



CS4001NI Programming

30% Individual Coursework

2023-24 Autumn

Student Name: Sudeshna Prajapati

London Met ID: 23050288

College ID: NP01CP4A230403

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.

Tables of Contents

Contents

1.Introduction	6
1.1 Brief on Coursework	6
1.2 Tools Used	6
2. Class Diagram.....	7
2.1 Teacher Class.....	7
2.2 Lecture Class.....	8
2.3 Tutor Class	9
2.4 TeacherGUI Class	10
2.5 Combination of all four classes	11
3. Pseudocode	12
3.1 TeacherGUI	12
4. Method Description	26
4.1 If(e.getSource()==lecturerButton)	26
4.2.If(e.getSource()==tutorButton	26
4.3. If(e.getSource()==gradeButton	26
4.4. If(e.getSource()==setSalaryButton	26
4.5 If(e.getSource()==removeTutorButton).....	27
4.6 If(e.getSource()==displayButton)	27
4.7 If(e.getSource()==displayButton1)	27
4.8 If(e.getSource()==clearButton)	27
4.9 If(e.getSource()==clearButton1)	27
5. Testing	28
5.1 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.	28
5.2 Test 2: : Evidences should be shown of:	30
5.3 Test 3: Test that appropriate dialog boxes appear when unsuitable values are entered for the Teacher ID, (include a screenshot of the dialog box, together with a corresponding screenshot of the GUI, showing the values that were entered)	43

6. Error Detection	47
6.1 Syntax Error.....	47
6.1.1 Syntax Error Detection.....	48
6.2 Logical Error	49
6.2.1 Logical Error Detection	50
7. Conclusion	51
7.1 Evaluation of the work	51
7.2 Reflection on what I learned from the coursework.....	51
7.3 Difficulties I encountered	51
7.4 How I overcame the difficulties	51
8.References.....	51
9.Appendix	53
9.1 Appendix of Teacher class	53
9.2.Code of Lecturer Class	57
9.3. Code of the Tutor class.....	62
9.4 Appendix of TeacherGUI class	67

Tables of Figures

Figure 1: Compiling all java files.....	29
Figure 2: Running GUI	29
Figure 3: Filling the textfield of lecturer.....	31
Figure 4: Lecturer added	32
Figure 5: Filling the textfield of gradeAssignment.....	34
Figure 6: Assignment graded successfully	35
Figure 7: Filling the textfield of Tutor	36
Figure 8: Tutor added successfully	37
Figure 9: filling the textfield for set salary	38
Figure 10: New salary has been assigned	39
Figure 11: New salary has been updated.....	40
Figure 12: Filling textfield for remove Tutor	41
Figure 13: Result of removeTutor.....	42
Figure 14:Lecturer button pressed.	44
Figure 15: Add lecturer was clicked with unfilled values.....	45
Figure 16:Press add lecturer again after adding	46
Figure 17: Error of syntax.....	47
Figure 18: Error Detection of syntax error	48
Figure 19: Logical error	49
Figure 20: Error detection of logical error	50

Table of Tables

Table 1: Class diagram of Teacher	7
Table 2: Class diagram of lecture.....	8
Table 3: Class diagram of tutor	9
Table 4: Class diagram of TeacherGUI	10
Table 6: Testing 1	28
Table 7: Testing 2	30
Table 8: Testing 3	43

1.Introduction

1.1Brief on Coursework

In this coursework ,I made GUI which contains all the given components. I made TeacherGUI class which holds lecturer and tutor objects. Then I made text fields for the GUI. I also implemented add lecturer button in which when the button is pressed the input values of the given components is added to an array list of Teacher class. I made a add Tutor button where the input values are used to create a new object of the type Tutor which is added to the array list of Teacher class. I also made the button for grade assignments in which the input value of teacher id is compared to the existing teacher id and if valid teacher id is given it is used to generate the grade assignments from lecturer class. The set salary of Tutor button set the salary of Tutor when valid teacher id has been entered. Then I made the remove Tutor button to remove tutor when the valid teacher id is entered. Also I made the display button to display the information in a appropriate way and at last a clear button to clear all the values from the text fields.

1.2 Tools Used

I have used Blue J to write my java code and compile the program. Blue J is a development environment that allows you to develop java programs quickly and in a efficient way. Blue J is a windows based platforms for Java Development kit(JDK). I have also used MS-Word to write my report of the coursework. I also used draw.io to make ta class diagram.

2. Class Diagram

2.1 Teacher Class

Teacher
<div><div>-teacher\$name: String</div><div>-address: String</div><div>-working\$type: String</div><div>-employment\$status: String</div><div>-teacher\$id: int</div><div>-working\$hours: int</div></div>
<div><div>+<<constructor>>Teacher(teacher\$id: int, teacher\$name: String, address: String, working\$type: String, employment\$status: String)</div><div>+getTeacher\$id(): int</div><div>+ getTeacher\$name(): String</div><div>+ getAddress(): String</div><div>+getEmployment\$status(): String</div><div>+ getWorking\$type(): String</div><div>+ getWorking\$hours(): int</div><div>+ setWorking\$hours(working\$hours: int): void</div><div>+ display(): void</div></div>

Table 1: Class diagram of Teacher

2.2 Lecture Class

Lecturer
-department: String -yearsofexperience: int -gradedscore: int -hasgraded: boolean
+<<constructor>>Lecturer(department: String, yearsofexperience: int,gradedscore: int,hasgraded: Boolean) +getDepartment(): String +getYearsofexperience():int +getGradedscore(): int +getHasgraded(): int +setGradedscore(gradedscore: int): void +Gradeassignment(): void +display(): void

Table 2: Class diagram of lecture

2.3 Tutor Class

Tutor
<ul style="list-style-type: none">-salary: double-specialization: String-academic\$qualification: String-performance\$index: int-iscertified: boolean
<ul style="list-style-type: none">+<<constructor>>Tutor(salary: double,specialization: String, academic\$qualifications: String,performance\$index:int)+getSalary(): double+getSpecialization(): String+getAcademic\$qualifications(): String+getPerformance\$index(): int+getIscertified(): Boolean+setSalary(salary: double, performance\$index:int): void+removeTutor(): void+display(): void

Table 3: Class diagram of tutor

2.4 TeacherGUI Class

TeacherGUI
<p>Frame1:JFrame</p> <p>lecturerButton,gradeButton,displayButton,clearButton,displayButton1,clearButton1,tutorButton,setSalaryButton,removeTutorButton:JButton</p> <p>teacherIdField,teacherNameField,addressField,workingTypeField,employmentStatusField,departmentField,yearsOfExperienceField,workingHoursField,gradedScoreField,teacherIdField2,teacherNameField2,addressField2,workingTypeField2,employmentStatusField2,workingHoursField2,salaryField,specializationField,academicQualificationsField,performanceIndexField:JTextField</p> <p>lecturerPanel, TutorPanel: JPanel</p> <p>al1: ArrayList</p>
<p>+<<constructor>>TeacherGUI()</p> <p>-actionPerformed(ActionEvent e):void</p> <p>-main(String[]args):void</p>

Table 4: Class diagram of TeacherGUI

2.5 Combination of all four classes

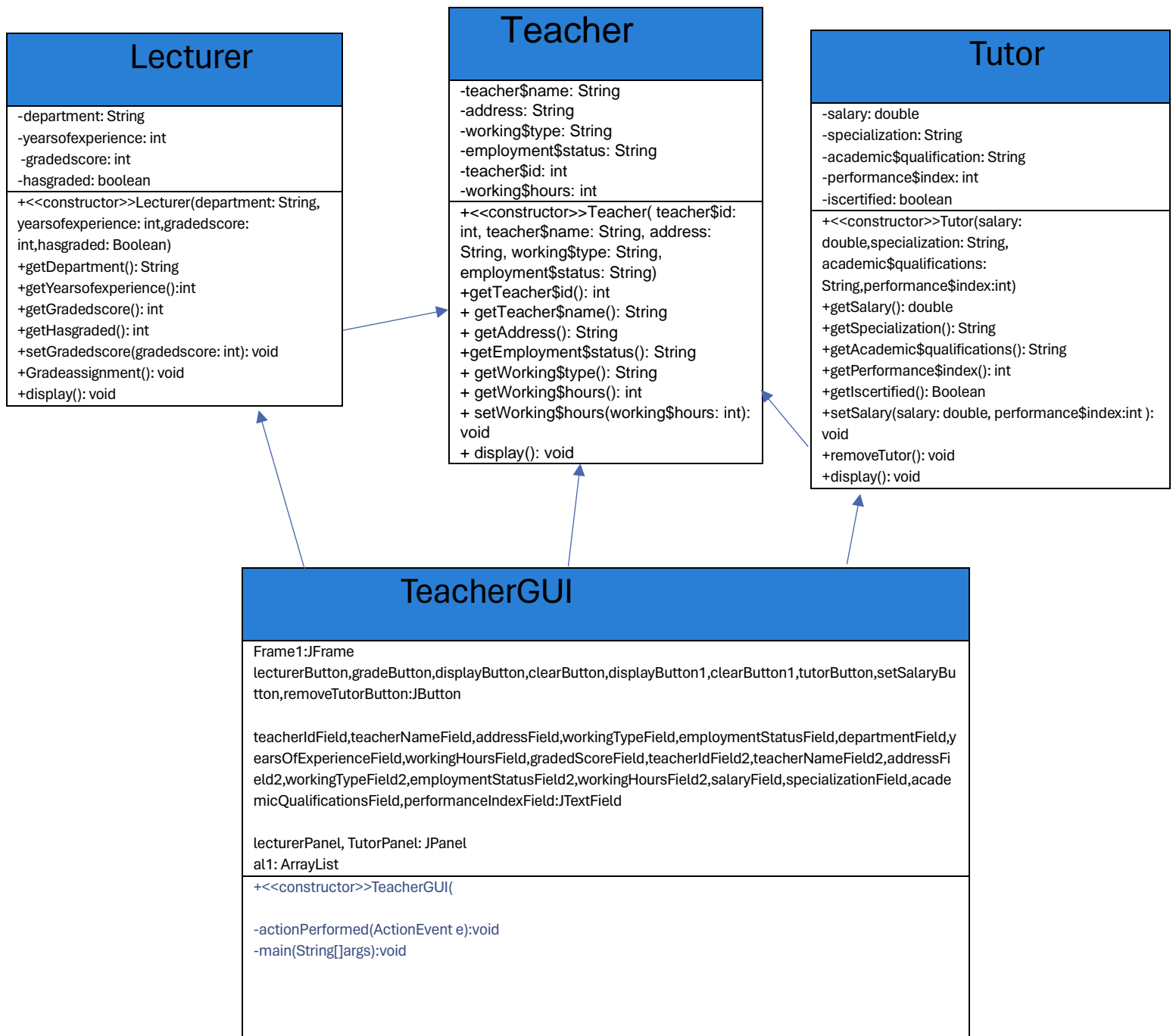


Table 5: Combination of all four classes:

3. Pseudocode

3.1 TeacherGUI

IMPORT necessary libraries

DECLARE class TeacherGUI extents JFrame

GENERATE frame1

SET the background color to lecturerpanel

GENERATE all the JLabels components for lecture panel

GENERATE all the JLabels components for Tutor panel

GENERATE all the JTextFields components for lecture panel

GENERATE all the JTextFields components for tutor panel

GENERATE all the JButton given

CALL the arraylist Teacher

DECLARE instance variable TeacherGUI

Al1 is equals to new ArrayList Teacher

JTabbedPane tabbedPane is equals to new JTabbedPane

CREATE the lecturer tab

JFrame frame1

SET background color to tutor panel

INITIALIZE all the JLabel components for lecturer tab

INITIALIZE all the JLabel components for tutor tab

DECLARE all the JButton components for lecturer tab

DECLARE all the JButtons components for tutor tab

ASSIGN all the JTextFields components for lecturer tab

ASSIGN all the JTextFields components for tutor tab

SET all the setBounds for JTextField of lecturer tab

SET all the setBounds for JTextField of tutor tab

SET all the setBounds for JLabels of lecture

SET all the setBounds for JLabels of tutor

SET all the setBounds for JButtons of lecture

SET all the setBounds for JButtons of tutor

ADD all the JLabels components to lecturer panel

ADD all the JLabels components to tutor panel

ADD all the JTextFields components to lecturer panel

ADD all the JTextFields components to tutor panel

ADD all the JButtons for lecturer panel

ADD all the JButtons for tutor panel

ADD all the JButtons to addActionListner

SET the tutor panel.setLayout to null

SET the lecturer panel.setLayout to null

ASSIGN setVisible to true

ASSIGN DefaultCloseOperation to JFrame.EXIT_ON_CLOSE

ASSIGN setSize to 600,400

@Override

DECLARE a public method named actionPerformed that takes parameter ActionEvent object called e

IF e.getSource is equals to lecturerButton

IF teacherIdField.getText.isEmpty or
teacherNameField.getText.isEmpty.addressField.getText.isEmpty.workingTypeField.getText.isEmpty.employmentStatusField.getText.isEmpty.workingHoursField.getText.isEmpty.departmentField.getText.isEmpty.yearsOfExperienceField.getText.isEmpty

JOptionPane.showMessageDialog frame1,empty textfield found, error
please fill up the given information

ELSE

TRY

int teacherId is equals to Integer.parseInt(teacherIdField.getText

String teacherName is equals to teacherNameField.getText

String address is equals to addressField.getText

String workingType is equals to workingTypeField.getText

String employmentStatus is equals to
employmentStatusField.getText

int workingHours is equals to
Integer.parseInt(workingHoursField.getText

String department is equals to departmentField.getText

int yearsOfExperience is equals to
Integer.parseInt(yearsOfExperienceField.getText

```
Lecturer lecturerobj is equals to new  
Lecturer(teacherId,teacherName,  
address,workingType,employmentStatus,department,yearsOfExper  
ience ,workingHours)
```

```
IF al1 isEmpty
```

```
    Al1.add to lecturerobj
```

```
    JOptionPane.showMessageDialog(frame1 lecturer is added)
```

```
END if
```

```
ELSE
```

```
    DECLARE Boolean add is equals to true
```

```
    FOR Teacher Tobj to al1
```

```
        IF Tobj instanceof Lecturer
```

```
            IF teacherId is equals to Tobj.getTeacher$id
```

```
                SHOW
```

```
                JOptionPane.showMessageDialog
```

```
                frame1, this id has been already used.
```

```
                Add is equals to false
```

```
            END if
```

```
        END if
```

```
    IF add is equals to true
```

```
        Al1.add lecturer obj
```

```
        JOptionPane.showMessageDialog (frame1,teacher id  
is added)
```

ELSE

JOptionPane.showMessageDialog(frame1, teacher id
is already exist.

END IF

Catch NumberFormatException

JOptionPane.showMessageDialog frame1,error message , invalid input

END catch

END IF

ELSE IF(e.getSource is equals to tutorbutton

TRY

int teacherId is equals to Integer.parseInt(teacherIdField2.getText());

String teacherName is equals to teacherNameField2.getText();

String address is equals to addressField2.getText();

String workingType is equals to workingTypeField2.getText();

String employmentStatus is equals to employmentStatusField2.getText();

int workingHours is equals to

Integer.parseInt(workingHoursField2.getText());

double salary=Double.parseDouble(salaryField.getText());

String specialization=specializationField.getText();

String academicQualifications= academicQualificationsField.getText();

int performanceIndex= Integer.parseInt(performanceIndexField.getText());


```
Tutor Tutorobj= new Tutor(teacherId,teacherName,  
address,workingType,employmentStatus,workingHours,salary,specializati  
on,academicQualifications,performanceIndex);
```

```
IF al1 isEmpty
```

```
    Al1.add to Tutorobj
```

```
    JOptionPane.showMessageDialog(frame1 Tutor is added)
```

```
END IF
```

```
ELSE
```

```
    DECLARE Boolean add is equals to true
```

```
    FOR Teacher Tobj to al1
```

```
        IF Tobj instanceof Tutor
```

```
            IF teacherId is equals to Tobj.getTeacher$Id
```

```
                SHOW
```

```
                JOptionPane.showMessageDialog  
                frame1, this id has been already used.
```

```
                Add is equals to false
```

```
            END IF
```

```
        END IF
```

```
    IF add is equals to true
```

```
        Al1.add lecturer obj
```

```
        JOptionPane.showMessageDialog (frame1,teacher id  
is added
```

```
    ELSE
```

JOptionPane.showMessageDialog(frame1, Tutor is added.

END IF

Catch NumberFormatException

JOptionPane.showMessageDialog frame1,error message , invalid input

END IF

ELSE IF e.getSource is equals to gradeButton

IF teacherIdField.getText.isEmpty or gradedScoreField.getText.isEmpty or departmentField.getText.isEmpty or yearsOfExperienceField.getText.isEmpty

JOptionPane.showMessageDialog frame1, empty textfield found,please fill yp the given info , JOptionPane,warning message

END IF

ELSE

int teacherId is equals to Integer.parseInt(teacherIdField.getText

String department is equals to departmentField.getText

int yearsOfExperience is equals to

Integer.parseInt(yearsOfExperienceField.getText

int gradedScore is equals to

Integer.parseInt(gradedScoreField.getText

FOR Teacher Tobj to al1

IF Tobj instanceof Lecturer

IF teacherId is equals to Tobj.getTeacher\$id

Lecturer L is equals to (Lecturer)Tobj

L.Gradeassignment(gradedscore,
department, years of experience)

SHOW

JOptionPane.showMessageDialog
frame1, this assignment has been
graded.

break

END IF

ELSE

JOptionPane.showMessageDialog(null,.
Teacher id is not a tutor

ELSE

JOptionPane.showMessageDialog(null
Teacher id is not found)

END IF

ELSE IF(e.getSource is equals to setSalary Button

IF teacherIdField2.getText.isEmpty or salaryField.getText isEmpty or
performanceIndexField.getText .isEmpty

JOptionPane.showMessageDialog frame1, empty textfield found,please fill
up the given info , JOptionPane,warning message.

ELSE

int teacherId2 is equals to Integer.parseInt teacherIdField2.getText

double salary is equals to Double.parseDouble salaryField.getText

int performanceIndex is equals to

Integer.parseInt(performanceIndexField.getText

FOR Teacher Tobj to all

IF Tobj instanceof Tutor

IF teacherId2 is equals to Tobj.getTeacher\$id

Tutor T=(Tutor)Tobj;

T.setSalary(salary,performanceIndex);

JOptionPane.showMessageDialog(frame1,"The new salary has been assigned.");

break

END IF

ELSE

JOptionPane.showMessageDialog(frame1,teacher id is not a tutor

ELSE

JOptionPane.showMessageDialog(frame1, Tutor is not found.

END IF

ELSE IF e.getSource is equals to removeTutor

Int teacherId2 is equals to parseInt
teacherIdField2.getText

IF teacherId2 is equals to 0

JOptionPane.showMessageDialog
frame1, empty textfield found,please fill
up the given info , JOptionPane,warning
message.

ELSE

Teacher tutorToRemove is equals to
null

Boolean teacherFound is equals to false

FOR Teacher Tobj to al1

IF (Tobj.getTeacher\$id I equals
to teacher Id2)

IF Tobj instanceof Tutor

Tutor T=(Tutor)Tobj;

T.setSalary(salary
,performanceIndex);

END IF

JOptionPane.showMessag
eDialog(numm.certified
tutors cannot be removed.)

Break

END IF

ELSE

JOptionPane.showMessageDialog
g(null,teachr id is not a tutor)

IF teacherfound

Tutor

tutorTOrRemove.remobeTutor

Al1.remove tutorToRemove

JOptionPane.showMessageDialog
g(null, teacher id is not found)

END IF

ELSE IF e.getSource is equals to displayButton

IF teacherIDField.getText.isEmpty

JOptionPane.showMessageDialog JOptionPane.showMessageDialog
frame1, empty textfield found,please fill up the given info ,
JOptionPane,warning message.

ELSE

TRY

Int teacherId is equals ti Integer .parseInt teacherIDField.getText

IF teacherId is equals to 0

JOptionPane.showMessageDialog frame1, empty textfield found,please fill
up the given info , JOptionPane,warning message.

FOR teacher Tobj al1

IF Tobj instanceof Lecturer

Lecturer L is equals to Lecturer Tobj

```

        IF teacherId is equals to Tobj.getTeacher$Id

            JOptionPane.showMessageDialog frame1,call all the
            parameters

            Break

        END IF

    END IF

    CATCH NumberFormatException n

        JOptionPane.showMessageDialog frame1,error message ,ERROR

ELSE IF e.getSource is equals to displayButton1

    int teacherId2 is equals to Integer.parseInt teacherIdField2.getText

    IF teacherId2 is equals to 0

        JOptionPane.showMessageDialog JOptionPane.showMessageDialog
        frame1, empty textfield found,please fill up the given info ,
        JOptionPane,warning message.

    END IF

ELSE

    FOR Teacher Tobj: al1

        IF Tobj instance of Tutor

            Tutor Tis equals to Tutor Tobj

            IF teacherId2 is equals to Tibj.getTeacher$Id

                JOptionPane.showMessageDialog frame1, call all
                parameter

                Break

            END IF

        END IF

    END FOR

END IF

```

END IF

END IF

ELSE IF e.getSource is equals to clearButton

teacherIdField.setText

teacherNameField.setText

addressField.setText

workingTypeField.setText

employmentStatusField.setText

departmentField.setText

yearsOfExperienceField.setText

workingHoursField.setText

gradedScoreField.setText

ELSE IF e.getSource is equals to clearButton1

teacherIdField2.setText

teacherNameField2.setText

addressField2.setText

workingTypeField2.setText

employmentStatusField2.setText

workingHoursField2.setText

salaryField.setText

specializationField.setText

academicQualificationsField.setText

performanceIndexField.setText

CALL the main method

New TeacherGUI

4. Method Description

4.1 If(e.getSource()==lecturerButton)

When the lecturer Button id clicked, firstly it check whether the text fields are empty or not . If the text field are empty it throws an error message but if the text field are filled then the values input of teacher id, teacher name, address, working type, employment status, graded score and years of experience creates a new object of type lecturer and then added to an array list of Teacher class.

4.2.If(e.getSource()==tutorButton

When the tutor button is pressed, it will check the empty textfield first then if the input values of teacher id, teacher name, address, working type, employment status, working hours, salary, specialization, academic qualifications and performance index creates a new object type tutor which is then added to the array list of teacher class and displays an information dialog box.

4.3. If(e.getSource()==gradeButton

While the grade assignment is clicked, the input values of teacher id is compared to the existing teacher id and if a valid teacher od has been entered, it grade the assignments from the lecturer class. It also displays all the data that has been entered with an information dialog box.

4.4. If(e.getSource()==setSalaryButton

The set salary of tutor checks whether the text field is empty or not. Then after entering the teacher id new salary and performance index it set the salary of the tutor with a dialog box indicating the new id has been assigned.

4.5 If(e.getSource()==removeTutorButton)

While the button is clicked, it checks the empty text field and if the valid teacher id , performance index and working hours has been declared then it is used to remove the tutor.

4.6 If(e.getSource()==displayButton)

When the button is pressed all the information that have been filled in the lecturer panel is displayed.

4.7 If(e.getSource()==displayButton1)

When the button is pressed , all the information that have been filled in the tutor panel is displayed.

4.8 If(e.getSource()==clearButton)

When the button is pressed, all the values from the text fields of lecture panel is cleared.

4.9 If(e.getSource()==clearButton1)

When the button is pressed, all the values from the text fields of tutor panel is cleared.

5. Testing

5.1 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.

Test No.	1
Objectives	To test that the program can be compiled and run using command prompt.
Action	<ul style="list-style-type: none">-Open cmd in the location of the java file of the package.-Compile all the four java files including parent and child class.-Inspection of class TeacherGUI.
Expected Result	TeacherGUI must be opened in cmd.
Actual Result	TeacherGUI was opened in cmd.
Conclusion	The test was successful.

Table 5: Testing 1

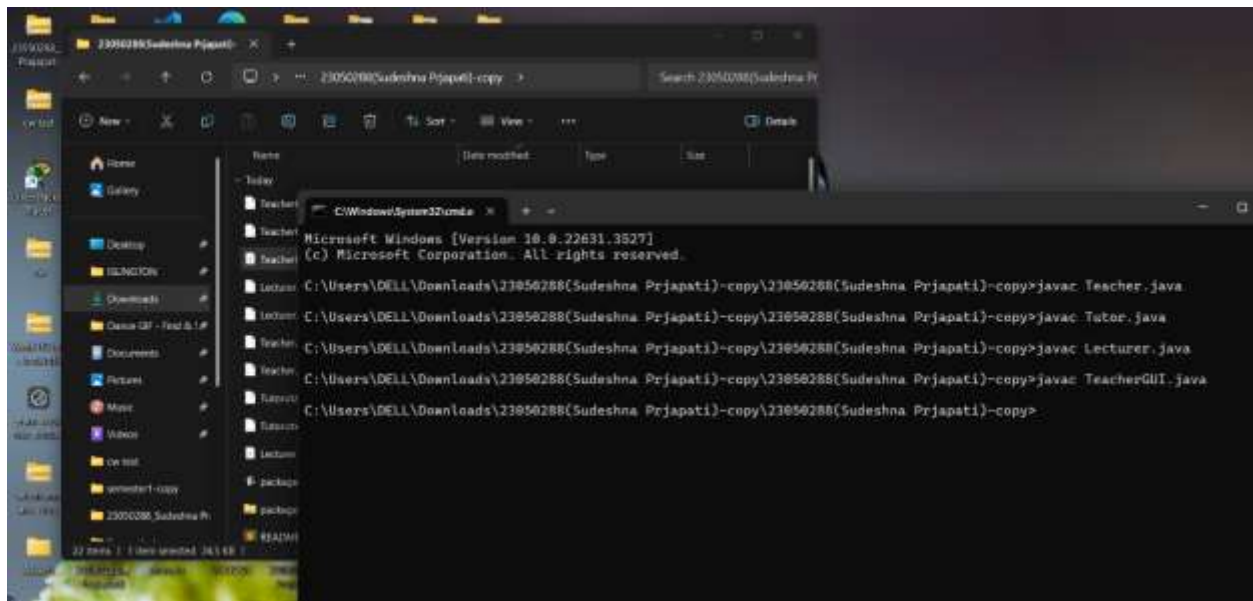


Figure 1: Compiling all java files

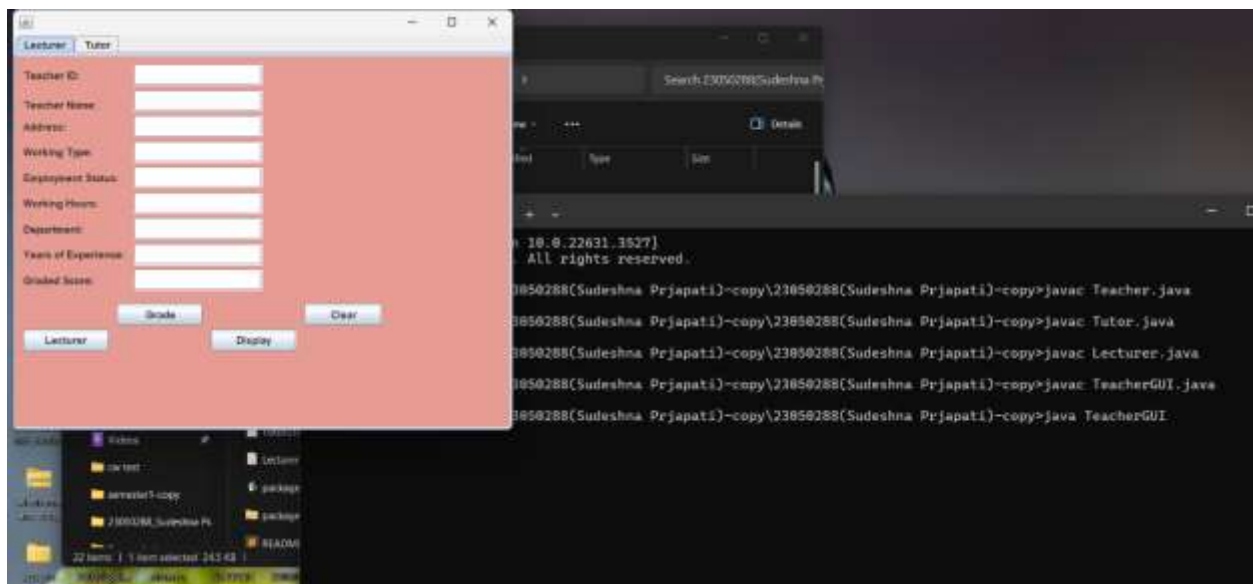


Figure 2: Running GUI

5.2 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

Test No.	2
Objectives	Evidences should be shown of: <ul style="list-style-type: none">a. Add the Lecturerb. Add the Tutorc. Grade Assignments from Lecturerd. Set the salarye. Remove the tutor
Action	<ul style="list-style-type: none">-All the text fields of lecturer were filled and the lecturer button was clicked.-All the text fields of tutor were filled and the tutor button was clicked.-All the text fields of lecturer were filled and the grade button was clicked.-RemoveTutor button was clicked.
Expected Result	All the buttons of evidences must work properly.
Actual Result	The evidences should be showed worked properly.
Conclusion	The test was successful.

Table 6: Testing 2

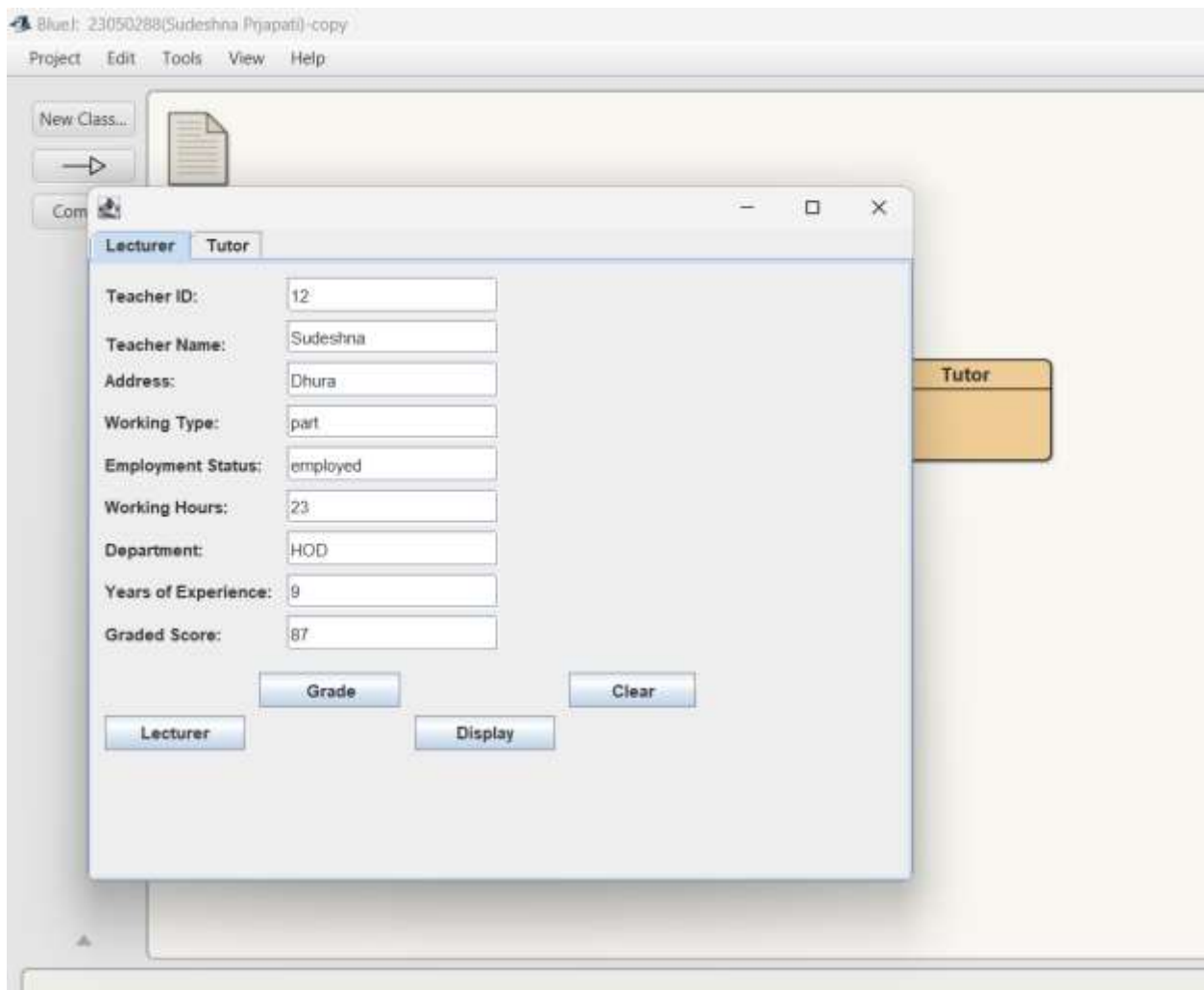


Figure 3: Filling the textfield of lecturer

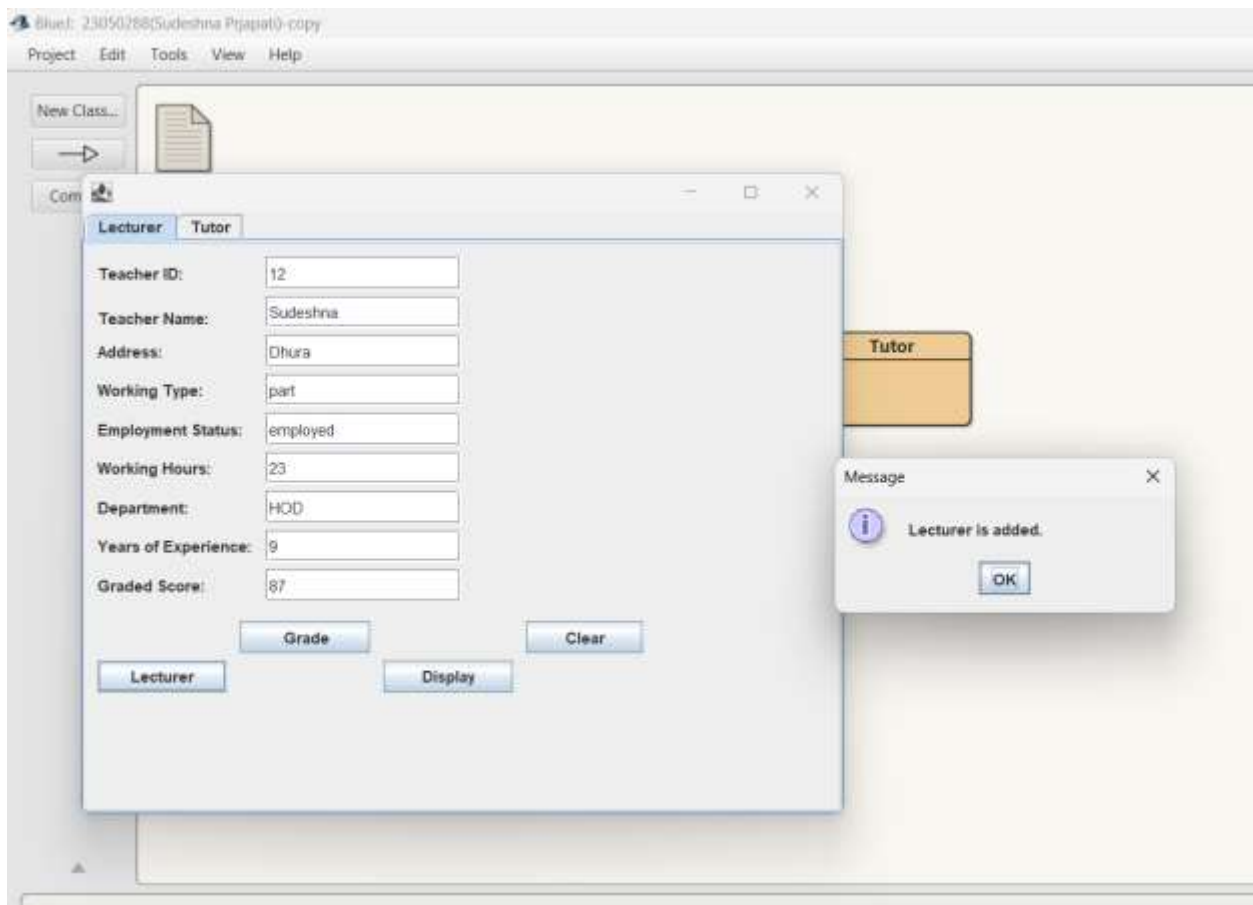
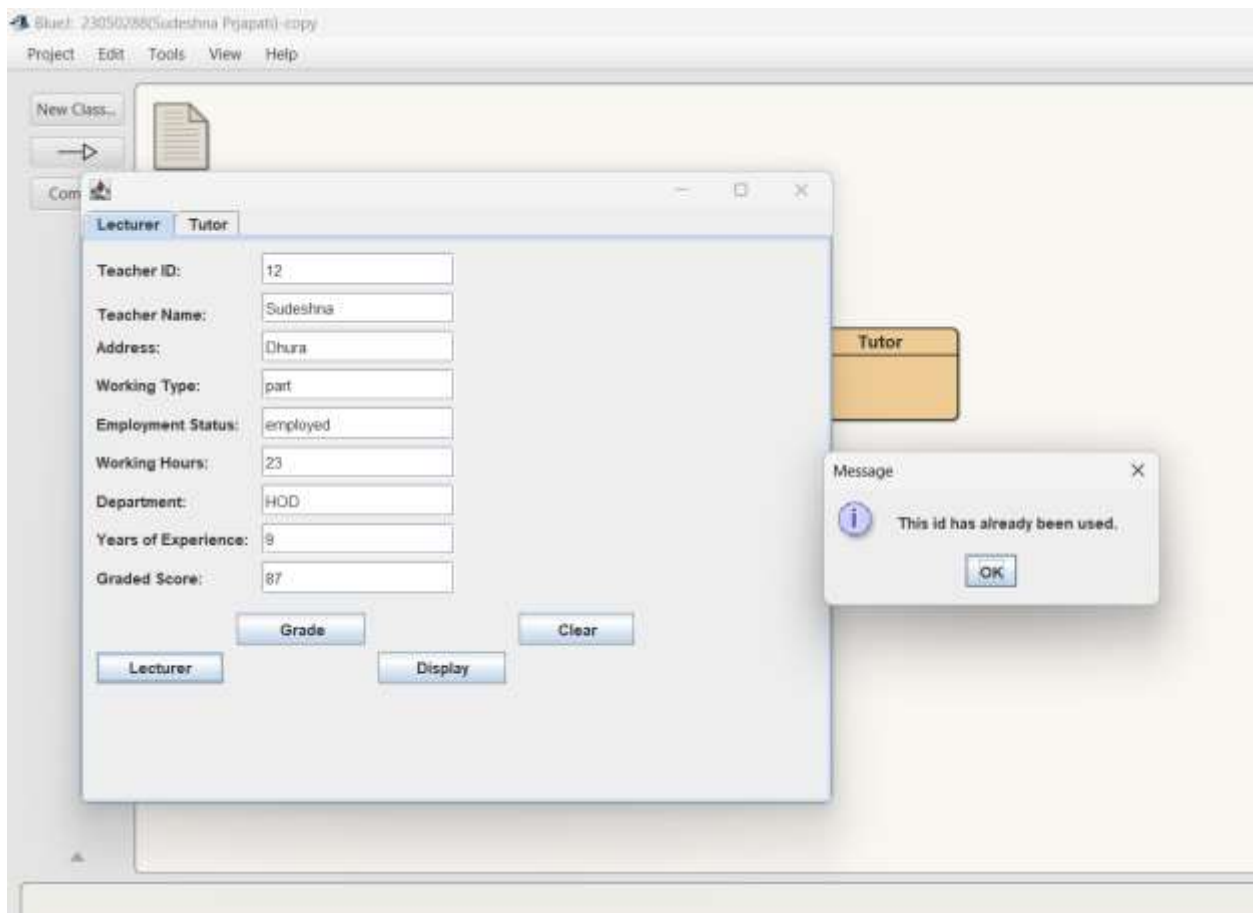


Figure 4: Lecturer added



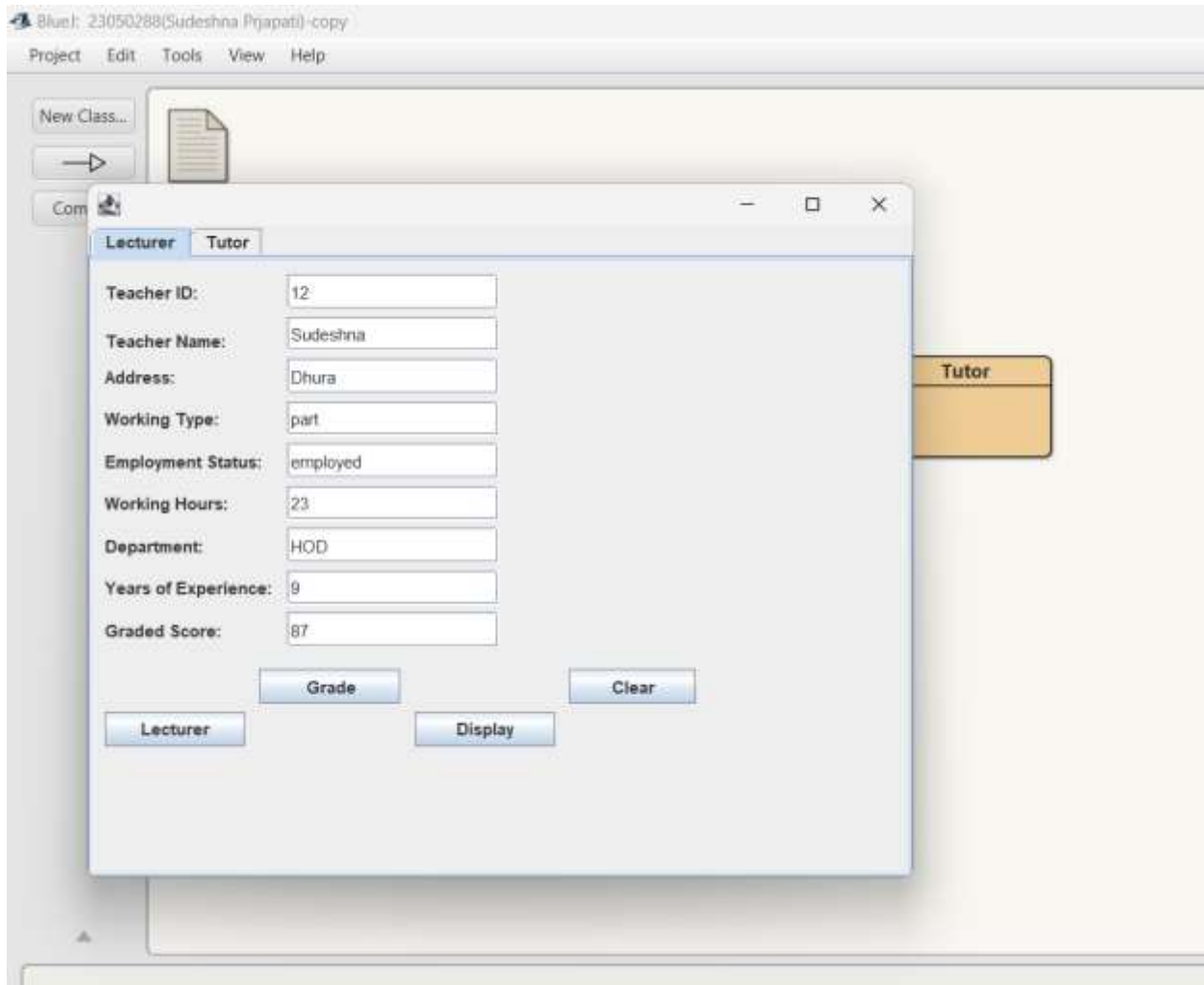


Figure 5: Filling the textfield of gradeAssignment

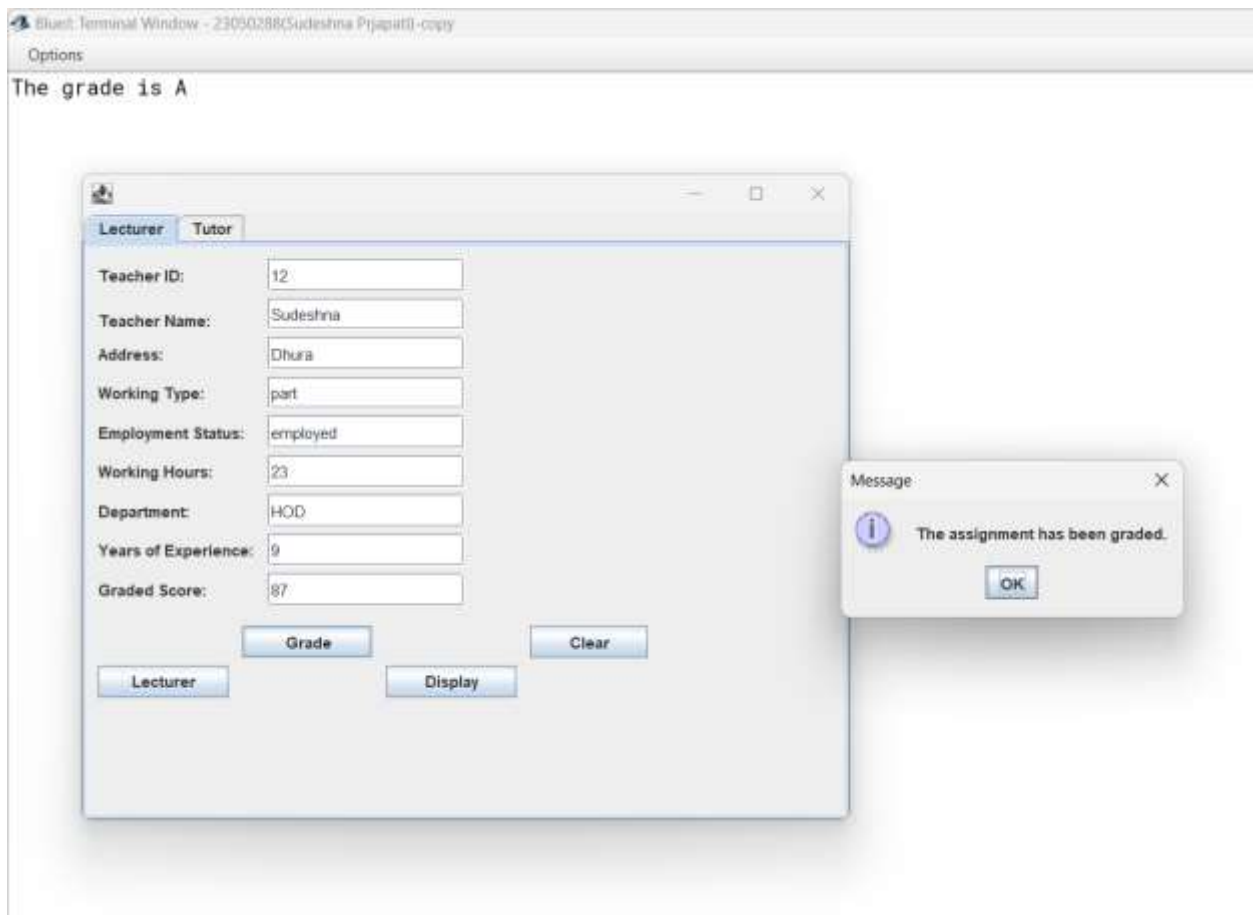


Figure 6: Assignment graded successfully

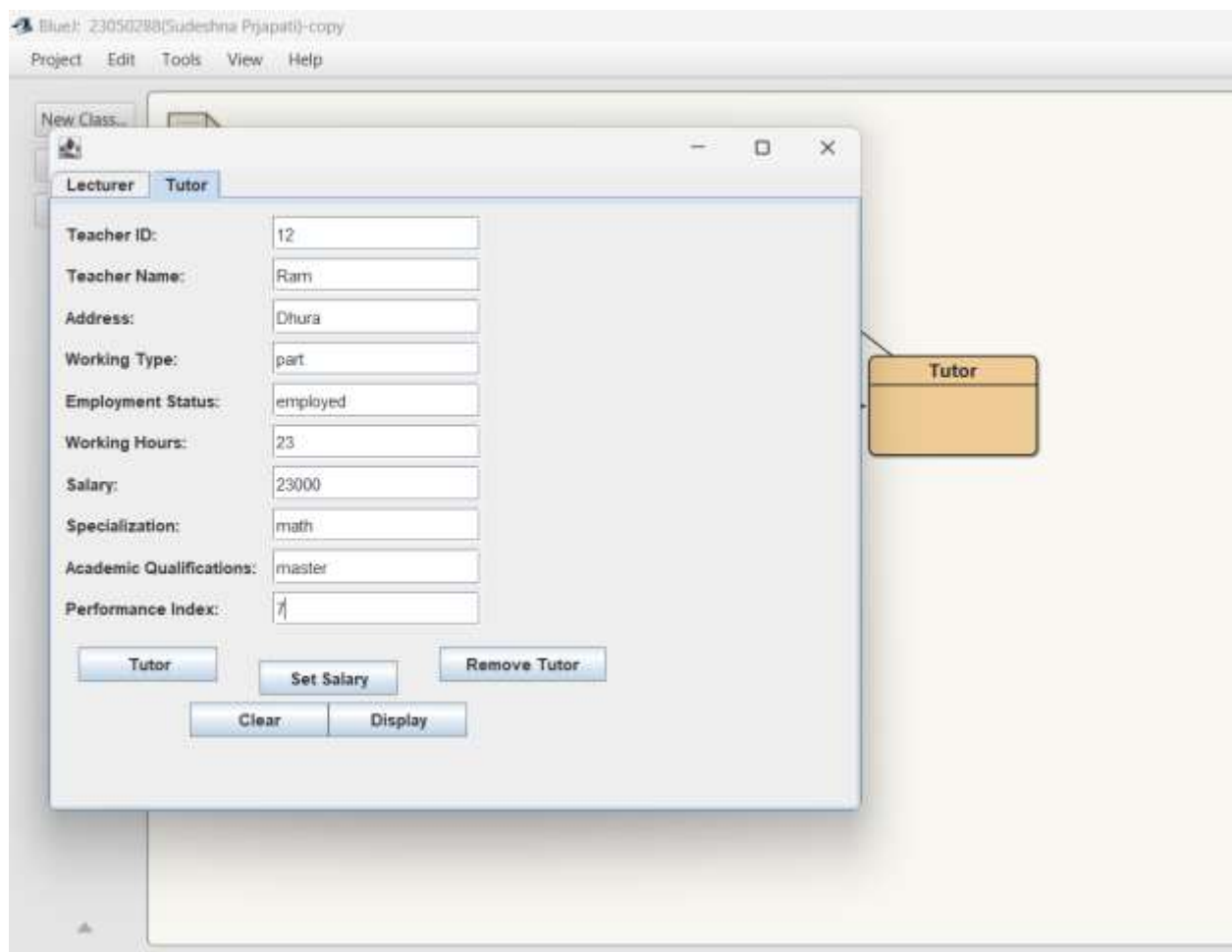


Figure 7: Filling the textfield of Tutor

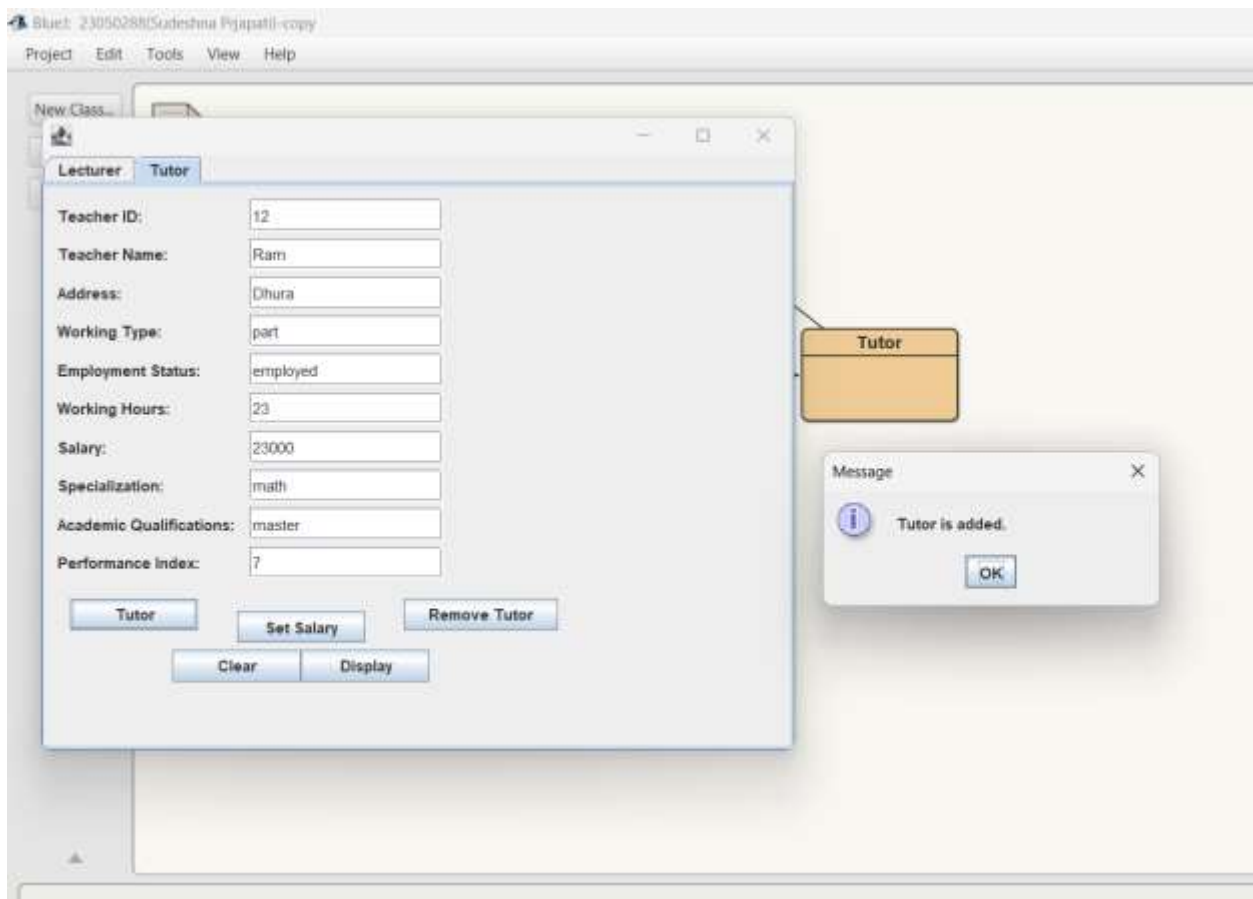


Figure 8: Tutor added successfully

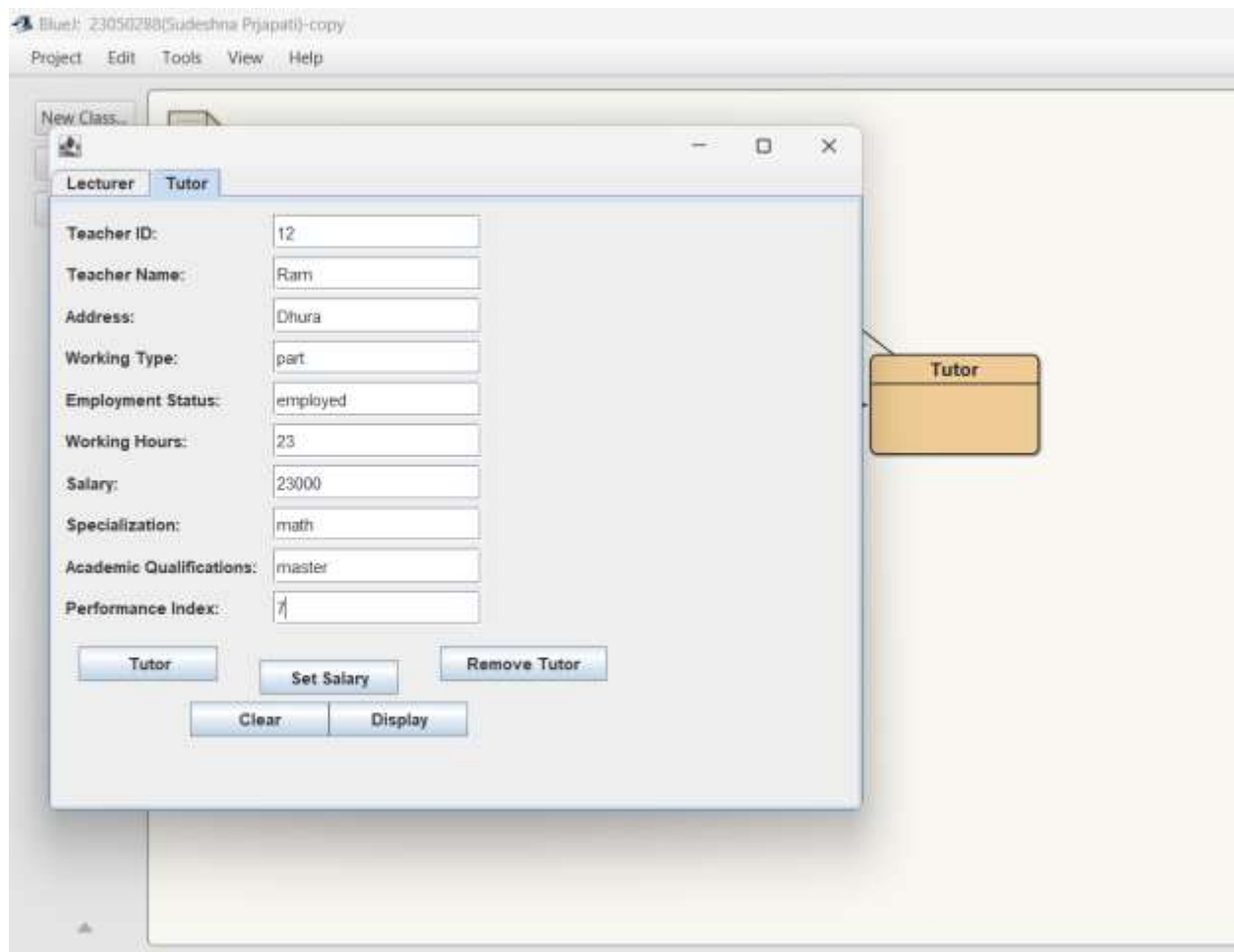


Figure 9: filling the textfield for set salary

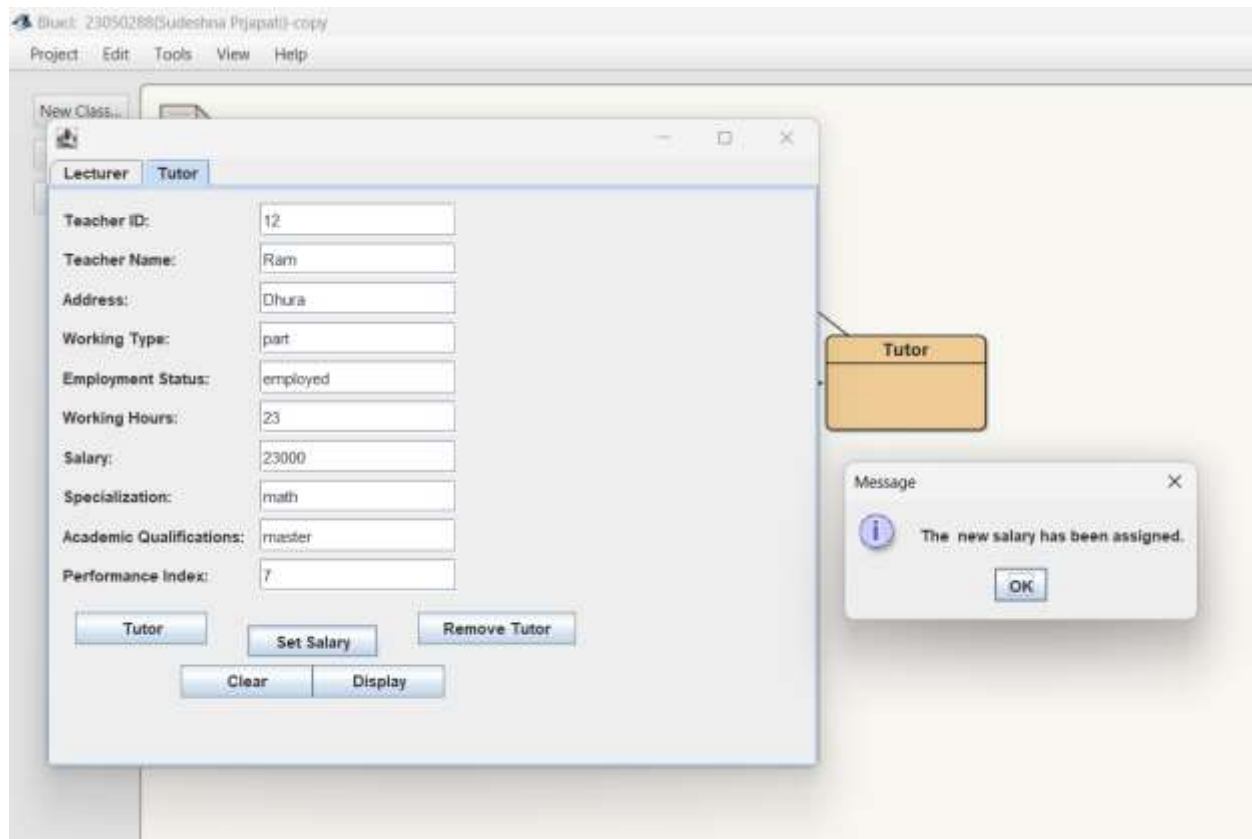


Figure 10: New salary has been assigned

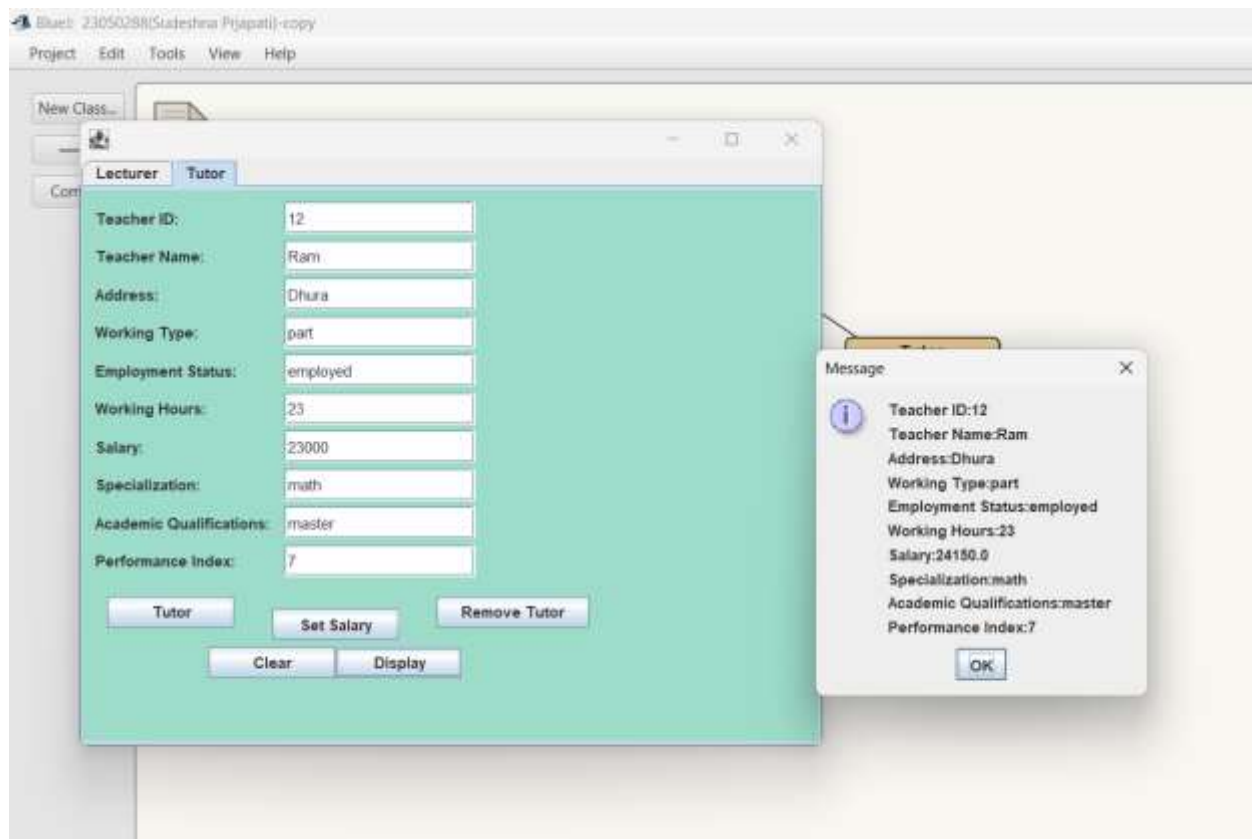


Figure 11: New salary has been updated.

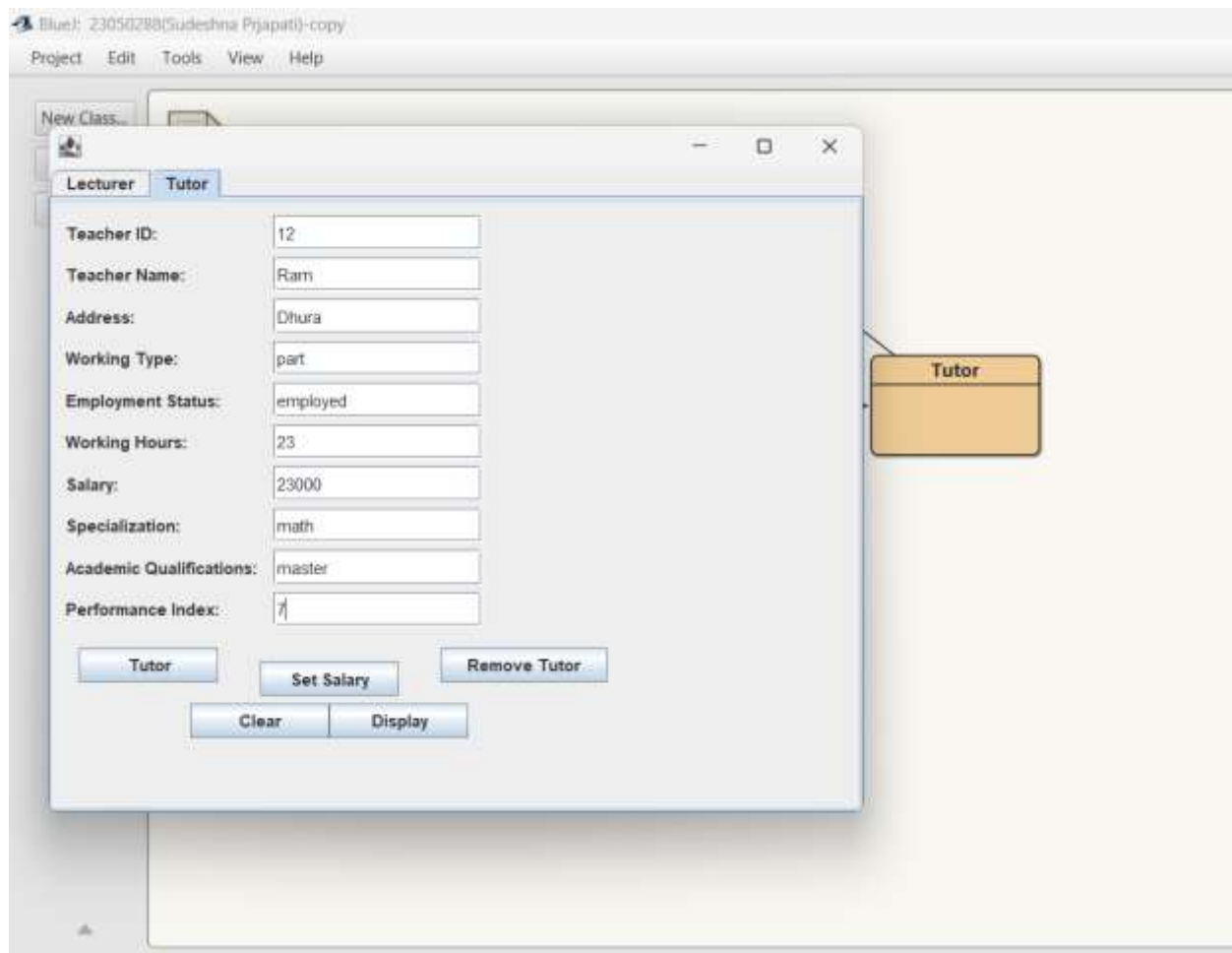


Figure 12: Filling textfield for remove Tutor

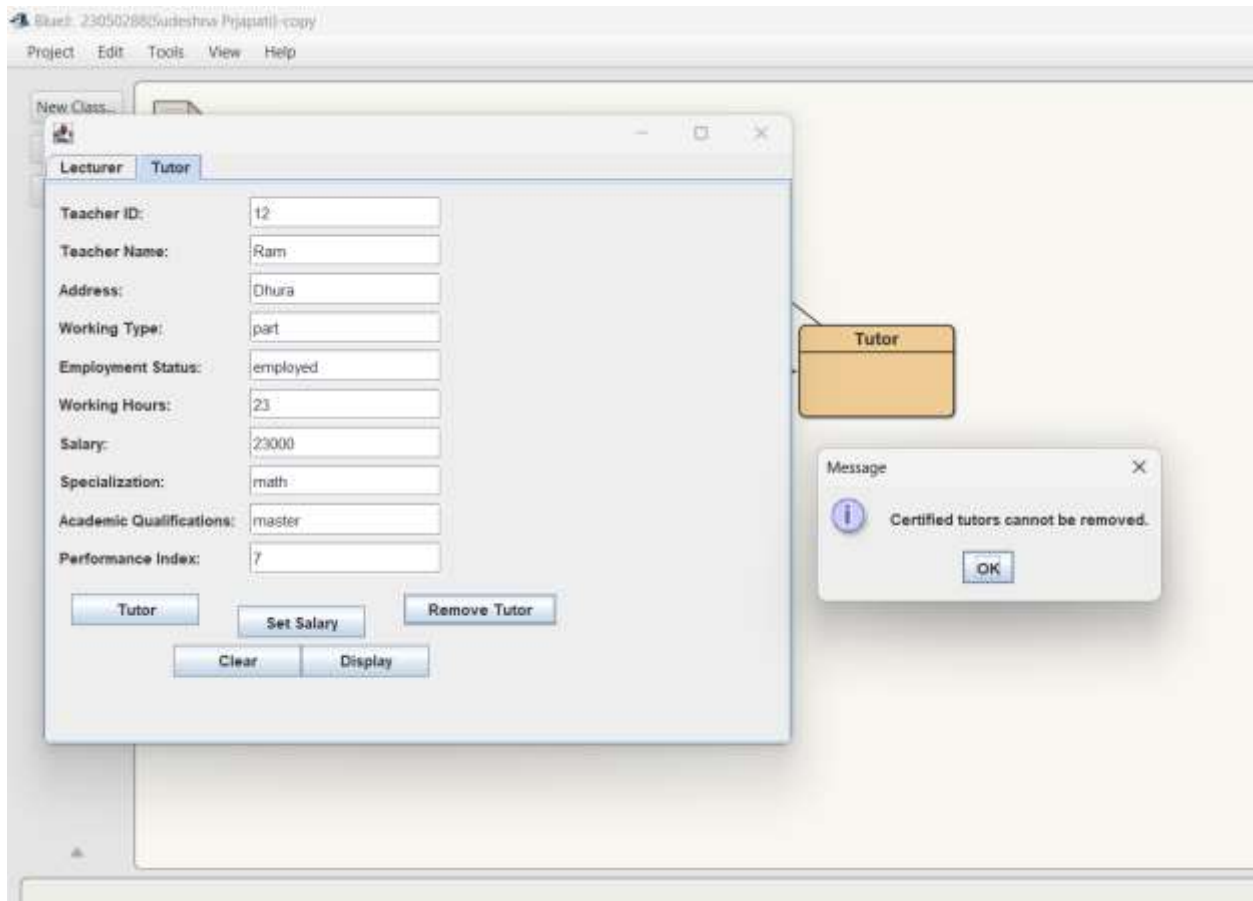


Figure 13: Result of removeTutor

5.3 Test 3: Test that appropriate dialog boxes appear when unsuitable values are entered for the Teacher ID, (include a screenshot of the dialog box, together with a corresponding screenshot of the GUI, showing the values that were entered)

Test No.	3
Objectives	To test that appropriate dialog boxes appear when unsuitable values are entered for the adding Lecturer .
Action	<ul style="list-style-type: none"> -Lecturer button was pressed without entering data. -Lecturer button was pressed with wrong data. -Press display with wrong id - Press lecturer button again after adding it.
Expected Result	Dialog should be shown with its valid problem in the pane.
Actual Result	Dialog box was shown with its valid problem in the pane.
Conclusion	The test was successful.

Table 7: Testing 3

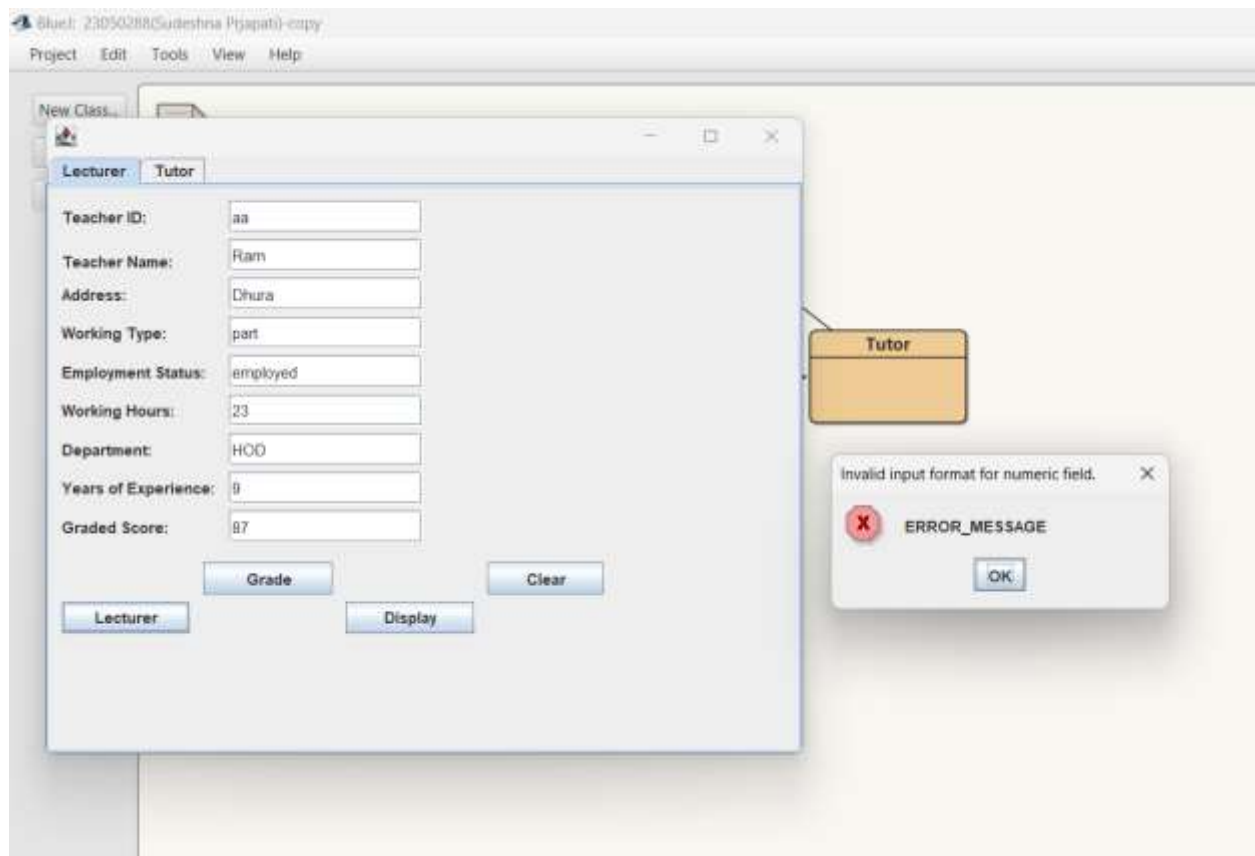


Figure 14:Lecturer button pressed.

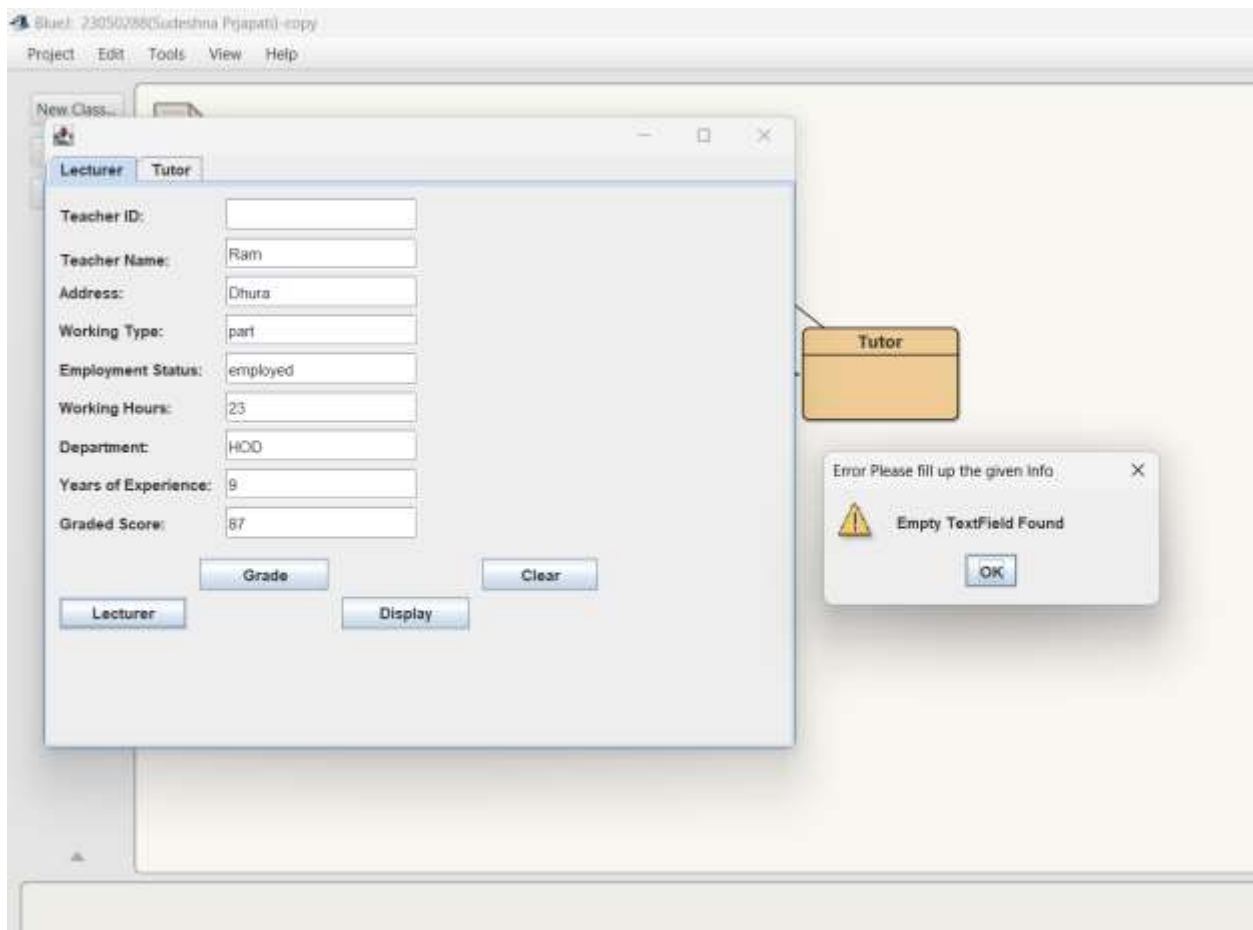


Figure 15: Add lecturer was clicked with unfilled values

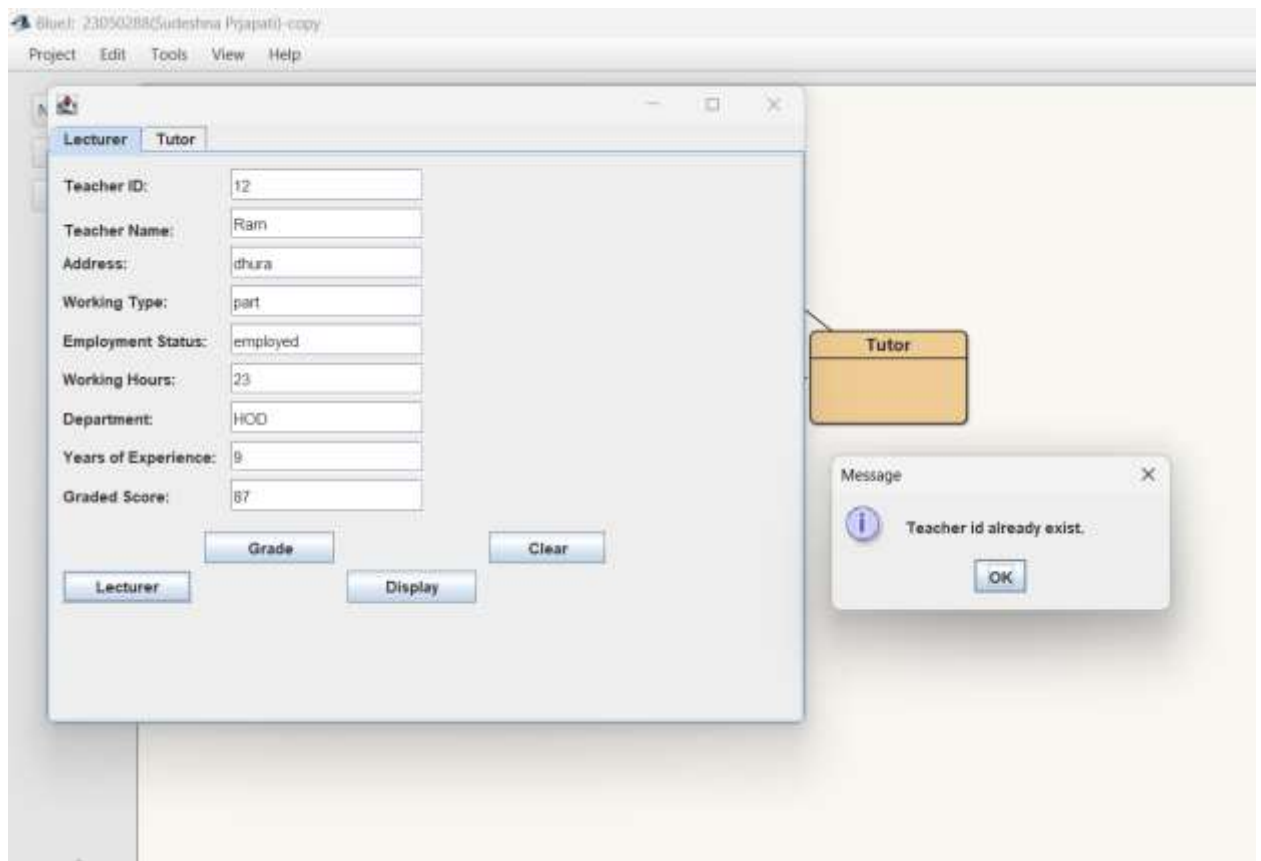


Figure 16: Press add lecturer again after adding

6. Error Detection

6.1 Syntax Error

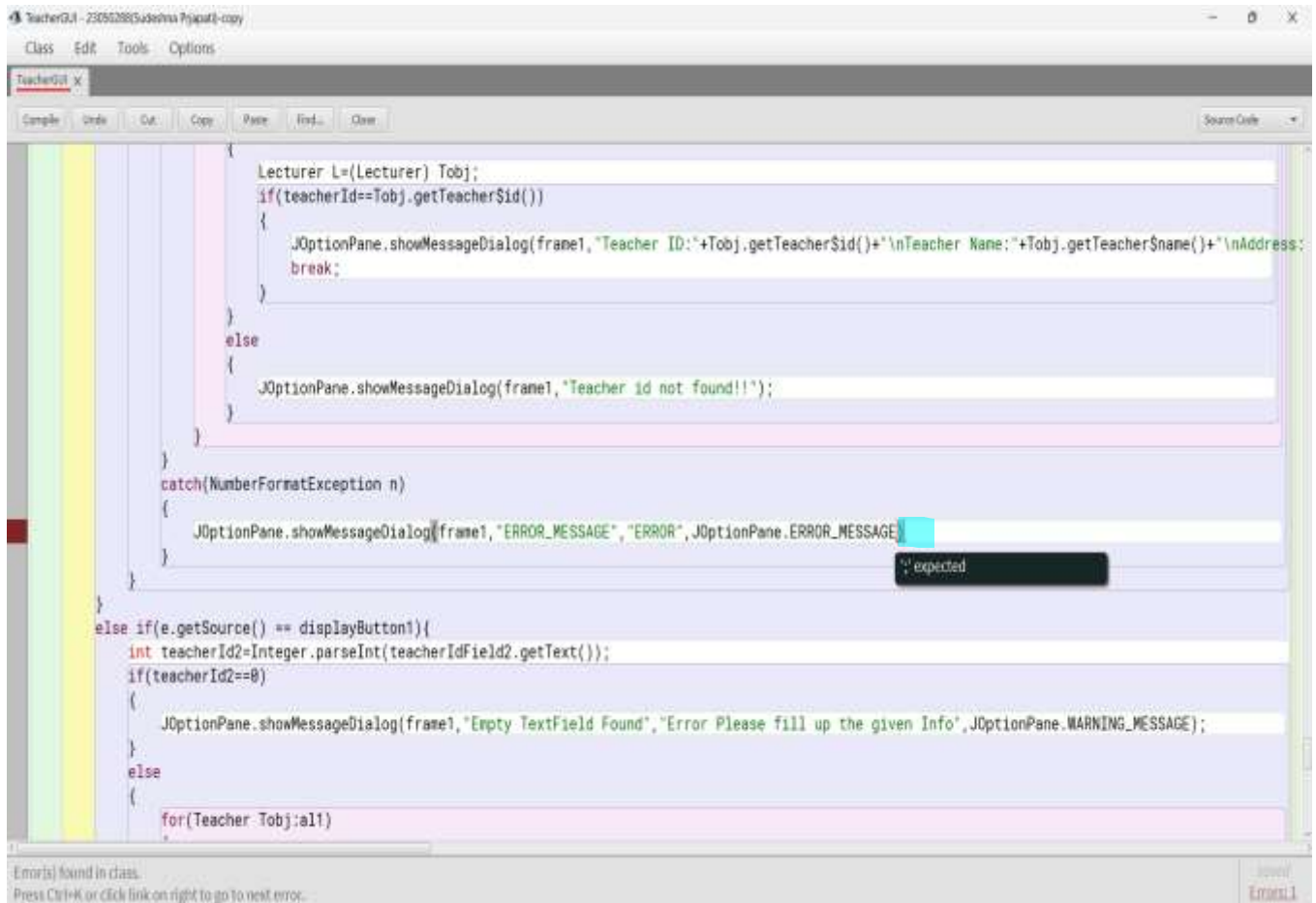


Figure 17: Error of syntax

6.1.1 Syntax Error Detection

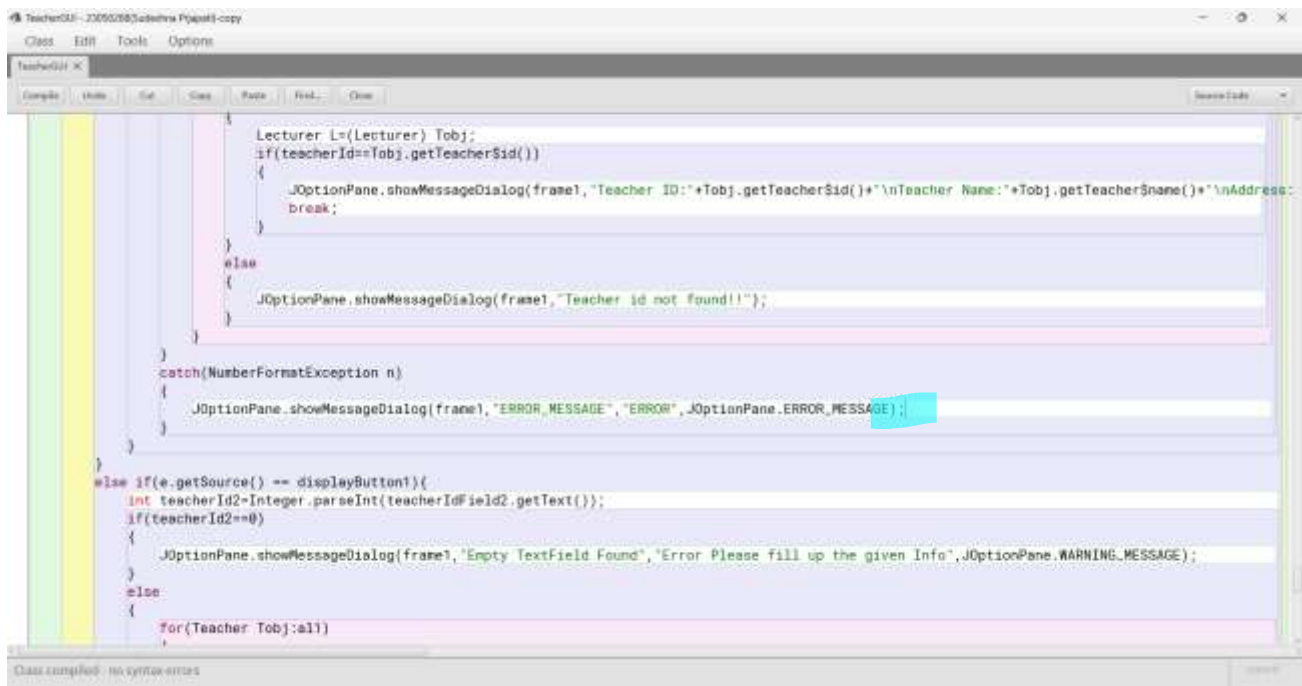
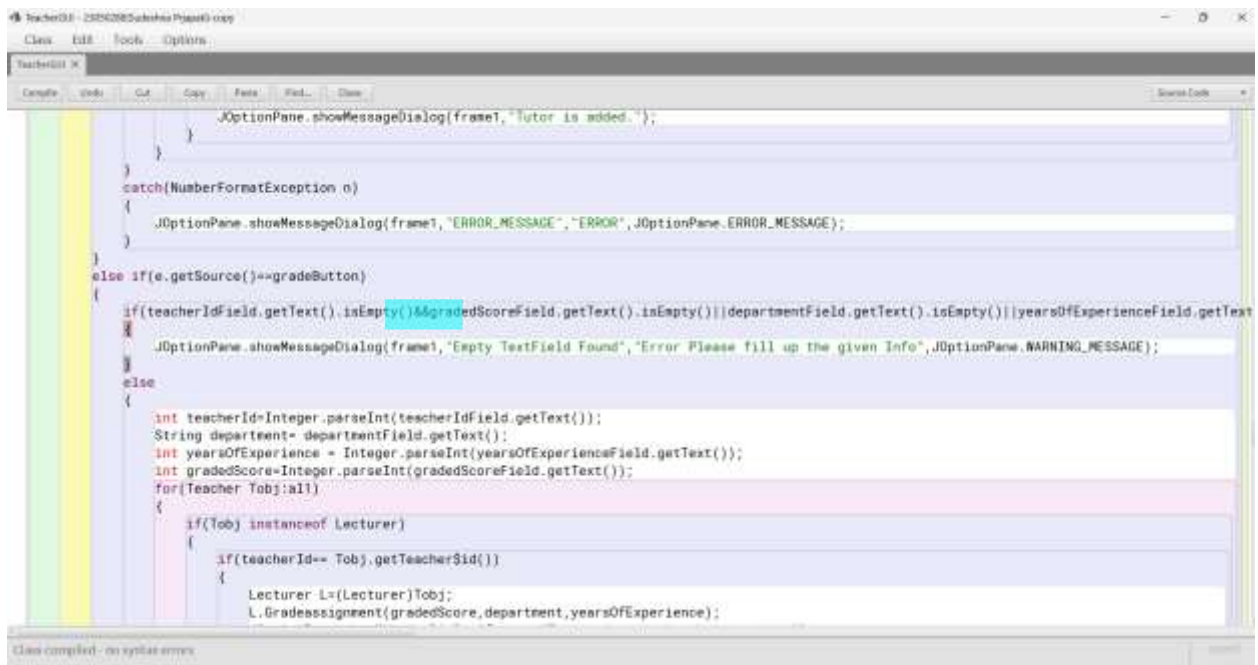


Figure 18: Error Detection of syntax error

The semicolon(;) is an important component in Java syntax as it is used to distinguish between the statements. If the semicolon at the end of the line is missed, it throws a syntax error. While compiling the code, the compiler will throw an error indicating the missing of semicolon preventing the no syntax error.

To run the code or to solve the syntax error, we should add a semicolon at the end of the code to terminate the statement and run.

6.2 Logical Error



```
TeacherGUI X
Compile  Undo  Cut  Copy  Paste  Find...  Close  Search Code

JOptionPane.showMessageDialog(frame1, "Tutor is added.");
}
}
catch(NumberFormatException n)
{
    JOptionPane.showMessageDialog(frame1, "ERROR_MESSAGE", "ERROR", JOptionPane.ERROR_MESSAGE);
}
}
else if(e.getSource()==gradeButton)
{
    if((teacherIdField.getText().isEmpty() && gradedScoreField.getText().isEmpty()) || departmentField.getText().isEmpty() || yearsOfExperienceField.getText().isEmpty())
    {
        JOptionPane.showMessageDialog(frame1, "Empty TextField Found", "Error Please fill up the given Info", JOptionPane.WARNING_MESSAGE);
    }
    else
    {
        int teacherId= Integer.parseInt(teacherIdField.getText());
        String department= departmentField.getText();
        int yearsOfExperience = Integer.parseInt(yearsOfExperienceField.getText());
        int gradedScore= Integer.parseInt(gradedScoreField.getText());
        for(Teacher Tobj:all)
        {
            if(Tobj instanceof Lecturer)
            {
                if(teacherId== Tobj.getTeacherId())
                {
                    Lecturer L=(Lecturer)Tobj;
                    L.Gradeassignment(gradedScore, department, yearsOfExperience);
                }
            }
        }
    }
}
```

Class compiled - no syntax errors

Figure 19: Logical error

6.2.1 Logical Error Detection

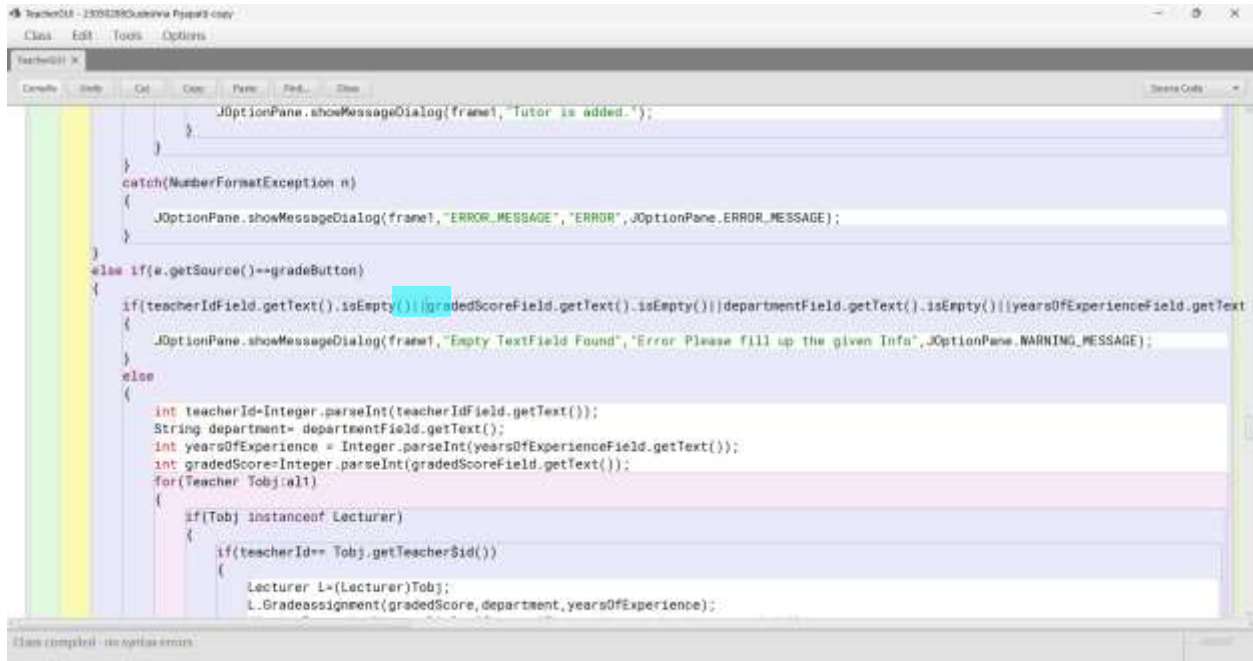


Figure 20: Error detection of logical error

In the above code, the condition(`teacherIdField.getText().isEmpty()&&gradedScoreField.getText().isEmpty()`) the symbol is not correct but the program still runs, this is logical error. To solve the problem the condition must be(`teacherIdField.getText().isEmpty()&&gradedScoreField.getText().isEmpty()`).

7. Conclusion

7.1 Evaluation of the work

I made GUI including all the textfields, buttons, labels, panel with all the given components. Then I created 9 buttons i.e. lecturer, tutor, grade assignment, set salary, remove tutor, both display and clear button for both lecture and tutor panel. I made them work and added the information in all the text fields.

7.2 Reflection on what I learned from the coursework

When I went through my coursework, I learned about the working of GUI, buttons on GUI. I knew a lot more on vast about Java. I learned about the upcasting and downcasting. I got to know more about the command prompt and level up my skills. This coursework helped me know more about the Ms word draw.io, etc.

7.3 Difficulties I encountered

I suffered on my GUI part and more on the buttons parts. I got stuck on testing part. Buttons were very hard to keep in use. While writing reports, the testing part was a little more time taking.

7.4 How I overcame the difficulties

I overcame my difficulties by practicing and researching here and there. I also took help from the teachers guidelines and the slides provides in my second teacher. I took reference from the websites (freecodecamp.org, n.d.).

8. References

freecodecamp.org. (n.d.).

9.Appendix

9.1 Appendix of Teacher class

```
public class Teacher
{
    private String teacher$name;
    private String address;
    private String working$type;
    private String employment$status;
    private int teacher$id;
    private int working$hours;

    public Teacher(int teacher$id,String teacher$name,String address,String
working$type,String employment$status)
    {
        this.teacher$name=teacher$name;
        this.address=address;
        this.working$type=working$type;
        this.employment$status=employment$status;
        this.teacher$id=teacher$id;
    }
}
```

```
public int getTeacher$Id()
{
    return this.teacher$Id;
}

public String getTeacher$name()
{
    return this.teacher$name;
}

public String getAddress()
{
    return this.address;
}

public String getEmployment$status()
{
    return this.employment$status;
}

public String getWorking$type()
{
    return this.working$type;
}
```

```

    }

    public int getWorking$hours()

    {

        return this.working$hours;

    }


    public void setWorking$hours(int working$hours)

    {

        this.working$hours=working$hours;

    }

    public void display()

    {

        System.out.println("the id of the teacher is="+teacher$id);

        System.out.println("the name of the teacher is="+teacher$name);

        System.out.println("the address of the teacher is="+address);

        System.out.println("the working$type of the teacher is="+working$type);

        System.out.println("the    employment$status    of    the    teacher

is="+employment$status);

        if(working$hours ==0)

```

```
{  
    System.out.println("working$hours is not assigned");  
}  
}  
}
```


9.2.Code of Lecturer Class

```
public class Lecturer extends Teacher
{
    private String department;

    private int yearsofexperience;

    private int gradedscore;

    private boolean hasgraded;


    public Lecturer(int teacher$Id,String teacher$name,String address,String
working$type,String employment$status,String department,int yearsofexperience,int
working$hours)
    {
        super( teacher$Id, teacher$name,address, working$type, employment$status);

        this.department=department;

        this.yearsofexperience=yearsofexperience;

        this.gradedscore=0;

        setWorking$hours( working$hours);//a setter method is created

        this.hasgraded=false;

    }

    public String getDepartment()
    {
```

```

        return this.department;
    }

    public int getYearsofexperience()
    {
        return this.yearsofexperience;
    }

    public int getGradedscore()
    {
        return this.gradedscore;
    }

    public boolean getHasgraded()
    {
        return this.hasgraded;
    }

    public void setGradedscore(int gradedscore)
    {
        this.gradedscore=gradedscore;
    }

    public void Gradeassignment(int gradedscore,String department,int
    yearsofexperience)
    {
        if(yearsofexperience>=5 && this.department==department)
        {

```

```
if(gradedscore>=70)
{
    System.out.println("The grade is A");
}
else if(gradedscore>=60 && gradedscore<70)
{
    System.out.println("The grade is B");
}
else if(gradedscore>=50 && gradedscore<60)
{
    System.out.println("The grade is C");
}
else if(gradedscore>=40 && gradedscore<50)
{
    System.out.println("The grade is D");
}
else
{
    System.out.println("The grade is E");
}
hasgraded=true;
}
```

```

else

{
    System.out.println("The mark is not graded yet. ");
}

}

public void display()
{
    super.display();
    if(hasgraded==true)
    {
        System.out.println("Lecturer gradedscore is"+gradedscore );
        System.out.println("Lecturer department is"+department);
        System.out.println("Lecturer yearsofexperience is"+ yearsofexperience);

    }

    else

    {
        System.out.println("The mark is not graded yet");

    }

}

```

}

9.3. Code of the Tutor class

```
public class Tutor extends Teacher
{
    private double salary;

    private String specialization;

    private String academic$qualifications;

    private int performance$index;

    private boolean iscertified;


    public Tutor(int teacher$id,String teacher$name,String address,String
working$type,String employment$status,int working$hours,double salary,String
specialization,String academic$qualifications,int performance$index)
    {
        super( teacher$id, teacher$name,address,working$type, employment$status);

        setWorking$hours( working$hours);

        this.salary=salary;

        this.specialization=specialization;

        this.academic$qualifications=academic$qualifications;

        this.performance$index=performance$index;

        iscertified=false;

    }

    public double getSalary()
```

```

{
    return this.salary;
}

public String getSpecialization()
{
    return this.specialization;
}

public String getAcademic$qualifications()
{
    return this.academic$qualifications;
}

public int getPerformance$index()
{
    return this.performance$index;
}

public boolean getIsCertified()
{
    return this.isCertified;
}

public void setSalary(double salary,int performance$index)
{
    if(performance$index>5 && getWorking$hours()>20)

```

```

{
    if(performance$index>=5 && performance$index<=7)
    {
        this.salary=salary+(0.05d*salary);
    }
    else if(performance$index>=8 && performance$index<=9)
    {
        this.salary=salary+(0.1d*salary);
    }
    else
    {
        this.salary=salary+(0.2d*salary);
    }
    this.iscertified=true;
}
else
{
    System.out.println("The salary is not approved.");
}
}

public void removeTutor()
{

```



```

if(iscertified==false)
{
    this.salary=0;

    this.specialization=null;

    this.academic$qualifications=null;

    this.performance$index=0;

    this.iscertified=false;

    System.out.println("The tutor has not been certified yet.");
}
else
{
    System.out.println("The tutor has been certified .");
}
}

```

```

public void display()
{
    if(iscertified==false)
    {
        super.display();
    }
    else

```

```
{  
    super.display();  
    System.out.println("the salary of the tutor is" +salary);  
    System.out.println("the specialization of the tutor is" +specialization);  
    System.out.println("the academic$qualifications of the tutor is"  
+academic$qualifications);  
    System.out.println("the performance$index of the tutor is" +performance$index);  
}  
  
}  
  
}
```

9.4 Appendix of TeacherGUI class

```
import javax.swing.JFrame;

import javax.swing.JTextField;

import javax.swing.JButton;

import javax.swing.JLabel;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

import javax.swing.JPanel;

import javax.swing.JTabbedPane;

import javax.swing.JOptionPane;

import java.awt.Color;


public class TeacherGUI extends JFrame implements ActionListener

{

    JFrame frame1;

    JLabel

teacherIdLabel,teacherNameLabel,addressLabel,workingTypeLabel,employmentStatus

Label,workingHoursLabel,departmentLabel,yearsOfExperienceLabel,gradedScoreLabel

;

    JLabel

teacherIdLabel2,teacherNameLabel2,addressLabel2,workingTypeLabel2,employmentSt

atusLabel2,workingHoursLabel2,salaryLabel,specializationLabel,academicQualifications

Label,performanceIndexLabel;
```

TextField

teacherIdField,teacherNameField,addressField,workingTypeField,employmentStatusField,departmentField,yearsOfExperienceField,workingHoursField,gradedScoreField;

TextField

teacherIdField2,teacherNameField2,addressField2,workingTypeField2,employmentStatusField2,workingHoursField2,salaryField,specializationField,academicQualificationsField,performanceIndexField;

Button

lecturerButton,gradeButton,displayButton,clearButton,displayButton1,clearButton1,tutorButton,setSalaryButton,removeTutorButton;

ArrayList<Teacher> al1;

public TeacherGUI()

{

al1=new ArrayList<Teacher>();

JTabbedPane tabbedPane = new JTabbedPane();

JPanel lecturerPanel = new JPanel();

lecturerPanel.setBackground(new Color(230, 156, 147));

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

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

```
addressLabel = new JLabel("Address:");  
workingTypeLabel = new JLabel("Working Type:");  
employmentStatusLabel = new JLabel("Employment Status:");  
workingHoursLabel = new JLabel("Working Hours:");  
departmentLabel = new JLabel("Department:");  
yearsOfExperienceLabel = new JLabel("Years of Experience:");  
gradedScoreLabel=new JLabel("Graded Score:");
```

```
JPanel tutorPanel = new JPanel();  
tutorPanel.setBackground(new Color( 158, 220, 202 ));  
teacherIdLabel2 = new JLabel("Teacher ID:");  
JLabel teacherNameLabel2 = new JLabel("Teacher Name:");  
JLabel addressLabel2 = new JLabel("Address:");  
JLabel workingTypeLabel2 = new JLabel("Working Type:");  
JLabel employmentStatusLabel2 = new JLabel("Employment Status:");  
JLabel workingHoursLabel2 = new JLabel("Working Hours:");  
JLabel salaryLabel = new JLabel("Salary:");  
JLabel specializationLabel = new JLabel("Specialization:");  
JLabel academicQualificationsLabel = new JLabel("Academic Qualifications:");  
JLabel performanceIndexLabel = new JLabel("Performance Index:");
```

```
lecturerButton = new JButton("Lecturer");
```

```
gradeButton = new JButton("Grade");
```

```
displayButton = new JButton("Display");
```

```
clearButton = new JButton("Clear");
```

```
tutorButton = new JButton("Tutor");
```

```
setSalaryButton = new JButton("Set Salary");
```

```
removeTutorButton = new JButton("Remove Tutor");
```

```
displayButton1=new JButton("Display");
```

```
clearButton1 = new JButton("Clear");
```

```
teacherIdField = new JTextField();
```

```
teacherNameField = new JTextField();
```

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

```
workingTypeField = new JTextField();
```

```
employmentStatusField = new JTextField();
```

```
workingHoursField = new JTextField();
```

```
departmentField = new JTextField();
```

```
yearsOfExperienceField = new JTextField();
```

```
gradedScoreField=new JTextField();
```

```
//declaring textfields for tutor tab
```

```
teacherIdField2 = new JTextField();
```

```
teacherNameField2 = new JTextField();
```

```
addressField2 = new JTextField();
```

```
workingTypeField2 = new JTextField();
```

```
employmentStatusField2 = new JTextField();
```

```
workingHoursField2 = new JTextField();
```

```
salaryField = new JTextField();
```

```
specializationField = new JTextField();
```

```
academicQualificationsField = new JTextField();
```

```
performanceIndexField = new JTextField();
```

```
teacherIdField.setBounds(140, 10, 150, 25);
```

```
teacherNameField.setBounds(140, 40, 150, 25);
```

```
addressField.setBounds(140, 70, 150, 25);
```

```
workingTypeField.setBounds(140, 100, 150, 25);
```

```
employmentStatusField.setBounds(140, 130, 150, 25);
```

```
workingHoursField.setBounds(140, 160, 150, 25);
```

```
departmentField.setBounds(140, 190, 150, 25);
```

```
yearsOfExperienceField.setBounds(140, 220, 150, 25);  
gradedScoreField.setBounds(140,250,150,25);
```

```
teacherIdField2.setBounds(160, 10, 150, 25);  
teacherNameField2.setBounds(160, 40, 150, 25);  
addressField2.setBounds(160, 70, 150, 25);  
workingTypeField2.setBounds(160, 100, 150, 25);  
employmentStatusField2.setBounds(160, 130, 150, 25);  
workingHoursField2.setBounds(160, 160, 150, 25);  
salaryField.setBounds(160, 190, 150, 25);  
specializationField.setBounds(160, 220, 150, 25);  
academicQualificationsField.setBounds(160, 250, 150, 25);  
performanceIndexField.setBounds(160, 280, 150, 25);
```

```
teacherIdLabel.setBounds(10, 10, 100, 25);  
teacherNameLabel.setBounds(10, 45, 100, 25);  
addressLabel.setBounds(10, 70, 100, 25);  
workingTypeLabel.setBounds(10, 100, 100, 25);  
employmentStatusLabel.setBounds(10, 130, 120, 25);  
workingHoursLabel.setBounds(10, 160, 100, 25);
```



```
departmentLabel.setBounds(10, 190, 100, 25);  
yearsOfExperienceLabel.setBounds(10, 220, 130, 25);  
gradedScoreLabel.setBounds(10,250,100,25);
```

```
teacherIdLabel2.setBounds(10, 10, 100, 25);  
teacherNameLabel2.setBounds(10, 40, 100, 25);  
addressLabel2.setBounds(10, 70, 100, 25);  
workingTypeLabel2.setBounds(10, 100, 100, 25);  
employmentStatusLabel2.setBounds(10, 130, 120, 25);  
workingHoursLabel2.setBounds(10, 160, 100, 25);  
salaryLabel.setBounds(10, 190, 100, 25);  
specializationLabel.setBounds(10, 220, 100, 25);  
academicQualificationsLabel.setBounds(10, 250, 150, 25);  
performanceIndexLabel.setBounds(10, 280, 120, 25);
```

```
lecturerButton.setBounds(10, 320, 100, 25);  
gradeButton.setBounds(120, 290, 100, 25);  
displayButton.setBounds(230, 320, 100, 25);  
clearButton.setBounds(340, 290, 90, 25);
```

```
tutorButton.setBounds(20, 320, 100, 25);  
setSalaryButton.setBounds(150, 330, 100, 25);  
removeTutorButton.setBounds(280, 320, 120, 25);  
displayButton1.setBounds(200,360,100,25);  
clearButton1.setBounds(100,360,100,25);
```

```
lecturerPanel.add(teacherIdLabel);  
lecturerPanel.add(teacherNameLabel);  
lecturerPanel.add(addressLabel);  
lecturerPanel.add(workingTypeLabel);  
lecturerPanel.add(employmentStatusLabel);  
lecturerPanel.add(workingHoursLabel);  
lecturerPanel.add(departmentLabel);  
lecturerPanel.add(yearsOfExperienceLabel);  
lecturerPanel.add(gradedScoreLabel);
```

```
tutorPanel.add(teacherIdLabel2);  
tutorPanel.add(teacherNameLabel2);
```

```
tutorPanel.add(addressLabel2);  
tutorPanel.add(workingTypeLabel2);  
tutorPanel.add(employmentStatusLabel2);  
tutorPanel.add(salaryLabel);  
tutorPanel.add(workingHoursLabel2);  
tutorPanel.add(specializationLabel);  
tutorPanel.add(academicQualificationsLabel);  
tutorPanel.add(performanceIndexLabel);
```

```
lecturerPanel.add(teacherIdField);  
lecturerPanel.add(teacherNameField);  
lecturerPanel.add(addressField);  
lecturerPanel.add(workingTypeField);  
lecturerPanel.add(employmentStatusField);  
lecturerPanel.add(workingHoursField);  
lecturerPanel.add(departmentField);  
lecturerPanel.add(yearsOfExperienceField);  
lecturerPanel.add(gradedScoreField);
```

```
tutorPanel.add(teacherIdField2);  
tutorPanel.add(teacherNameField2);  
tutorPanel.add(addressField2);
```

```
tutorPanel.add(workingTypeField2);  
tutorPanel.add(employmentStatusField2);  
tutorPanel.add(workingHoursField2);  
tutorPanel.add(salaryField);  
tutorPanel.add(specializationField);  
tutorPanel.add(academicQualificationsField);  
tutorPanel.add(performanceIndexField);
```

```
lecturerPanel.add(lecturerButton);  
lecturerPanel.add(gradeButton);  
lecturerPanel.add(displayButton);  
lecturerPanel.add(clearButton);  
tabbedPane.addTab("Lecturer", lecturerPanel);
```

```
tutorPanel.add(tutorButton);  
tutorPanel.add(setSalaryButton);  
tutorPanel.add(removeTutorButton);  
tutorPanel.add(displayButton1);  
tutorPanel.add(clearButton1);  
tabbedPane.addTab("Tutor", tutorPanel);  
add(tabbedPane);//adding JTabbedPane to JFrame
```

```

    lecturerButton.addActionListener(this);

    tutorButton.addActionListener(this);

    gradeButton.addActionListener(this);

    setSalaryButton.addActionListener(this);

    removeTutorButton.addActionListener(this);

    displayButton.addActionListener(this);

    clearButton.addActionListener(this);

    clearButton1.addActionListener(this);

    displayButton1.addActionListener(this);


    tutorPanel.setLayout(null);

    lecturerPanel.setLayout(null);

    setVisible(true);

    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    setSize(600,500);

}


@Override

public void actionPerformed(ActionEvent e)

```

```

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

        if(teacherIdField.getText().isEmpty()||teacherNameField.getText().isEmpty()||addressField.getText().isEmpty()||workingTypeField.getText().isEmpty()||employmentStatusField.getText().isEmpty()||workingHoursField.getText().isEmpty()||departmentField.getText().isEmpty()||yearsOfExperienceField.getText().isEmpty())
        {

            JOptionPane.showMessageDialog(frame1,"Empty TextField Found","Error Please fill up the given Info",JOptionPane.WARNING_MESSAGE);

        }
        else
        {
            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();
            }
            catch (Exception e)
            {
                JOptionPane.showMessageDialog(frame1,"Invalid Input","Error Please fill up the given Info",JOptionPane.WARNING_MESSAGE);
            }
        }
    }
}

```

```

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

        Lecturer lecturerobj= new Lecturer(teacherId,teacherName,
        address,workingType,employmentStatus,department,yearsOfExperience
        ,workingHours);

        if(al1.isEmpty())

        {

            al1.add(lecturerobj);

            JOptionPane.showMessageDialog(frame1,"Lecturer is added.");

        }

        else

        {

            boolean add= true;

            for (Teacher Tobj:al1)

            {

                if(Tobj instanceof Lecturer)

                {

                    if(teacherId==Tobj.getTeacher$id())

                    {

                        JOptionPane.showMessageDialog(frame1,"This id has already
                        been used.");

                        add = false;

                        break;

                    }

                }

            }

        }

```

```

        }
    }
    if (add==true)
    {
        al1.add(lecturerobj);

        JOptionPane.showMessageDialog(frame1,"The id is added.");
    }
    else
    {
        JOptionPane.showMessageDialog(frame1,"Teacher id already exist.");

    }
}

catch(NumberFormatException n)
{

    JOptionPane.showMessageDialog(frame1,"ERROR_MESSAGE","Invalid
input format for numeric field.", JOptionPane.ERROR_MESSAGE);

}

}

}
else if(e.getSource()==tutorButton)

```



```

{
    try{
        int teacherId = Integer.parseInt(teacherIdField2.getText());

        String teacherName = teacherNameField2.getText();

        String address = addressField2.getText();

        String workingType = workingTypeField2.getText();

        String employmentStatus = employmentStatusField2.getText();

        int workingHours = Integer.parseInt(workingHoursField2.getText());

        double salary = Double.parseDouble(salaryField.getText());

        String specialization = specializationField.getText();

        String academicQualifications = academicQualificationsField.getText();

        int performanceIndex = Integer.parseInt(performanceIndexField.getText())

        Tutor Tutorobj = new Tutor(teacherId, teacherName,
        address, workingType, employmentStatus, workingHours, salary, specialization, academicQualifications, performanceIndex);

        if(al1.isEmpty())
        {
            al1.add(Tutorobj);

            JOptionPane.showMessageDialog(frame1, "Tutor is added.");
        }
        else
        {
            boolean add = true;

```

```

for (Teacher Tobj:al1)
{
    if(Tobj instanceof Tutor)
    {
        if(teacherId==Tobj.getTeacher$id())
        {
            add = false;

            JOptionPane.showMessageDialog(frame1,"This id is already
            used.");
            break;
        }
    }
}

if (add==true)
{
    al1.add(Tutorobj);

    JOptionPane.showMessageDialog(frame1,"Tutor is added.");
}

}

catch(NumberFormatException n)
{

```

```

JOptionPane.showMessageDialog(frame1,"ERROR_MESSAGE","ERROR",JOpt
ionPane.ERROR_MESSAGE);

    }
}
else if(e.getSource()==gradeButton)
{

    if(teacherIdField.getText().isEmpty()||gradedScoreField.getText().isEmpty()||depa
rtmentField.getText().isEmpty()||yearsOfExperienceField.getText().isEmpty())
    {

        JOptionPane.showMessageDialog(frame1,"Empty TextField Found","Error
Please fill up the given Info",JOptionPane.WARNING_MESSAGE);

    }
else
{

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

    String department= departmentField.getText();

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

    int gradedScore=Integer.parseInt(gradedScoreField.getText());

    for(Teacher Tobj:al1)
    {

        if(Tobj instanceof Lecturer)
        {

```

```

if(teacherId== Tobj.getTeacher$id())
{
    Lecturer L=(Lecturer)Tobj;
    L.Gradeassignment(gradedScore,department,yearsOfExperience);
    JOptionPane.showMessageDialog(frame1,"The assignment has
        been graded.");
    break;
}
else
{
    JOptionPane.showMessageDialog(null,"Teacher id is not a tutor.");
}
}
else
{
    JOptionPane.showMessageDialog(null,"Teacher id is not found.");
}
}
}

else if(e.getSource()==setSalaryButton)
{

```

```

if(teacherIdField2.getText().isEmpty()||salaryField.getText().isEmpty()||performanceIndexField.getText().isEmpty())

{

    JOptionPane.showMessageDialog(frame1,"Empty TextField Found","Error
Please fill up the given Info",JOptionPane.WARNING_MESSAGE);

}

else

{

    int teacherId2=Integer.parseInt(teacherIdField2.getText());

    double salary=Double.parseDouble(salaryField.getText());

    int performanceIndex=Integer.parseInt(performanceIndexField.getText());

    for (Teacher Tobj:a1)

    {

        if( teacherId2==Tobj.getTeacher$id())

        {

            if(Tobj instanceof Tutor)

            {

                Tutor T=(Tutor)Tobj;

                T.setSalary(salary,performanceIndex);

                JOptionPane.showMessageDialog(frame1,"The new salary has
                been assigned.");

                break;

            }

        }

    }

}

```

```

        else
        {
            JOptionPane.showMessageDialog(null,"Teacher id is not a tutor.");
        }
    }
    else
    {
        JOptionPane.showMessageDialog(null,"Teacher id is not found.");
    }
}

}

}

else if(e.getSource()==removeTutorButton)
{
    int teacherId2=Integer.parseInt(teacherIdField2.getText());

    if(teacherId2==0)
    {
        JOptionPane.showMessageDialog(frame1,"Empty TextField Found","Error
Please fill up the given Info",JOptionPane.WARNING_MESSAGE);
    }

    else
    {
        Teacher tutorToRemove = null;

```

```

boolean teacherfound = false;

for(Teacher Tobj:al1)
{
    if(Tobj.getTeacher$Id()==teacherId2){
        if(Tobj instanceof Tutor)
        {
            Tutor tutor=(Tutor) Tobj;

            if(tutor.getIsCertified())
            {

                JOptionPane.showMessageDialog(null,"Certified tutors cannot be
                removed.");

                return;
            }
            else
            {
                tutorToRemove = tutor;

                teacherfound = true;

                break;
            }
        }
    }
    else
    {

```

```

        JOptionPane.showMessageDialog(null,"Teacher id is not a tutor.");
    }
}
}
if(teacherfound)
{
    ((Tutor) tutorToRemove).removeTutor();
    al1.remove(tutorToRemove);
    JOptionPane.showMessageDialog(null,"Tutor has been removed.");
}
else
{
    JOptionPane.showMessageDialog(null,"Teacher id is not found!!");
}
}
}
else if(e.getSource()==displayButton)
{
    if(teacherIdField.getText().isEmpty())
    {
        JOptionPane.showMessageDialog(frame1,"Empty TextField Found","Error
Please fill up the given Info",JOptionPane.WARNING_MESSAGE);
    }
}

```



```

else
{
    try
    {
        int teacherId = Integer.parseInt(teacherIdField.getText());

        if (teacherId==0)
        {
            JOptionPane.showMessageDialog(frame1,"Empty TextField
Found","Error Please fill up the given Info",JOptionPane.WARNING_MESSAGE);
        }

        for(Teacher Tobj:al1)
        {
            if(Tobj instanceof Lecturer)
            {
                Lecturer L=(Lecturer) Tobj;

                if(teacherId==Tobj.getTeacher$id())
                {

                    JOptionPane.showMessageDialog(frame1,"Teacher
ID:"+Tobj.getTeacher$id()+"\nTeacher
Name:"+Tobj.getTeacher$name()+"\nAddress:"+Tobj.getAddress()
+"\nWorking Type:"+Tobj.getWorking$hours()+"\nEmployment
Status:"+Tobj.getEmployment$status()+"\nWorking
Hours:"+Tobj.getWorking$hours()+"\nDepartment:"+L.getDepartme

```

```

        nt()+"\nYears of Experience:"+L.getYearsofexperience()+"\nGraded
        Score:"+L.getGradedscore());

        break;

    }

}

else

{

    JOptionPane.showMessageDialog(frame1,"Teacher id not found!!");

}

}

}

catch(NumberFormatException n)

{

    JOptionPane.showMessageDialog(frame1,"ERROR_MESSAGE","ERRO
    R",JOptionPane.ERROR_MESSAGE);

}

}

}

else if(e.getSource() == displayButton1){

    int teacherId2=Integer.parseInt(teacherIdField2.getText());

    if(teacherId2==0)

    {

```

```

Found","Error Please fill up the given
Info",JOptionPane.WARNING_MESSAGE);

}

else

{

    for(Teacher Tobj:al1)

    {

        if(Tobj instanceof Tutor)

        {

            Tutor T=(Tutor) Tobj;

            if(teacherId2==Tobj.getTeacher$Id())

            {

                JOptionPane.showMessageDialog(frame1,"Teacher
                ID:"+Tobj.getTeacher$Id()+"\nTeacher
                Name:"+Tobj.getTeacher$name()+"\nAddress:"+Tobj.getAddress()
                +"\nWorking Type:"+Tobj.getWorking$hours()+"\nEmployment
                Status:"+Tobj.getEmployment$status()+"\nWorking
                Hours:"+Tobj.getWorking$hours()+"\nSalary:"+T.getSalary()+"\nSp
                ecialization:"+T.getSpecialization()+"\nAcademic
                Qualifications:"+T.getAcademic$qualifications()+"\nPerformance
                Index:"+T.getPerformance$index());

                break;

            }

        }

    }

```

```

        }
    }
}

else if(e.getSource()==clearButton)
{
    teacherIdField.setText("");
    teacherNameField.setText("");
    addressField.setText("");
    workingTypeField.setText("");
    employmentStatusField.setText("");
    departmentField.setText("");
    yearsOfExperienceField.setText("");
    workingHoursField.setText("");
    gradedScoreField.setText("");
}

else if(e.getSource()==clearButton1)
{
    teacherIdField2.setText("");
    teacherNameField2.setText("");
    addressField2.setText("");
    workingTypeField2.setText("");
    employmentStatusField2.setText("");
}

```

```
        workingHoursField2.setText("");  
        salaryField.setText("");  
        specializationField.setText("");  
        academicQualificationsField.setText("");  
        performanceIndexField.setText("");  
    }  
}  
  
public static void main(String[] args) {  
    new TeacherGUI();  
}  
}
```