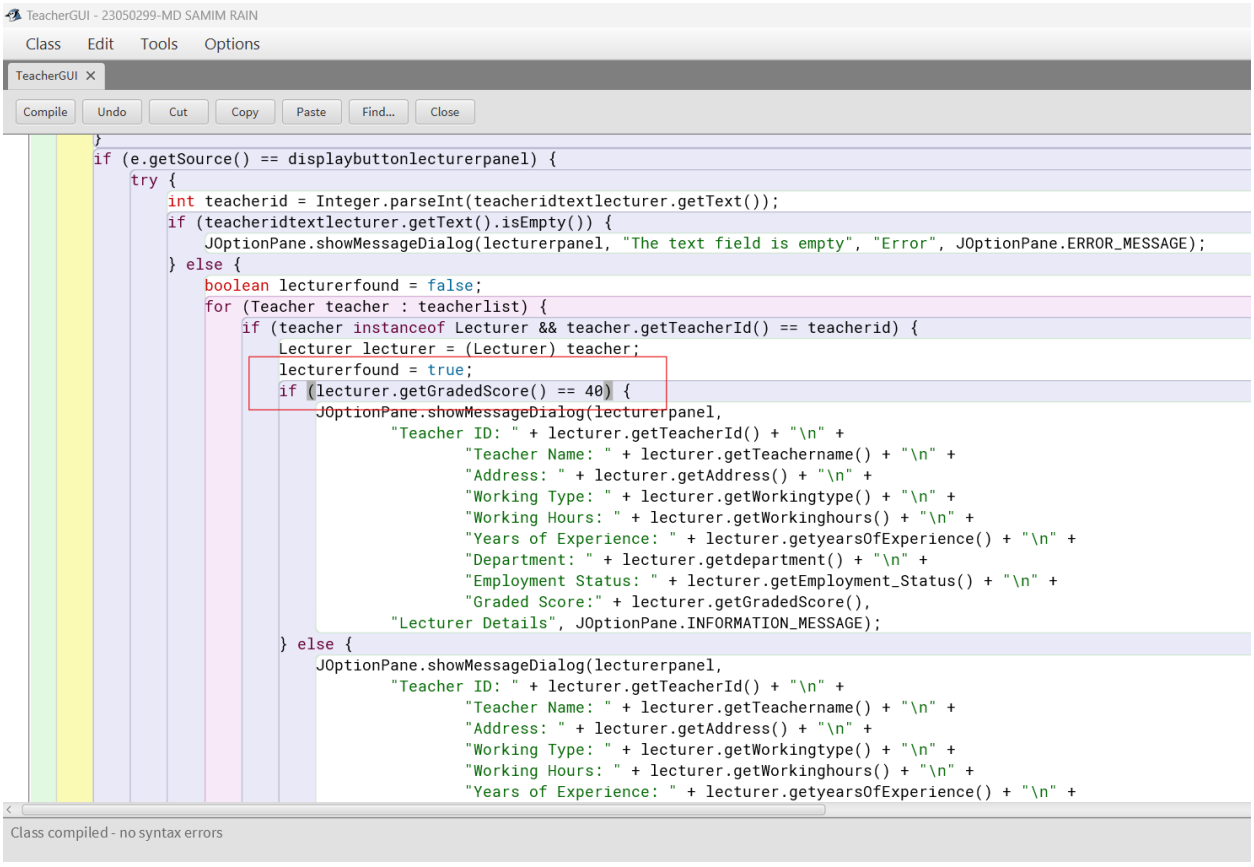


Figure 24: logical error.

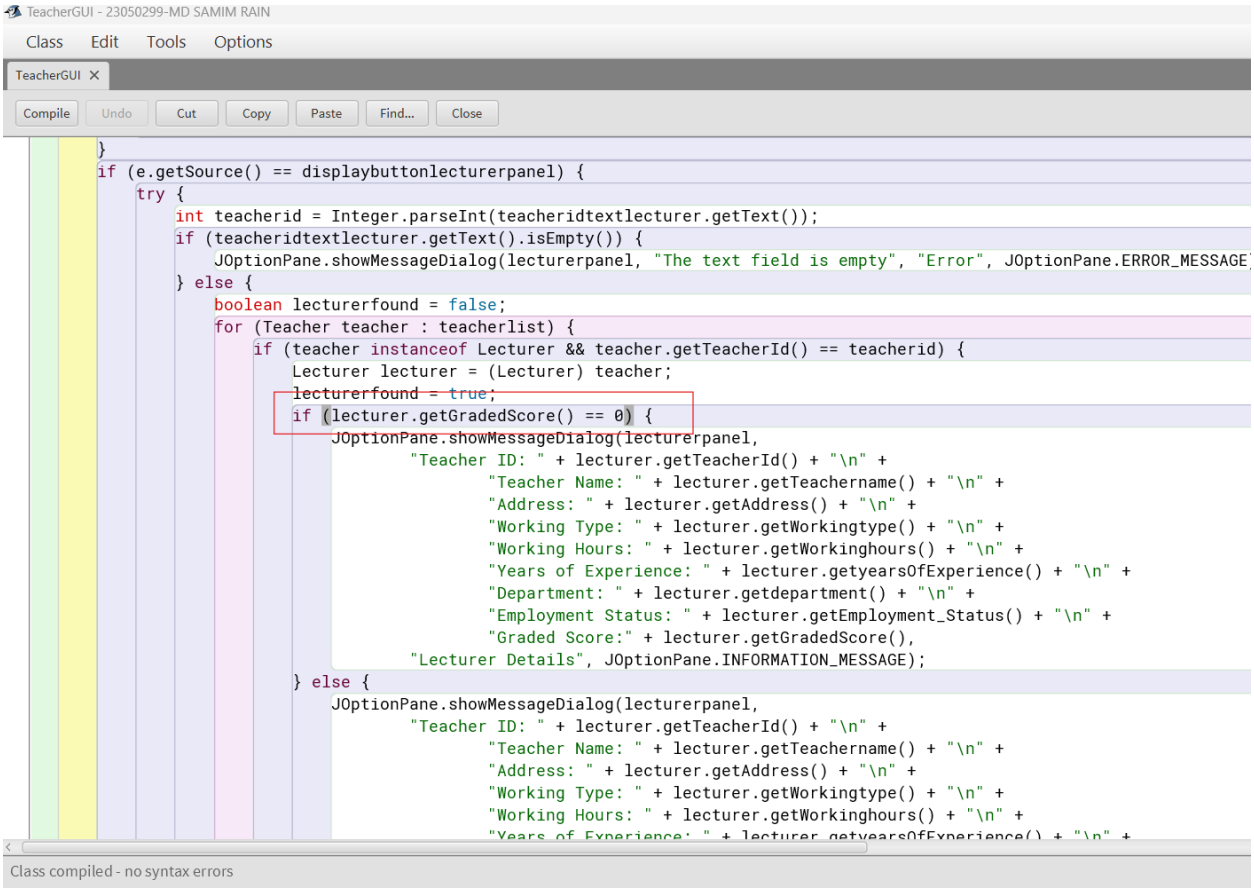


Figure 25: logical error solved.

7.3 Syntax Error

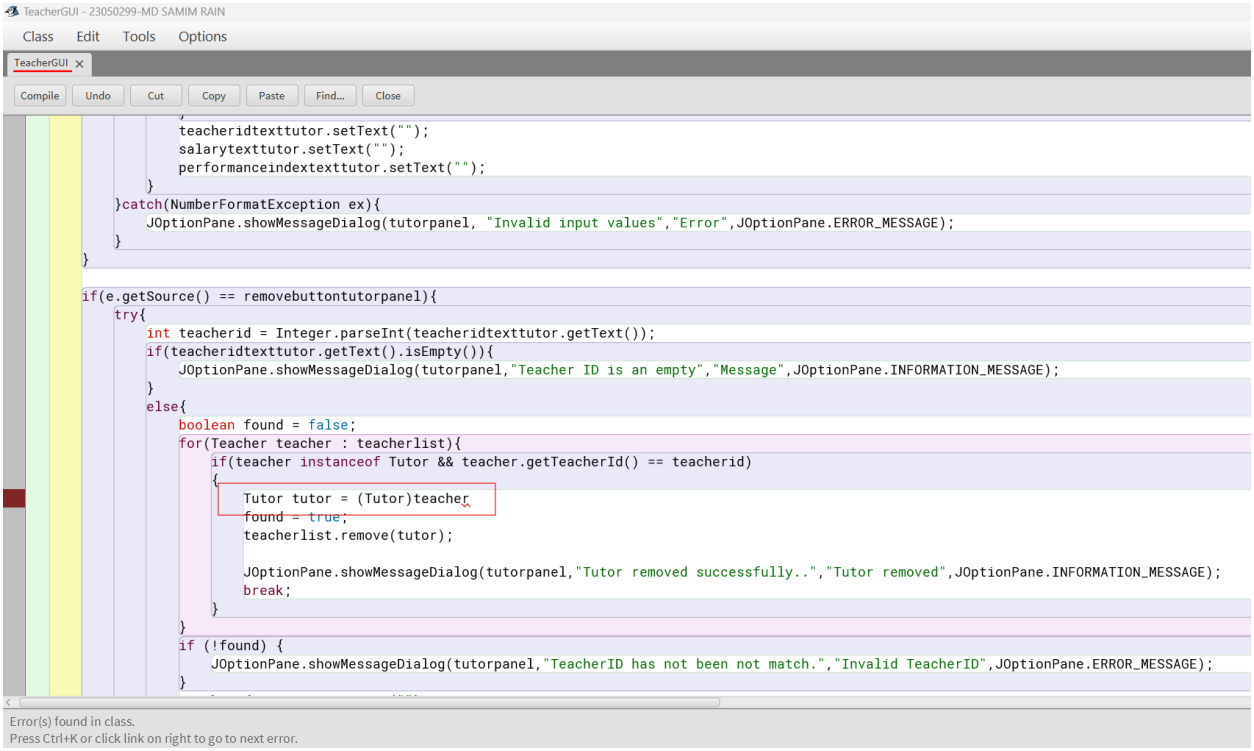


Figure 26: Figure of syntax error.

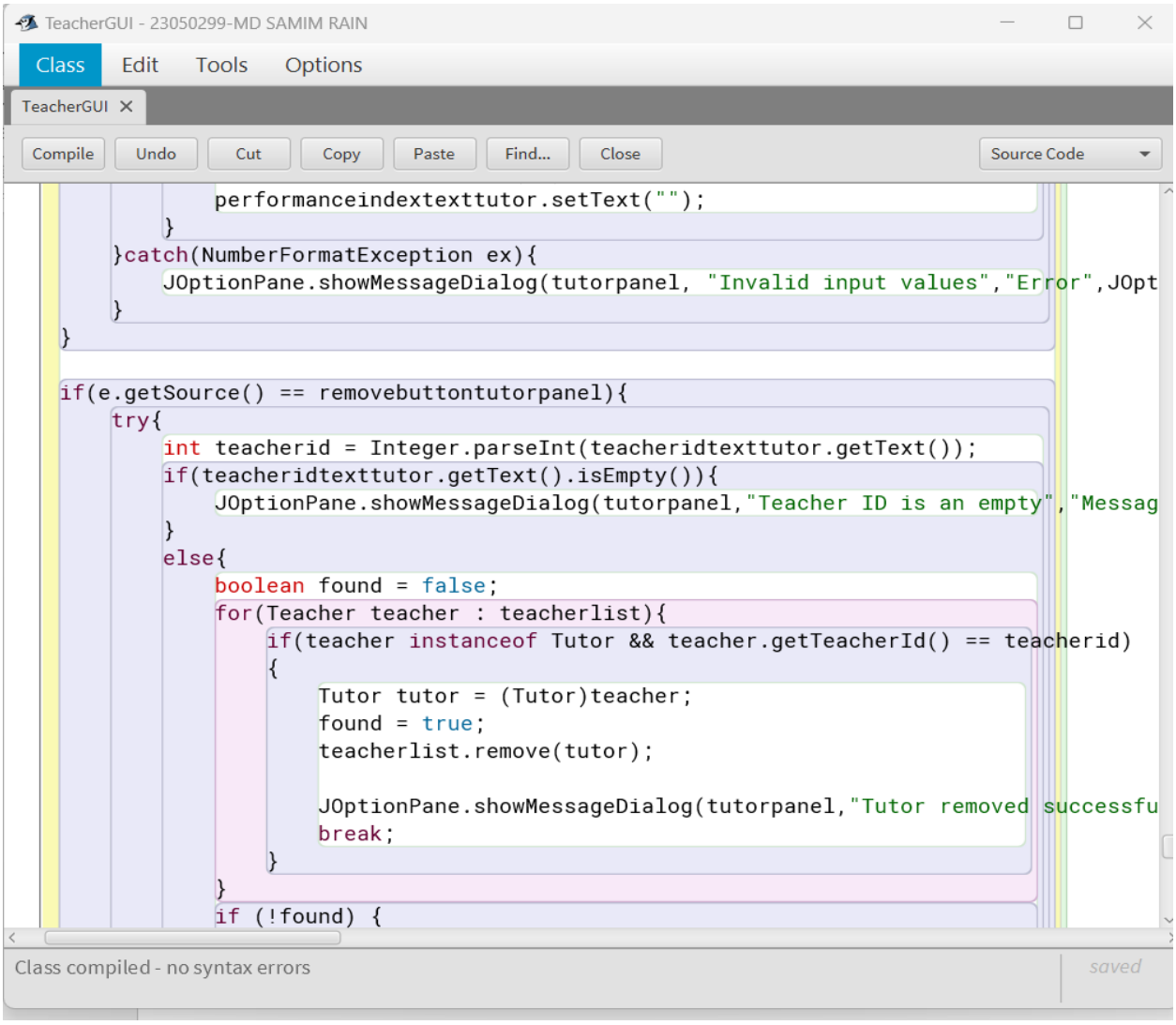


Figure 27: solved of Syntax Error.

8. Conclusion

This assignment involved implementing a Java program using object-oriented concepts, emphasizing inheritance and encapsulation. It deepened my understanding of object-oriented programming and highlighted challenges with grading logic and constructor parameter handling.

Building the Teacher GUI (Teacher Management System) app was like going on a big coding adventure! It was a bit like exploring a maze where you never knew what challenge would pop up next. One of the trickiest parts was making sure all the buttons and stuff showed up correctly on the screen. Sometimes, things would disappear or get jumbled, and I had to dig into the code to fix it.

Getting everything to look just right was another big challenge. It was kind of like solving a super complicated puzzle where every piece had to fit perfectly. Understanding how all the different parts of the interface worked together was tough too. I had to organize everything carefully to make sure it flowed smoothly. Our program has a user-friendly interface, so it's easy for people to use easily.

Overall, our program is a helpful tool for schools and other educational places to keep track of their teachers, making things run more smoothly behind the scenes.

9. Bibliography

(n.d.).

Bluej, I. o. (2022, Aug 29). *geeksforgeeks*. Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/introduction-of-bluej/

Definition, M. W. (n.d.). *javatpoint*. Retrieved from javatpoint: https://www.javatpoint.com/ms-word-definition

Hartman, J. (2023, December 26). *What is Java? Definition, Meaning & Features of Java Platforms*. Retrieved from GURU99: https://www.guru99.com/java-tutorial.html

Hope, C. (2020, 02 06). *Draw.io*. Retrieved from Computer Hope: https://www.computerhope.com/jargon/d/drawio.htm

Tutorial, J. (2024, Apr 26). *Java Tutorial*. Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/java/

10. Appendix

Teacher Java.

```
public class Teacher
{

    private int teacherId;
    private String teachername;
    private String address;
    private String workingtype;
    private String employmentstatus;
    private int workinghours;

    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;
    }

    public int getTeacherId()
    {
        return teacherId;
    }
    public String getTeachername()
    {
        return teachername;
```

```
    }
    public String getAddress()
    {
        return address;
    }
    public String getWorkingtype()
    {
        return workingtype;
    }
    public String getEmployment_Status()
    {
        return employmentstatus;
    }
    public int getWorkinghours()
    {
        return workinghours;
    }

    public void setWorkinghours(int workinghours)
    {
        this.workinghours=workinghours;
    }

    public void display()
    {
        System.out.println("Teacher Details");
        System.out.println("");
        System.out.println("TeacherID: "            + teacherId);
        System.out.println("Teacher Name :"         + teachername);
        System.out.println("Teacher address : "     +address);
```

```
        System.out.println("Teacher working type: "  +workingtype);
        System.out.println("Employment status: "     +employmentstatus);
        if(workinghours==0)
        {
            System.out.println("The teacher working hours :"+workinghours);
        }
        else
        {
            System.out.println("The teacher working hours is not assigned ");
        }
        System.out.println("");
    }
}
```

Lecturer Java.

```
public class Lecturer extends Teacher
{

    private String Department;
    private int YearsOfExperience;
    private int gradedScore;
    private boolean hasGraded;

    public Lecturer(int teacherId,String teachername,String address,String
workingtype,String employmentstatus,int workinghours,String Department, int
YearsOfExperience)
    {
        super(teacherId,teachername,address,workingtype ,employmentstatus);
        setWorkinghours(workinghours);
```

```
    this.Department=Department;
    this.YearsOfExperience=YearsOfExperience;
    this.gradedScore=0;
    this.hasGraded=false;

}

public String getdepartment()
{
    return Department;
}
public int getyearsOfExperience()
{
    return YearsOfExperience;
}
public int getGradedScore()
{
    return gradedScore;
}
public boolean getHasGraded()
{
    return hasGraded;
}

public void setGradedScore(int gradedScore)
{
    this.gradedScore=gradedScore;
}
```

```
    public void gradedAssignment(int gradedScore, String Department, int
YearsOfExperience)
    {

        if( YearsOfExperience >= 5 && this.Department.equals(Department))
        {

            if( gradedScore>=70 )
            {
                System.out.println("The graded assignment score is :A");
            }
            else if(gradedScore>=60)
            {
                System.out.println("The graded assignment score is B");
            }
            else if(gradedScore>=50)
            {
                System.out.println("The graded assignment score is C");
            }
            else if(gradedScore>=40)
            {
               System.out.println("The graded assignment score is D");
            }
            else
            {
                System.out.println("The graded assignment score is E");
            }

            hasGraded=true;
            this.gradedScore=gradedScore;
```

```
            System.out.println("Graded Assignment successfully.");
        }
            else
            {
                System.out.println("The lecturer cannot graded assignment yet");
            }
    }

    @Override
    public void display()
    {

        super.display();



        System.out.println("");
        System.out.println("Lecturer Details");
        System.out.println("Department Name :"              + Department);
        System.out.println("Teacher year of experience :"   + YearsOfExperience);

        if(gradedScore ==0)
        {
            System.out.println("Graded score :"             + gradedScore);
        }
        else
        {
            System.out.println("Assignment cannot graded yet.");
        }
    }
}
```

Tutor Java.

```
public class Tutor extends Teacher
{

    private double salary;
    private String specialization;
    private String academicqualifications;
    private int performanceIndex;
    private boolean isCertified;



    public Tutor(int teacherId,String teachername,String address,String
workingtype,String employmentstatus,int workinghours,
                double salary, String specialization, String academicqualifications, int
performanceIndex)
    {

        super(teacherId,teachername,address,workingtype ,employmentstatus);

        super.setWorkinghours(workinghours);

        this.salary=salary;
        this.specialization=specialization;
        this.academicqualifications=academicqualifications;
        this.performanceIndex=performanceIndex;
        this.isCertified=false;
    }

    public double getSalary()
    {
```

```
        return salary;
    }
    public String getSpecialization()
    {
        return specialization;
    }
    public String getAcademicqualifications()
    {
        return academicqualifications;
    }
    public int getPerformanceIndex()
    {
        return performanceIndex;
    }
    public boolean getIsCertified()
    {
        return isCertified;
    }

    public void setSalary(double salary, int performanceIndex)
    {

        if (performanceIndex > 5 && getWorkinghours() > 20)
        {

            double appraisalPercentage;
            if (performanceIndex >= 5 && performanceIndex <= 7)
            {
                appraisalPercentage = 0.05;
            } else if (performanceIndex >= 8 && performanceIndex <= 9)
```

```
            {
                appraisalPercentage = 0.10;
            } else
            {
                appraisalPercentage = 0.20;
            }

            this.salary = salary + (appraisalPercentage * salary);
            this.isCertified = true;
            System.out.println("Tutuor salary has been approved.");
        }
        else
        {
            System.out.println("Tutor has not been certified yet salary cannot be
approved.");
        }
    }


    public void removeTutor()
    {
        if (!isCertified)
        {

            this.salary = 0;
            this.specialization = "";
            this.academicqualifications = "";
            this.performanceIndex = 0;
            this.isCertified = false;
            System.out.println("Removed tutor successfully.");
```

```
        } else {
            System.out.println("Cannot remove certified tutor.");
        }
    }



    public void display()
    {
        System.out.println("");

        super.display();

        if (isCertified) {

            System.out.println("Tutor Details:");
            System.out.println("Salary: "               + salary);
            System.out.println("Specialization: "       + specialization);
            System.out.println("Academic Qualifications:"+ academicqualifications);
            System.out.println("Performance Index: "    + performanceIndex);
        }
    }
}
```

TeacherGUI.

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JPanel;
import javax.swing.border.Border;
import java.awt.Color;
import java.awt.Font;
import javax.swing.JOptionPane;
import javax.swing.BorderFactory;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.util.ArrayList;



public class TeacherGUI implements ActionListener {



    private JFrame frame;

    private JPanel backgroundpanel,lecturerpanel,tutorpanel;

    private JLabel
title,choosebackgroundpanel,managementlabel,clickbackgroundpanel,imagelabel,tea
cheridlecturerpanellabel,teachernamelecturerpanellabel,addresslecturerpanellabel,wo
rkingtypelecturerpanellabel,workinghourslecturerpanellabel,

yearsofexperiencelecturerpanellabel,departmentlecturerpanellabel,employmentstatusl
```

```
ecturerpanellabel,teacheridtutorpanellabel,teachernametutorpanellabel,addresstutorp
anellabel,workingtypetutorpanellabel,

workinghourstutorpanellabel,salarytutorpanellabel,specializationtutorpanellabel,emplo
ymentstatustutorpanellabel,academicqualificationtutorpanellabel,performanceindextut
orpanellabel,gradedscorelecturerpanellabel;

    private JTextField
teacheridtextlecturer,teachernametextlecturer,addresstextlecturer,workingtypetextlect
urer,workinghourstextlecturer,yearsofexperiencetextlecturer,departmenttextlecturer,

employmentstatustextlecturer,teacheridtexttutor,teachernametexttutor,addresstexttuto
r,workingtypetexttutor,workinghourstexttutor,salarytexttutor,specializationtexttutor,

employmentstatustexttutor,academicqualificationtexttutor,performanceindextexttutor,g
radedscoretextlecturer;

    private JButton
lecturerbuttonbackgroundpanel,tutorbuttonbackgroundpanel,exitbuttonbackgroundpan
el,addbuttonlecturerpanel,gradedassignmentsbuttonlecturerpanel,displaybuttonlecture
rpanel,

clearbuttonlecturerpanel,backbuttonlecturerpanel,backbuttontutorpanel,addtutorbutton
panel,setsalarybuttontutorpanel,removebuttontutorpanel,displaybuttontutorpanel,clear
buttontutorpanel;

    private ArrayList<Teacher>  teacherlist;


    public TeacherGUI(){
```

```
        teacherlist = new ArrayList<>();


        frame = new JFrame();

        title = new JLabel();



        backgroundpanel = new JPanel();



        managementlabel = new JLabel("Teacher management system.");
        managementlabel.setBackground(new Color(121,52,32));
        managementlabel.setBounds(325,38,400,35);
        managementlabel.setFont(new Font("Times New Roman",Font.ITALIC,30));
        backgroundpanel.add(managementlabel);

        choosebackgroundpanel = new JLabel("Choose an Option");
        choosebackgroundpanel.setBounds(282,114,344,38);
        choosebackgroundpanel.setFont(new Font("Times New
Roman",Font.BOLD,33));
        backgroundpanel.add(choosebackgroundpanel);



        clickbackgroundpanel = new JLabel("Click here!");
        clickbackgroundpanel.setBounds(535,115,173,38);
        clickbackgroundpanel.setFont(new Font("Times New Roman",Font.BOLD,30));
```

```
        backgroundpanel.add(clickbackgroundpanel);



        lecturerbuttonbackgroundpanel = new JButton("Lecturer");
        lecturerbuttonbackgroundpanel.setBounds(326,169,147,46);
        lecturerbuttonbackgroundpanel.setBackground(new Color(231,120,40));
        lecturerbuttonbackgroundpanel.setFont(new Font("Times New
Roman",Font.CENTER_BASELINE,20));
        Border borderlbg = BorderFactory.createLineBorder(new Color(253,95,85));
        lecturerbuttonbackgroundpanel.setBorder(borderlbg);
        lecturerbuttonbackgroundpanel.addActionListener(this);
        backgroundpanel.add(lecturerbuttonbackgroundpanel);



        tutorbuttonbackgroundpanel = new JButton("Tutor");
        tutorbuttonbackgroundpanel.setBounds(327,235,120,43);
        tutorbuttonbackgroundpanel.setBackground(new Color(90,227,27));
        tutorbuttonbackgroundpanel.setFont(new Font("Times New
Roman",Font.CENTER_BASELINE,20));
        Border bordertbg = BorderFactory.createLineBorder(new Color(101,203,197));
        tutorbuttonbackgroundpanel.setBorder(bordertbg);
        tutorbuttonbackgroundpanel.addActionListener(this);
        backgroundpanel.add(tutorbuttonbackgroundpanel);



        exitbuttonbackgroundpanel = new JButton("Exit");
        exitbuttonbackgroundpanel.setBounds(328,296,83,42);
        exitbuttonbackgroundpanel.setBackground(new Color(194,0,235));
```

```
        exitbuttonbackgroundpanel.setFont(new Font("Times New
Roman",Font.CENTER_BASELINE,20));
        Border borderetb = BorderFactory.createEmptyBorder(1, 1, 01, 01);
        exitbuttonbackgroundpanel.setBorder(borderetb);
        exitbuttonbackgroundpanel.addActionListener(this);
        backgroundpanel.add(exitbuttonbackgroundpanel);



        ImageIcon icon = new ImageIcon("C:\\Users\\rains\\OneDrive\\Desktop\\MD
SAMIM\\classroom.png");



        imagelabel = new JLabel();
        imagelabel.setBounds(248, 92, icon.getIconWidth(), icon.getIconHeight());
        imagelabel.setIcon(icon);
        backgroundpanel.add(imagelabel);

        backgroundpanel.setBackground(new Color(56,179,227));
        backgroundpanel.setBounds(0,0,1000,720);
        backgroundpanel.setLayout(null);
        frame.add(backgroundpanel);



        lecturerpanel = new JPanel();



        teacheridlecturerpanellabel = new JLabel("Teacher ID :");
```

```
        teacheridlecturerpanellabel.setBounds(53,75,120,30);
        teacheridlecturerpanellabel.setForeground(Color.BLACK);
        teacheridlecturerpanellabel.setFont(new Font("Times New
Roman",Font.PLAIN,20));
        lecturerpanel.add(teacheridlecturerpanellabel);

        teacheridtextlecturer = new JTextField();
        teacheridtextlecturer.setBounds(200,75,200,30);
        teacheridtextlecturer.setFont(new Font("Arial",Font.ROMAN_BASELINE,14));
        teacheridtextlecturer.setForeground(Color.black);
        Border border = BorderFactory.createLineBorder(Color.black,1);
        teacheridtextlecturer.setBorder(border);
        lecturerpanel.add(teacheridtextlecturer);



        teachernamelecturerpanellabel = new JLabel("Teacher name :");
        teachernamelecturerpanellabel.setBounds(53,150,150,33);
        teachernamelecturerpanellabel.setForeground((Color.BLACK));
        teachernamelecturerpanellabel.setFont(new Font("Times New
Roman",Font.PLAIN,20));
        lecturerpanel.add(teachernamelecturerpanellabel);

        teachernametextlecturer = new JTextField();
        teachernametextlecturer.setBounds(200,150,200,30);
        teachernametextlecturer.setFont(new
Font("Arial",Font.ROMAN_BASELINE,14));
        teachernametextlecturer.setForeground(Color.BLACK);
        Border bordertnl = BorderFactory.createLineBorder(Color.black,1);
        teachernametextlecturer.setBorder(bordertnl);
        lecturerpanel.add(teachernametextlecturer);
```

```
        addresslecturerpanellabel = new JLabel("Address :");
        addresslecturerpanellabel.setBounds(53,228,105,33);
        addresslecturerpanellabel.setForeground(Color.BLACK);
        addresslecturerpanellabel.setFont(new Font("Times New
Roman",Font.PLAIN,20));
        lecturerpanel.add(addresslecturerpanellabel);

        addresstextlecturer = new JTextField();
        addresstextlecturer.setBounds(200,228,200,30);
        addresstextlecturer.setForeground(Color.BLACK);
        addresstextlecturer.setFont(new Font("Arial",Font.ROMAN_BASELINE,14));
        Border borderal = BorderFactory.createLineBorder(Color.black,1);
        addresstextlecturer.setBorder(borderal);
        lecturerpanel.add(addresstextlecturer);

        workingtypelecturerpanellabel = new JLabel("Working type :");
        workingtypelecturerpanellabel.setBounds(53,313,150,33);
        workingtypelecturerpanellabel.setForeground(Color.BLACK);
        workingtypelecturerpanellabel.setFont(new Font("Times New
Roman",Font.PLAIN,20));
        lecturerpanel.add(workingtypelecturerpanellabel);

        workingtypetextlecturer = new JTextField();
        workingtypetextlecturer.setBounds(200,313,200,30);
        workingtypetextlecturer.setForeground(Color.BLACK);
        workingtypetextlecturer.setFont(new Font("Arial",Font.ROMAN_BASELINE,14));
        Border borderwtl = BorderFactory.createLineBorder(Color.black,1);
        workingtypetextlecturer.setBorder(borderwtl);
        lecturerpanel.add(workingtypetextlecturer);
```

```
        gradedscorelecturerpanellabel = new JLabel("Garded Score :");
        gradedscorelecturerpanellabel.setBounds(53,400,150,36);
        gradedscorelecturerpanellabel.setForeground(Color.BLACK);
        gradedscorelecturerpanellabel.setFont(new Font("Times New
Roman",Font.PLAIN,20));
        lecturerpanel.add(gradedscorelecturerpanellabel);

        gradedscoretextlecturer = new JTextField();
        gradedscoretextlecturer.setBounds(200,400,200,30);
        gradedscoretextlecturer.setForeground(Color.BLACK);
        gradedscoretextlecturer.setFont(new Font("Arial",Font.ROMAN_BASELINE,14));
        Border bordergsl = BorderFactory.createLineBorder(Color.black,1);
        gradedscoretextlecturer.setBorder(bordergsl);
        lecturerpanel.add(gradedscoretextlecturer);

        workinghourslecturerpanellabel = new JLabel("Working Hours :");
        workinghourslecturerpanellabel.setBounds(541,75,150,30);
        workinghourslecturerpanellabel.setForeground(Color.BLACK);
        workinghourslecturerpanellabel.setFont(new Font("Times New
Roman",Font.PLAIN,20));
        lecturerpanel.add(workinghourslecturerpanellabel);

        workinghourstextlecturer = new JTextField();
        workinghourstextlecturer.setBounds(720,75,200,30);
        workinghourstextlecturer.setForeground(Color.BLACK);
        workinghourstextlecturer.setFont(new
Font("Arial",Font.ROMAN_BASELINE,14));
        Border borderwhl = BorderFactory.createLineBorder(Color.black,1);
        workinghourstextlecturer.setBorder(borderwhl);
```

```
        lecturerpanel.add(workinghourstextlecturer);

        yearsofexperiencelecturerpanellabel = new JLabel("Years of experience :");
        yearsofexperiencelecturerpanellabel.setBounds(541,150,200,33);
        yearsofexperiencelecturerpanellabel.setForeground(Color.BLACK);
        yearsofexperiencelecturerpanellabel.setFont(new Font("Times New
Roman",Font.PLAIN,20));
        lecturerpanel.add(yearsofexperiencelecturerpanellabel);

        yearsofexperiencetextlecturer=new JTextField();
        yearsofexperiencetextlecturer.setBounds(720,150,200,30);
        yearsofexperiencetextlecturer.setForeground(Color.BLACK);
        yearsofexperiencetextlecturer.setFont(new
Font("Arial",Font.ROMAN_BASELINE,14));
        Border borderyoel = BorderFactory.createLineBorder(Color.black,1);
        yearsofexperiencetextlecturer.setBorder(borderyoel);
        lecturerpanel.add(yearsofexperiencetextlecturer);

        departmentlecturerpanellabel = new JLabel("Department :");
        departmentlecturerpanellabel.setBounds(541,228,143,33);
        departmentlecturerpanellabel.setForeground(Color.BLACK);
        departmentlecturerpanellabel.setFont(new Font("Times New
Roman",Font.PLAIN,20));
        lecturerpanel.add(departmentlecturerpanellabel);

        departmenttextlecturer =new JTextField();
        departmenttextlecturer.setBounds(720,228,200,30);
        departmenttextlecturer.setForeground(Color.BLACK);
        departmenttextlecturer.setFont(new Font("Arial",Font.ROMAN_BASELINE,14));
        Border borderdl = BorderFactory.createLineBorder(Color.black,1);
```

```
        departmenttextlecturer.setBorder(borderdl);
        lecturerpanel.add(departmenttextlecturer);

        employmentstatuslecturerpanellabel = new JLabel("Employment Status :");
        employmentstatuslecturerpanellabel.setBounds(541,313,200,33);
        employmentstatuslecturerpanellabel.setForeground(Color.BLACK);
        employmentstatuslecturerpanellabel.setFont(new Font("Times New
Roman",Font.PLAIN,20));
        lecturerpanel.add(employmentstatuslecturerpanellabel);

        employmentstatustextlecturer = new JTextField();
        employmentstatustextlecturer.setBounds(720,313,200,30);
        employmentstatustextlecturer.setFont(new
Font("Arial",Font.ROMAN_BASELINE,14));
        employmentstatustextlecturer.setForeground(Color.BLACK);
        Border borderesl = BorderFactory.createLineBorder(Color.black,1);
        employmentstatustextlecturer.setBorder(borderesl);
        lecturerpanel.add(employmentstatustextlecturer);



        addbuttonlecturerpanel = new JButton("Add Lecturer");
        addbuttonlecturerpanel.setBounds(113,520,150,30);
        addbuttonlecturerpanel.addActionListener(this);
        lecturerpanel.add(addbuttonlecturerpanel);

        gradedassignmentsbuttonlecturerpanel = new JButton("Graded Assignments");
        gradedassignmentsbuttonlecturerpanel.setBounds(113,590,165,30);
        gradedassignmentsbuttonlecturerpanel.addActionListener(this);
        lecturerpanel.add(gradedassignmentsbuttonlecturerpanel);
```

```
displaybuttonlecturerpanel = new JButton("Display");
displaybuttonlecturerpanel.setBounds(300,520,94,30);
displaybuttonlecturerpanel.addActionListener(this);
lecturerpanel.add(displaybuttonlecturerpanel);

clearbuttonlecturerpanel = new JButton("Clear");
clearbuttonlecturerpanel.setBounds(300,590,94,30);
clearbuttonlecturerpanel.addActionListener(this);
lecturerpanel.add(clearbuttonlecturerpanel);

backbuttonlecturerpanel = new JButton("Back");
backbuttonlecturerpanel.setBounds(878,22,103,35);
backbuttonlecturerpanel.addActionListener(this);
lecturerpanel.add(backbuttonlecturerpanel);


lecturerpanel.setBackground(new Color(255,20,147));
lecturerpanel.setBounds(0,0,1000,720);
lecturerpanel.setLayout(null);



tutorpanel = new JPanel();



teacheridtutorpanellabel = new JLabel("Teacher ID :");
teacheridtutorpanellabel.setBounds(53,75,120,30);
teacheridtutorpanellabel.setForeground(Color.BLACK);
```

```
        teacheridtutorpanellabel.setFont(new Font("Times New
Roman",Font.PLAIN,20));
        tutorpanel.add(teacheridtutorpanellabel);

        teacheridtexttutor = new JTextField();
        teacheridtexttutor.setBounds(200,75,200,30);
        teacheridtexttutor.setFont(new Font("Arial",Font.ROMAN_BASELINE,14));
        teacheridtexttutor.setForeground(Color.BLACK);
        Border bordertidt = BorderFactory.createLineBorder(Color.black,1);
        teacheridtexttutor.setBorder(bordertidt);
        tutorpanel.add(teacheridtexttutor);

        teachernametutorpanellabel = new JLabel("Teacher name :");
        teachernametutorpanellabel.setBounds(53,150,150,33);
        teachernametutorpanellabel.setForeground(Color.BLACK);
        teachernametutorpanellabel.setFont(new Font("Times New
Roman",Font.PLAIN,20));
        tutorpanel.add(teachernametutorpanellabel);

        teachernametexttutor = new JTextField();
        teachernametexttutor.setBounds(200,150,200,30);
        teachernametexttutor.setForeground(Color.BLACK);
        teachernametexttutor.setFont(new Font("Arial",Font.ROMAN_BASELINE,14));
        Border bordertnt = BorderFactory.createLineBorder(Color.black,1);
        teachernametexttutor.setBorder(bordertnt);
        tutorpanel.add(teachernametexttutor);

        addresstutorpanellabel = new JLabel("Address :");
        addresstutorpanellabel.setBounds(53,228,105,33);
        addresstutorpanellabel.setForeground(Color.BLACK);
```

```
        addresstutorpanellabel.setFont(new Font("Times New Roman",Font.PLAIN,20));
        tutorpanel.add(addresstutorpanellabel);

        addresstexttutor = new JTextField();
        addresstexttutor.setBounds(200,228,200,30);
        addresstexttutor.setFont(new Font("Arial",Font.ROMAN_BASELINE,14));
        addresstexttutor.setForeground(Color.BLACK);
        Border borderat = BorderFactory.createLineBorder(Color.black,1);
        addresstexttutor.setBorder(borderat);
        tutorpanel.add(addresstexttutor);

        workingtypetutorpanellabel = new JLabel("Working type :");
        workingtypetutorpanellabel.setBounds(53,313,150,33);
        workingtypetutorpanellabel.setForeground(Color.BLACK);
        workingtypetutorpanellabel.setFont(new Font("Times New
Roman",Font.PLAIN,20));
        tutorpanel.add(workingtypetutorpanellabel);

        workingtypetexttutor = new JTextField();
        workingtypetexttutor.setBounds(200,313,200,30);
        workingtypetexttutor.setFont(new Font("Arial",Font.ROMAN_BASELINE,14));
        workingtypetexttutor.setForeground(Color.BLACK);
        Border borderwtt = BorderFactory.createLineBorder(Color.black,1);
        workingtypetexttutor.setBorder(borderwtt);
        tutorpanel.add(workingtypetexttutor);

        workinghourstutorpanellabel = new JLabel("Working Hours :");
        workinghourstutorpanellabel.setBounds(53,395,150,33);
        workinghourstutorpanellabel.setForeground(Color.BLACK);
```

```
        workinghourstutorpanellabel.setFont(new Font("Times New
Roman",Font.PLAIN,20));
        tutorpanel.add(workinghourstutorpanellabel);

        workinghourstexttutor = new JTextField();
        workinghourstexttutor.setBounds(200,395,200,30);
        workinghourstexttutor.setFont(new Font("Arial",Font.ROMAN_BASELINE,14));
        workingtypetexttutor.setForeground(Color.BLACK);
        Border borderwht = BorderFactory.createLineBorder(Color.black,1);
        workinghourstexttutor.setBorder(borderwht);
        tutorpanel.add(workinghourstexttutor);

        salarytutorpanellabel = new JLabel("Salary :");
        salarytutorpanellabel.setBounds(520,75,143,33);
        salarytutorpanellabel.setForeground(Color.BLACK);
        salarytutorpanellabel.setFont(new Font("Times New Roman",Font.PLAIN,20));
        tutorpanel.add(salarytutorpanellabel);

        salarytexttutor = new JTextField();
        salarytexttutor.setBounds(720,75,200,30);
        salarytexttutor.setFont(new Font("Arial",Font.ROMAN_BASELINE,14));
        salarytexttutor.setForeground(Color.BLACK);
        Border borderst = BorderFactory.createLineBorder(Color.black,1);
        salarytexttutor.setBorder(borderst);
        tutorpanel.add(salarytexttutor);

        specializationtutorpanellabel = new JLabel("Specialization :");
        specializationtutorpanellabel.setBounds(520,150,143,33);
        specializationtutorpanellabel.setForeground(Color.BLACK);
```

```
        specializationtutorpanellabel.setFont(new Font("Times New
Roman",Font.PLAIN,20));
        tutorpanel.add(specializationtutorpanellabel);

        specializationtexttutor = new JTextField();
        specializationtexttutor.setBounds(720,150,200,30);
        specializationtexttutor.setFont(new Font("ARial",Font.ROMAN_BASELINE,14));
        specializationtexttutor.setForeground(Color.BLACK);
        Border borderspt = BorderFactory.createLineBorder(Color.black,1);
        specializationtexttutor.setBorder(borderspt);
        tutorpanel.add(specializationtexttutor);

        employmentstatustutorpanellabel = new JLabel("Employment Status :");
        employmentstatustutorpanellabel.setBounds(520,228,200,33);
        employmentstatustutorpanellabel.setForeground(Color.BLACK);
        employmentstatustutorpanellabel.setFont(new Font("Times New
Roman",Font.PLAIN,20));
        tutorpanel.add(employmentstatustutorpanellabel);

        employmentstatustexttutor = new JTextField();
        employmentstatustexttutor.setBounds(720,228,200,30);
        employmentstatustexttutor.setFont(new
Font("Arial",Font.ROMAN_BASELINE,14));
        Border borderest = BorderFactory.createLineBorder(Color.black,1);
        employmentstatustexttutor.setBorder(borderest);
        tutorpanel.add(employmentstatustexttutor);

        academicqualificationtutorpanellabel = new JLabel("Academic Qualification :");
        academicqualificationtutorpanellabel.setBounds(520,315,200,22);
        academicqualificationtutorpanellabel.setForeground(Color.BLACK);
```

```
        academicqualificationtutorpanellabel.setFont(new Font("Times New
Roman",Font.PLAIN,20));
        tutorpanel.add(academicqualificationtutorpanellabel);

        academicqualificationtexttutor = new JTextField();
        academicqualificationtexttutor.setBounds(720,313,200,30);
        academicqualificationtexttutor.setFont(new
Font("Arial",Font.ROMAN_BASELINE,14));
        academicqualificationtexttutor.setForeground(Color.BLACK);
        Border borderaqt = BorderFactory.createLineBorder(Color.black,1);
        academicqualificationtexttutor.setBorder(borderaqt);
        tutorpanel.add(academicqualificationtexttutor);

        performanceindextutorpanellabel = new JLabel("Performance Index :");
        performanceindextutorpanellabel.setBounds(520,395,200,33);
        performanceindextutorpanellabel.setForeground(Color.BLACK);
        performanceindextutorpanellabel.setFont(new Font("Times NEw
Roman",Font.PLAIN,20));
        tutorpanel.add(performanceindextutorpanellabel);

        performanceindextexttutor = new JTextField();
        performanceindextexttutor.setBounds(720,395,200,30);
        performanceindextexttutor.setFont(new
Font("Arial",Font.ROMAN_BASELINE,14));
        performanceindextexttutor.setForeground(Color.BLACK);
        Border borderpit = BorderFactory.createLineBorder(Color.black,1);
        performanceindextexttutor.setBorder(borderpit);
        tutorpanel.add(performanceindextexttutor);
```

```
addtutorbuttonpanel = new JButton("Add Tutor");
addtutorbuttonpanel.setBounds(113,500,150,30);
addtutorbuttonpanel.addActionListener(this);
tutorpanel.add(addtutorbuttonpanel);

setsalarybuttontutorpanel = new JButton("Set Salary");
setsalarybuttontutorpanel.setBounds(113,560,150,30);
setsalarybuttontutorpanel.addActionListener(this);
tutorpanel.add(setsalarybuttontutorpanel);

removebuttontutorpanel = new JButton("Remove Tutor");
removebuttontutorpanel.setBounds(113,625,150,30);
removebuttontutorpanel.addActionListener(this);
tutorpanel.add(removebuttontutorpanel);

displaybuttontutorpanel = new JButton("Display");
displaybuttontutorpanel.setBounds(300,500,94,30);
displaybuttontutorpanel.addActionListener(this);
tutorpanel.add(displaybuttontutorpanel);

clearbuttontutorpanel = new JButton("Clear");
clearbuttontutorpanel.setBounds(300,560,94,30);
clearbuttontutorpanel.addActionListener(this);
tutorpanel.add(clearbuttontutorpanel);

backbuttontutorpanel = new JButton("Back");
backbuttontutorpanel.setBounds(878,22,103,35);
backbuttontutorpanel.addActionListener(this);
tutorpanel.add(backbuttontutorpanel);
```

```
    tutorpanel.setBackground(new Color(240,147,255));
    tutorpanel.setBounds(0,0,1000,720);
    tutorpanel.setLayout(null);




    frame.add(title);
    frame.setTitle("Teacher management system.");

    frame.setSize(1000,720);
    frame.setLayout(null);
    frame.setVisible(true);
    frame.setLocationRelativeTo(null);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setResizable(false);


}

@Override
public void actionPerformed(ActionEvent e)
{


    if (e.getSource() == lecturerbuttonbackgroundpanel) {
        frame.remove(backgroundpanel);
```

```
            frame.add(lecturerpanel);
            frame.revalidate();
            frame.repaint();
        }

        if (e.getSource() == tutorbuttonbackgroundpanel) {
            frame.remove(backgroundpanel);
            frame.add(tutorpanel);
            frame.revalidate();
            frame.repaint();
        }

        if (e.getSource()==backbuttonlecturerpanel){
            frame.add(backgroundpanel);
            frame.remove(lecturerpanel);
            frame.revalidate();
            frame.repaint();
        }

        if (e.getSource()==backbuttontutorpanel){
            frame.add(backgroundpanel);
            frame.remove(tutorpanel);
            frame.revalidate();
            frame.repaint();
        }
        if (e.getSource()==exitbuttonbackgroundpanel) {
            System.exit(0);
        }

        if (e.getSource() == addbuttonlecturerpanel) {
```

```
        try {
            int teacherid = Integer.parseInt(teacheridtextlecturer.getText());
            String teachername = teachernametextlecturer.getText();
            String address = addresstextlecturer.getText();
            String workingtype = workingtypetextlecturer.getText();
            String employmentstatus = employmentstatustextlecturer.getText();
            int workinghours = Integer.parseInt(workinghourstextlecturer.getText());
            String department = departmenttextlecturer.getText();
            int yearsofexperience =
Integer.parseInt(yearsofexperiencetextlecturer.getText());

            if (teacheridtextlecturer.getText().isEmpty() || teachername.isEmpty() ||
                address.isEmpty() || workingtype.isEmpty() ||
                employmentstatus.isEmpty() ||
workinghourstextlecturer.getText().isEmpty() ||
                department.isEmpty() ||
yearsofexperiencetextlecturer.getText().isEmpty()) {
                JOptionPane.showMessageDialog(lecturerpanel, "The fields is empty." +
"\n Please fill in all the blank
fields.","Message.",JOptionPane.INFORMATION_MESSAGE);
            } else {

                boolean teacheridExists = false;
                for (Teacher Lecturer : teacherlist) {
                    if (Lecturer.getTeacherId() == teacherid) {
                        teacheridExists = true;
                        break;
                    }
                }
```

```
                if (teacheridExists) {
                    JOptionPane.showMessageDialog(lecturerpanel, "Teacher ID already
exists. Please, fill in the unique ID.","Message.",JOptionPane.ERROR_MESSAGE);
                }
                else {
                    Lecturer Lecturer = new Lecturer(teacherid, teachername, address,
workingtype, employmentstatus, workinghours, department, yearsofexperience);



                    teacherlist.add(Lecturer);

                    JOptionPane.showMessageDialog(lecturerpanel, "Lecturer details
added successfuly.","Lectur Details.", JOptionPane.INFORMATION_MESSAGE);



                    teacheridtextlecturer.setText("");
                    teachernametextlecturer.setText("");
                    addresstextlecturer.setText("");
                    workingtypetextlecturer.setText("");
                    workinghourstextlecturer.setText("");
                    yearsofexperiencetextlecturer.setText("");
                    employmentstatustextlecturer.setText("");
                    departmenttextlecturer.setText("");
                    gradedscoretextlecturer.setText("");
                }
            }
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(lecturerpanel, "Invalid input
values."+"\n(Teacher ID, Working hours, and Years of
experience.)","Error",JOptionPane.ERROR_MESSAGE);
```

```
        }
    }

    if (e.getSource() == gradedassignmentsbuttonlecturerpanel) {
        try {

            int teacherid = Integer.parseInt(teacheridtextlecturer.getText());
            int gradedscore = Integer.parseInt(gradedscoretextlecturer.getText());
            String department = departmenttextlecturer.getText();
            int yearsofexperience =
Integer.parseInt(yearsofexperiencetextlecturer.getText());



            if (teacheridtextlecturer.getText().isEmpty() ||
gradedscoretextlecturer.getText().isEmpty() || department.isEmpty() ||
yearsofexperiencetextlecturer.getText().isEmpty()) {
                JOptionPane.showMessageDialog(lecturerpanel, "The text fields
empty.","Error",JOptionPane.ERROR_MESSAGE);
            } else {
                boolean found = false;

                for (Teacher teacher : teacherlist) {

                    if (teacher instanceof Lecturer && teacher.getTeacherId() ==
teacherid) {
                        ((Lecturer)teacher).gradedAssignment(gradedscore, department,
yearsofexperience);
                        found = true;
```

```
                    JOptionPane.showMessageDialog(lecturerpanel, "Your graded
assignments has been
successfully.","Message",JOptionPane.INFORMATION_MESSAGE);
                    break;
                }
            }

            if (!found) {
                JOptionPane.showMessageDialog(lecturerpanel, "TeacherID not
found.");
            }

            teacheridtextlecturer.setText("");
            teachernametextlecturer.setText("");
            addresstextlecturer.setText("");
            workingtypetextlecturer.setText("");
            workinghourstextlecturer.setText("");
            yearsofexperiencetextlecturer.setText("");
            employmentstatustextlecturer.setText("");
            departmenttextlecturer.setText("");
            gradedscoretextlecturer.setText("");
        }
    } catch (NumberFormatException ex) {

        JOptionPane.showMessageDialog(lecturerpanel, "Invalid input. Please
enter valid values.", "Error", JOptionPane.ERROR_MESSAGE);
    }
}

if (e.getSource() == displaybuttonlecturerpanel) {
```

```
        try {
            int teacherid = Integer.parseInt(teacheridtextlecturer.getText());
            if (teacheridtextlecturer.getText().isEmpty()) {
                JOptionPane.showMessageDialog(lecturerpanel, "The text field is
empty", "Error", JOptionPane.ERROR_MESSAGE);
            } else {
                boolean lecturerfound = false;

                for (Teacher teacher : teacherlist) {
                    if (teacher instanceof Lecturer && teacher.getTeacherId() ==
teacherid) {

                        Lecturer lecturer = (Lecturer) teacher;
                        lecturerfound = true;
                        if (lecturer.getGradedScore() == 0) {

                            JOptionPane.showMessageDialog(lecturerpanel,
                                    "Teacher ID: " + lecturer.getTeacherId() + "\n" +
                                            "Teacher Name: " + lecturer.getTeachername() + "\n" +
                                            "Address: " + lecturer.getAddress() + "\n" +
                                            "Working Type: " + lecturer.getWorkingtype() + "\n" +
                                            "Working Hours: " + lecturer.getWorkinghours() + "\n" +
                                            "Years of Experience: " +
lecturer.getyearsOfExperience() + "\n" +
                                            "Department: " + lecturer.getdepartment() + "\n" +
                                            "Employment Status: " +
lecturer.getEmployment_Status() + "\n" +
                                            "Graded Score:" + lecturer.getGradedScore(),
                                    "Lecturer Details",
JOptionPane.INFORMATION_MESSAGE);
```

```
                    } else {
                        JOptionPane.showMessageDialog(lecturerpanel,
                                "Teacher ID: " + lecturer.getTeacherId() + "\n" +
                                        "Teacher Name: " + lecturer.getTeachername() + "\n" +
                                        "Address: " + lecturer.getAddress() + "\n" +
                                        "Working Type: " + lecturer.getWorkingtype() + "\n" +
                                        "Working Hours: " + lecturer.getWorkinghours() + "\n" +
                                        "Years of Experience: " +
lecturer.getyearsOfExperience() + "\n" +
                                        "Department: " + lecturer.getdepartment() + "\n" +
                                        "Employment Status: " +
lecturer.getEmployment_Status() + "\n" +
                                        "Graded Score: " + lecturer.getGradedScore(),
                                "Lecturer Details",
JOptionPane.INFORMATION_MESSAGE);
                    }
                    break;
                }
            }
            if (!lecturerfound) {
                JOptionPane.showMessageDialog(lecturerpanel,
                        "No lecturer details is added.", "Empty List",
JOptionPane.INFORMATION_MESSAGE);
            }
            teacheridtextlecturer.setText("");
            teachernametextlecturer.setText("");
            addresstextlecturer.setText("");
            workingtypetextlecturer.setText("");
            workinghourstextlecturer.setText("");
            yearsofexperiencetextlecturer.setText("");
```

```
                employmentstatustextlecturer.setText("");
                departmenttextlecturer.setText("");
                gradedscoretextlecturer.setText("");
            }
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(lecturerpanel, "Invalid input. Please
enter a numeric value in Teacher ID.", "Error", JOptionPane.ERROR_MESSAGE);
        }
    }


    if (e.getSource() == clearbuttonlecturerpanel) {

        teacheridtextlecturer.setText("");
        teachernametextlecturer.setText("");
        addresstextlecturer.setText("");
        workingtypetextlecturer.setText("");
        workinghourstextlecturer.setText("");
        yearsofexperiencetextlecturer.setText("");
        employmentstatustextlecturer.setText("");
        departmenttextlecturer.setText("");
        gradedscoretextlecturer.setText("");
    }

    if(e.getSource() == addtutorbuttonpanel){
        try{
            int teacherid = Integer.parseInt(teacheridtexttutor.getText());
            String teachername = teachernametexttutor.getText();
            String address = addresstexttutor.getText();
            String workingtype = workingtypetexttutor.getText();
```

```
            int workinghours = Integer.parseInt(workinghourstexttutor.getText());
            double salary = Double.parseDouble(salarytexttutor.getText());
            String specialization = specializationtexttutor.getText();
            String employmentstatus = employmentstatustexttutor.getText();
            String academicqualifications = academicqualificationtexttutor.getText();
            int performanceindex =
Integer.parseInt(performanceindextexttutor.getText());



            if((teacheridtexttutor.getText()).isEmpty() || teachername.isEmpty() ||
address.isEmpty() || workingtype.isEmpty() ||
workinghourstexttutor.getText().isEmpty() || salary<=0.0|| specialization.isEmpty() ||
employmentstatus.isEmpty() || academicqualifications.isEmpty() ||
performanceindextexttutor.getText().isEmpty()){
                JOptionPane.showMessageDialog(tutorpanel,"The text fields
empty.","Message",JOptionPane.INFORMATION_MESSAGE);
            }else{

                boolean teacheridExists = false;
                for(Teacher Tutor : teacherlist){
                    if(Tutor.getTeacherId() == teacherid){
                        teacheridExists = true;
                        break;
                    }
                }

                if(teacheridExists){
                    JOptionPane.showMessageDialog(tutorpanel,"Teacher id already
exists.","TeacherId same found.",JOptionPane.ERROR_MESSAGE);
                }
```

```
                else{

                    Tutor Tutor = new Tutor(teacherid, teachername, address,
workingtype, employmentstatus, workinghours, salary, specialization,
academicqualifications, performanceindex);

                    teacherlist.add(Tutor);

                    JOptionPane.showMessageDialog(tutorpanel,"Tutor add
successfully.","Tutor details",JOptionPane.INFORMATION_MESSAGE);

                    teacheridtexttutor.setText("");
                    teachernametexttutor.setText("");
                    addresstexttutor.setText("");
                    workingtypetexttutor.setText("");
                    workinghourstexttutor.setText("");
                    salarytexttutor.setText("");
                    specializationtexttutor.setText("");
                    employmentstatustexttutor.setText("");
                    academicqualificationtexttutor.setText("");
                    performanceindextexttutor.setText("");
                }
            }
        }catch (NumberFormatException ex){
            JOptionPane.showMessageDialog(tutorpanel,"Invalid input
values.","Invalid",JOptionPane.ERROR_MESSAGE);
        }
    }
```

```
        if(e.getSource() == setsalarybuttontutorpanel){
            try{
                int teacherid = Integer.parseInt(teacheridtexttutor.getText());
                double salary = Double.parseDouble(salarytexttutor.getText());
                int performanceindex =
Integer.parseInt(performanceindextexttutor.getText());



                if(teacheridtexttutor.getText().isEmpty() || salary <=0.0 ||
performanceindextexttutor.getText().isEmpty())
                {
                    JOptionPane.showMessageDialog(tutorpanel,"Fields are empty
founds.","Empty",JOptionPane.INFORMATION_MESSAGE);
                }
                else
                {

                    boolean found = false;
                    for (Teacher teacher : teacherlist)
                    {
                        if (teacher instanceof Tutor && teacher.getTeacherId() == teacherid)
                        {
                            Tutor tutor = (Tutor) teacher;
                            found = true;

                            tutor.setSalary(salary, performanceindex);
                            JOptionPane.showMessageDialog(tutorpanel,"Your set salary is
successfully.","Message",JOptionPane.INFORMATION_MESSAGE);

                            break;
```

```
                    }
                }

                if(! found){
                    JOptionPane.showMessageDialog(tutorpanel,"TeacherID has not
been match.","Error",JOptionPane.ERROR_MESSAGE);
                }

                teacheridtexttutor.setText("");
                teachernametexttutor.setText("");
                addresstexttutor.setText("");
                workingtypetexttutor.setText("");
                workinghourstexttutor.setText("");
                salarytexttutor.setText("");
                specializationtexttutor.setText("");
                employmentstatustexttutor.setText("");
                academicqualificationtexttutor.setText("");
                performanceindextexttutor.setText("");
            }
        }catch(NumberFormatException ex){

            JOptionPane.showMessageDialog(tutorpanel, "Invalid input
values","Error",JOptionPane.ERROR_MESSAGE);
        }
    }


    if (e.getSource() == removebuttontutorpanel) {
        try {
            int teacherID = Integer.parseInt(teacheridtexttutor.getText());
```

```
            if (teacherID == 0) {
                JOptionPane.showMessageDialog(null, "Teacher ID is empty.", "Empty
Teacher ID", JOptionPane.ERROR_MESSAGE);
            } else {
                boolean teacherFound = false;
                Teacher Tutorremove = null;

                for (Teacher teacher : teacherlist) {
                    if (teacher.getTeacherId() == teacherID) {
                        if (teacher instanceof Tutor) {
                            Tutor tutor = (Tutor) teacher;

                            if (tutor.getIsCertified()) {
                                JOptionPane.showMessageDialog(null, "Certified tutors cannot
be removed.", "Error", JOptionPane.ERROR_MESSAGE);
                                return;
                            } else {
                                teacherFound = true;
                                Tutorremove = tutor;
                                break;
                            }
                        }
                    }
                }
                if (teacherFound) {

                    ((Tutor) Tutorremove).removeTutor();
                    teacherlist.remove(Tutorremove);
                    JOptionPane.showMessageDialog(null, "Tutor removed successfully.",
"Success", JOptionPane.INFORMATION_MESSAGE);
```

```
                } else {

                    JOptionPane.showMessageDialog(null, "Teacher ID not found.",
"Error", JOptionPane.ERROR_MESSAGE);
                }
                teacheridtextlecturer.setText("");
                teachernametextlecturer.setText("");
                addresstextlecturer.setText("");
                workingtypetextlecturer.setText("");
                workinghourstextlecturer.setText("");
                yearsofexperiencetextlecturer.setText("");
                employmentstatustextlecturer.setText("");
                departmenttextlecturer.setText("");
                gradedscoretextlecturer.setText("");
            }
        } catch (NumberFormatException ex) {

            JOptionPane.showMessageDialog(tutorpanel, "Invalid input value for
Teacher ID.", "Error", JOptionPane.ERROR_MESSAGE);
        }
    }


    if (e.getSource() == displaybuttontutorpanel) {
        try{

            int teacherid =Integer.parseInt(teacheridtexttutor.getText());

            if(teacheridtexttutor.getText().isEmpty()){
```

```
                JOptionPane.showMessageDialog(tutorpanel,"TeacherId
empty.","Empty",JOptionPane.INFORMATION_MESSAGE);
            }
            else{
                boolean tutorfound = false;

                for(Teacher teacher : teacherlist){
                    if(teacher instanceof Tutor && teacher.getTeacherId()== teacherid){
                        Tutor tutor = (Tutor)teacher;
                        tutorfound = true;

                        JOptionPane.showMessageDialog(lecturerpanel,
                        "Teacher ID: " + tutor.getTeacherId() + "\n" +
                        "Teacher Name: " + tutor.getTeachername() + "\n" +
                        "Address: " + tutor.getAddress() + "\n" +
                        "Working Type: " + tutor.getWorkingtype() + "\n" +
                        "Working Hours: " + tutor.getWorkinghours() + "\n" +
                        "Salary: " + tutor.getSalary() + "\n" +
                        "Specialization: " + tutor.getSpecialization() + "\n" +
                        "Employment Status: " + tutor.getEmployment_Status()+ "\n"+
                        "Academic Qualification: " + tutor.getAcademicqualifications() + "\n"
+
                        "Performance index:" + tutor.getPerformanceIndex(),
                        "Tutor Details", JOptionPane.INFORMATION_MESSAGE);
                    break;
                    }
                }

                if(!tutorfound){
```

```
                    JOptionPane.showMessageDialog(tutorpanel, "Invalid
teacherId","Message",JOptionPane.ERROR_MESSAGE);
                }

                teacheridtexttutor.setText("");
                teachernametexttutor.setText("");
                addresstexttutor.setText("");
                workingtypetexttutor.setText("");
                workinghourstextlecturer.setText("");
                yearsofexperiencetextlecturer.setText("");
                employmentstatustexttutor.setText("");
                performanceindextexttutor.setText("");
                academicqualificationtexttutor.setText("");
                salarytexttutor.setText("");
            }
        }catch(NumberFormatException ex){

            JOptionPane.showMessageDialog(tutorpanel," invalid teacherid.");
        }
    }

    if(e.getSource() == clearbuttontutorpanel){

        teacheridtexttutor.setText("");
        teachernametexttutor.setText("");
        addresstexttutor.setText("");
        workingtypetexttutor.setText("");
        workinghourstextlecturer.setText("");
        yearsofexperiencetextlecturer.setText("");
        employmentstatustexttutor.setText("");
```

```
            performanceindextexttutor.setText("");
            academicqualificationtexttutor.setText("");
            salarytexttutor.setText("");
        }
    }

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