



**Islington college**  
(इसलिंग्टन कलेज)

## **CS4001NI Programming**

### **30% Individual Coursework**

**2023-24 Autumn**

**Student Name: Rupesh Kumar Mahato**

**London Met ID: 23047358**

**College ID: NP01NT4A230053**

**Group: N3**

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

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

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

## Contents

<b>1. INTRODUCTION</b>	1
1.1 About the project	1
1.2 Aims and objective	1
1.3.Tools used	2
<b>2.class diagram</b>	3
<b>3.Pseudocode</b>	4
<b>4.Method description</b>	16
<b>5. Testing</b>	17
<b>5.Error detection and correction</b>	22
<b>6.Conclusion</b>	25
Bibliography	26
Appendix	27

## Table of tables

Table 1: Test 1 .....	17
Table 2: test2 .....	18
Table 3:Test3 .....	21

## Table of figures

Figure 1:compile and run using command prompt .....	17
Figure 2:Added lecturer .....	18
Figure 3: output after adding lecturer .....	18
Figure 4:output after adding tutor .....	19
Figure 5:Adding tutor.....	19
Figure 6:Input for grade assignment .....	19
Figure 7: Lecturer was graded.....	19
Figure 8: output after lecturer is graded .....	19
Figure 9: output after setting salary .....	20
Figure 10: set salary.....	20
Figure 11: Dialog box test .....	21
Figure 12>Error detection 1(missing paranthesis).....	22
Figure 13: Correction of error 1 .....	23
Figure 14: Error 2 (missing semicolon).....	23
Figure 15: Correction of error 2 .....	23

## **Acknowledgement**

I want to express my heartfelt gratitude to all those who played a part in the completion of this report. I extend my deepest appreciation to Mr. Ujwal Subedi and Mrs. Astha Sharma for their invaluable assistance, guidance, and unwavering support throughout the entire process.

A special mention goes to Mrs. Astha Sharma for his insightful feedback and encouragement, which greatly enhanced the quality of this work. Furthermore, I would like to acknowledge Islington College for generously providing resources and facilities essential for conducting the research. This report would not have come to fruition without the contributions of each and every individual mentioned above. Thank you from the bottom of my heart

## 1. INTRODUCTION

Java, a class-based, object-oriented programming language, was developed by James Gosling at Sun Microsystems in 1995, later acquired by Oracle. It prioritizes simplicity, robustness, and security. Java's "write once, run anywhere" (WORA) principle allows compiled code to run on any platform supporting Java. This versatility makes it widely used for desktop, web, and mobile applications. (geeksforgeeks, 2024)

### 1.1 About the project

This project constitutes 30% of the total weightage for the course and will require a comprehensive understanding of Java GUI development, event handling, and object-oriented programming concepts. The coursework project involves developing a Graphical User Interface (GUI) for managing teachers' details using Java programming language. The GUI will provide a user-friendly interface for inputting, viewing, and managing information about teachers. The main features of the GUI will include text fields, buttons, and various components to interact with the data. Each button will have its own functionality, such as adding a new teacher, deleting a teacher, displaying teachers information, grading teachers, etc based on their criteria. Additionally, the GUI will incorporate validation mechanisms to ensure data integrity and error handling for user input.

### 1.2 Aims and objective

In this project, the goal is to integrate a class into the existing project developed for the first part of the coursework. This class should create a graphical user interface (GUI) for a system that manages information about teachers and their details, storing them in an ArrayList. The class should contain the main method and should be tested using the command prompt. Furthermore, a report about the program should be written.

The main objectives of the courses are as follows:

- Ensure that the system is easy to navigate, with clear instructions and error handling.
- Create an intuitive graphical interface for managing teachers and their information efficiently.
- Allow users to add new teachers (Lecturer or Tutor) and remove existing ones easily.
- Provide functionality to grade assignments for Lecturers and set salary for Tutors accurately.
- Enable users to view detailed information about selected teachers with simple click.

### 1.3.Tools used

#### ➤ BlueJ

BlueJ was designed for teaching object-oriented programming, providing visual representations of Java code. It offers a multi-view interface, allowing users to see both visual objects and source code. Users can create projects, add classes, compile code, and use tools like a code pad and regression testing. BlueJ supports multiple languages for a global user base. And I have done all the coding of my project in bluej. (Rouse, 2013)

#### ➤ MS Word

Microsoft Word is a leading word processing software known for its user-friendly interface, formatting options, and collaboration features. It's part of the Microsoft Office suite and offers templates, cloud integration, and compatibility with various file formats. Word is widely used for creating documents ranging from simple letters to complex reports, both professionally and personally.

#### ➤ draw.io

draw.io : Draw.io is a free web-based tool for creating various types of diagrams such as flowcharts, diagrams, mind maps, and organization charts.

It's seamlessly integrated with Google Drive, enabling automatic saving of your work in your Google Workspace or Gmail account. (Paraschiv, 2023)

## 2.class diagram

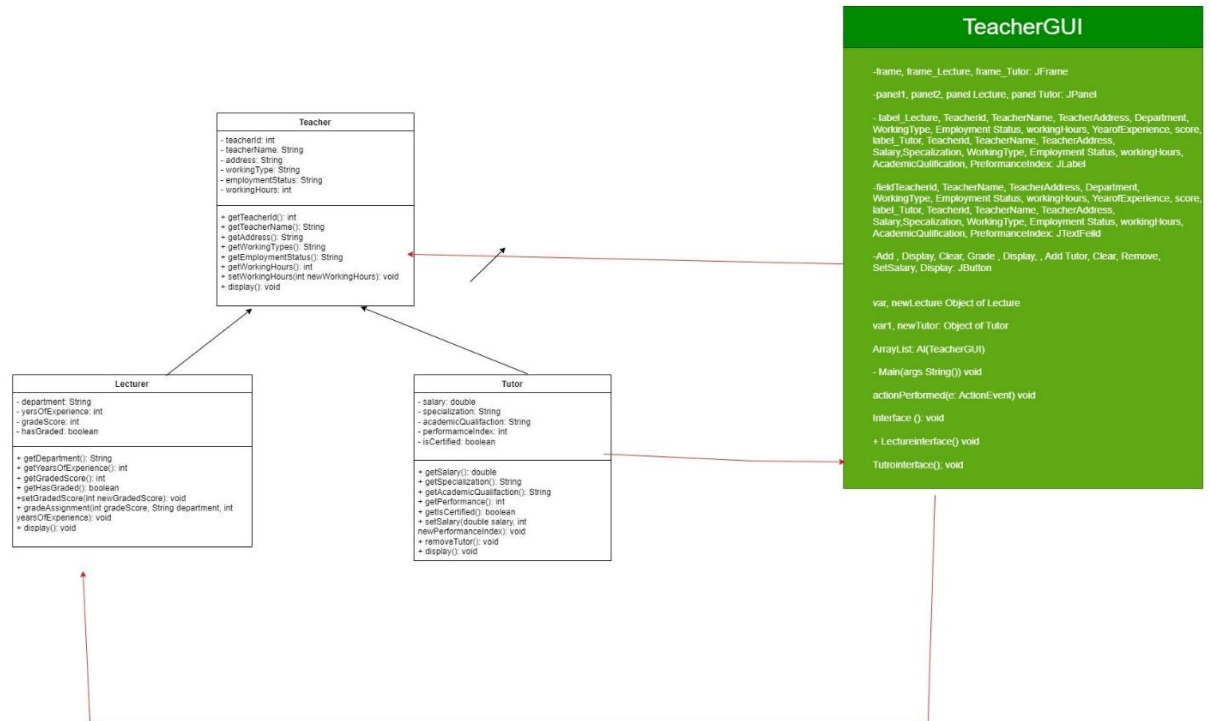


Figure 1: Combined classs diagram

### 3.Pseudocode

Pseudocode is like a language used in data science or web development to break down the steps of an algorithm in a way that's easy for people with basic programming knowledge to follow. It's a simplified version of code, written in plain language, making it accessible to a wide audience. (Urwin, 2024)

**START**

**DECLARE** a class TeacherGUI

**CREATE** a main method

**CREATE** an ArrayList tlist of Teacher objects

**CREATE** a JFrame jf

**SET** jf's size to 1300x1000

**SET** jf's layout to null

**SET** jf's visibility to true

**CREATE** a JPanel p1

**SET** p1's bounds to (0,0,1300,315)

**SET** p1's background color to LIGHT\_GRAY

**SET** p1's layout to null

**ADD** p1 to jf

**CREATE** a JLabel heading and set its text to "LECTURER"

**SET** heading's font to Times new Roman, plain, size 35



**SET** heading's bounds to (475,5,230,30)

**DECLARE** a JLabel l1 for Teacher ID

**SET** l1's font to Times new Roman, plain, size 16

**SET** l1's bounds to (10,60,115,30)

**DECLARE** a JTextField t1 for Teacher ID

**SET** t1's bounds to (138,65,130,25)

**DECLARE** a JLabel l2 for Teacher Name

**SET** l2's font to Times new Roman, plain, size 16

**SET** l2's bounds to (10,60,115,30)

**DECLARE** a JTextField t2for Teacher ID

**SET** t2's bounds to (138,65,130,25)

**DECLARE** a JLabel l3 for address

**SET** l3's font to Times new Roman, plain, size 16

**SET** l3's bounds to (10,60,115,30)

**DECLARE** a JTextField t3 for Teacher ID

**SET** t3's bounds to (138,65,130,25)

**DECLARE** a JLabel l4 for working type

**SET** l4's font to Times new Roman, plain, size 16

**SET** l4s bounds to (10,60,115,30)

**DECLARE** a JTextField t4 for Teacher ID

**SET** t4s bounds to (138,65,130,25)

**DECLARE** a JLabel l5 for Employment Status

**SET** l5's font to Times new Roman, plain, size 16

**SET** l5's bounds to (10,60,115,30)

**DECLARE** a JTextField t1 for Teacher ID

**DECLARE** a JLabel l6 for Working Hours

**SET** l6's font to Times new Roman, plain, size 16

**SET** l6's bounds to (10,60,115,30)

**DECLARE** a JTextField t6 for Teacher ID

**SET** t6's bounds to (138,65,130,25)

**DECLARE** a JLabel l7 for Department

**SET** l7's font to Times new Roman, plain, size 16

**SET** l7's bounds to (10,60,115,30)

**DECLARE** a JTextField t7 for Teacher ID

**SET** t7's bounds to (138,65,130,25)

**DECLARE** a JLabel l9 for Year of Experience

**SET** l9's font to Times new Roman, plain, size 16

**SET** l9's bounds to (10,60,115,30)

**DECLARE** a JTextField t9 for Teacher ID

**SET** t9's bounds to (138,65,130,25)

**DECLARE** a JLabel l10 for Department

**SET** l10's font to Times new Roman, plain, size 16

**SET** l10's bounds to (10,60,115,30)

**DECLARE** a JTextField t10 for Teacher ID

**SET** t10's bounds to (138,65,130,25)

**DECLARE** a JLabel l11 Graaded Score

**SET** l11's font to Times new Roman, plain, size 16

**SET** l11's bounds to (10,60,115,30)

**DECLARE** a JTextField t11 for Teacher ID

**SET** t11's bounds to (138,65,130,25)

**DECLARE** a JLabel l12 for Years of Experience

```
SET l12's font to Times new Roman, plain, size 16
SET l12's bounds to (10,60,115,30)
DECLARE a JTextField t12 for Teacher ID
SET t12's bounds to (138,65,130,25)
```

```
DECLARE a JLabel l13for Teacher id
SET l13's font to Times new Roman, plain, size 16
SET l13's bounds to (10,60,115,30)
DECLARE a JTextField t13 for Teacher ID
SET t13's bounds to (138,65,130,25)
```

```
CREATE a JButton b1 for grading
SET b1's bounds to (840, 270, 120, 30)
SET b1's background color to MAGENTA
SET b1's font to Times new Roman, plain, size 14
add ActionListener to b1:
```

```
DECLARE ActionListener:
```

```
actionPerformed method(ae):
```

```
try:
```

```
teacherID = t13.getText()
```

```
gradedScoreStr = t11.getText()
```

```
department = t10.getText()
```

```
yearsOfExperienceStr = t12.getText()
```

```
IF teacherID.isEmpty() or
gradedScoreStr.isEmpty()department.isEmpty() or
yearsOfExperienceStr.isEmpty():
```

```
    Add "Please fill all the fields." to
```

```
JOptionPane.showMessageDialog(null)
```

```
RETURN
```

**END IF**

**ADD ConvertToInt(gradedScoreStr) to gradedScore**

**ADD ConvertToInt(yearsOfExperienceStr) to  
yearsOfExperience**

**IF gradedScore == null or yearsOfExperience == null:**

**add "Enter correct numerical values for Graded Score and  
Years of Experience" to JOptionPane.showMessageDialog(null)**

**RETURN**

**END IF**

**ADD false to teacherFound**

**FOR EACH teacher in tlist:**

**IF teacherID equals teacher.getID():**

**IF teacher is an instance of Lecturer:**

**ADD teacher to Lecturer L1**

**Add L1.gradeAssignment(gradedScore, department,  
yearsOfExperience) to L1**

**ADD true to teacherFound**

**Add "Teacher ID: " + teacherID + "\nGraded Score: " +  
gradedScore + "\nDepartment: " + department + "\nYears of Experience:  
" + yearsOfExperience to JOptionPane.showMessageDialog(null,  
"Information", INFORMATION\_MESSAGE)**

**break**

**END IF**

**END IF**

**END FOR**

**IF teacherFound:**

```
        ADD "!!!Thank You!!! Graded Score Successful" to
JOptionPane.showMessageDialog(null)
    ELSE:
        ADD "Teacher not found with the given ID" to
JOptionPane.showMessageDialog(null)
    END IF

    CATCH Exception e:
        ADD "Enter Correct Information to Add Graded Score" to
JOptionPane.showMessageDialog(null)
    END TRY
END actionPerformed
END ActionListener
END ADD
```

```
    DECLARE JButton b2 for displaying
    SET b2's bounds to (20,270,120,30)
    SET b2's background color to GREEN
    SET b2's font to times new roman, plain, size 14
    ADD ActionListener to b2:
    DECLARE ActionListener:
    actionPerformed method(ae):
        FOR EACH teacher in tlist:
            IF teacher is an instance of Lecturer:
                CALL teacher.display()
            END IF
        END FOR
    END actionPerformed
    END ActionListener
End Add
```

**DECLARE** JButton b3 for adding  
**SET** b3's bounds to (1040, 70, 120, 30)  
**SET** b3's background color to GREEN  
**SET** b3's font to Times new Roman, plain, size 14  
**ADD** an ActionListener to b3 to handle adding

**DECLARE** JButton b4 for clearing  
**SET** b4's bounds to (1040,120,120,30)  
**SET** b4's background color to ORANGE  
**SET** b4's font to Times new Roman, plain, size 14  
**ADD** an ActionListener to b4 to handle clearing

**ADD** JLabels, JTextFields, and JButtons to p1

**CREATE** a JPanel p2  
**SET** p2's bounds to (0,325,1300,320)  
**SET** p2's background color to LIGHT\_GRAY  
**SET** p2's layout to null  
**ADD** p2 to jf

**CREATE** a JLabel heading1 and set its text to "TUTOR"  
**SET** heading1's font to Times new Roman, plain, size 35  
**SET** heading1's bounds to (475,5,230,30)

**DECLARE** a JLabel tl1 for Teacher ID  
**SET** tl1's font to Times new Roman, plain, size 16  
**SET** tl1's bounds to (10,60,115,30)  
**DECLARE** a JTextField tt1for Teacher ID  
**SET** tt1's bounds to (138,65,130,25)

**DECLARE** a JLabel tl2 for Teacher name

**SET** tl2's font to Times new Roman, plain, size 16  
**SET** tl2's bounds to (10,60,115,30)  
**DECLARE** a JTextField tt2for Teacher ID  
**SET** tt2's bounds to (138,65,130,25)

**DECLARE** a JLabel tl3 for Address  
**SET** tl3's font to Times new Roman, plain, size 16  
**SET** tl3's bounds to (10,60,115,30)  
**DECLARE** a JTextField tt3 for Teacher ID  
**SET** tt3's bounds to (138,65,130,25)

**DECLARE** a JLabel tl4 for Working Type  
**SET** tl4's font to Times new Roman, plain, size 16  
**SET** tl4's bounds to (10,60,115,30)  
**DECLARE** a JTextField tt4for Working Type  
**SET** tt4's bounds to (138,65,130,25)

**DECLARE** a JLabel tl5 for Employment Status  
**SET** tl5's font to Times new Roman, plain, size 16  
**SET** tl5's bounds to (10,60,115,30)  
**DECLARE** a JTextField tt5 for Employment Status  
**SET** tt5's bounds to (138,65,130,25)

**DECLARE** a JLabel tl6 for Working Hours  
**SET** tl6's font to Times new Roman, plain, size 16  
**SET** tl6's bounds to (10,60,115,30)  
**DECLARE** a JTextField tt6 for Working Hours  
**SET** tt6's bounds to (138,65,130,25)

**DECLARE** a JLabel tl7 for Salary

**SET** tl7's font to Times new Roman, plain, size 16  
**SET** tl7's bounds to (10,60,115,30)  
**DECLARE** a JTextField tt7 for Salary  
**SET** tt7's bounds to (138,65,130,25)

**DECLARE** a JLabel tl8 for Specialization  
**SET** tl8's font to Times new Roman, plain, size 16  
**SET** tl8's bounds to (10,60,115,30)  
**DECLARE** a JTextField tt8 for Specialization  
**SET** tt8's bounds to (138,65,130,25)

**DECLARE** a JLabel tl9 for Academic Qualification  
**SET** tl9's font to Times new Roman, plain, size 16  
**SET** tl9's bounds to (10,60,115,30)  
**DECLARE** a JTextField tt9 for Academic Qualification  
**SET** tt9's bounds to (138,65,130,25)

**DECLARE** a JLabel tl10 for Performance Index  
**SET** tl10's font to Times new Roman, plain, size 16  
**SET** tl10's bounds to (10,60,115,30)  
**DECLARE** a JTextField tt10 for Performance Index  
  
**SET** tt10's bounds to (138,65,130,25)

**DECLARE** a JLabel tl11 for New Salary  
**SET** tl11's font to Times new Roman, plain, size 16  
**SET** tl11's bounds to (10,60,115,30)  
**DECLARE** a JTextField tt11 for New Salary  
**SET** tt11's bounds to (138,65,130,25)



**DECLARE** a JLabel tl12 for New Performance Index  
**SET** tl12's font to Times new Roman, plain, size 16  
**SET** tl12's bounds to (10,60,115,30)  
**DECLARE** a JTextField tt12 for Performance Index  
**SET** tt12's bounds to (138,65,130,25)

**DECLARE** a JLabel tl14 for Working Hours  
**SET** tl14's font to Times new Roman, plain, size 16  
**SET** tl14's bounds to (10,60,115,30)  
**DECLARE** a JTextField tt14 for Working Hours  
**SET** tt14's bounds to (138,65,130,25)

**DECLARE** JButton tb1 for adding  
**SET** tb1's bounds to (1040, 70, 120, 30)  
**SET** tb1's background color to GREEN  
**SET** tb1's font to Times new Roman, plain, size 14  
**ADD** an ActionListener to tb1 to handle adding

**DECLARE** JButton tb2 for clearing  
**SET** tb2's bounds to (1040,120,120,30)  
**SET** tb2's background color to ORANGE  
**SET** tb2's font to Times new Roman, plain, size 14  
**ADD** an ActionListener to tb2 to handle clearing

**DECLARE** JButton tb3 for Salary  
**SET** tb3's bounds to (1040, 70, 120, 30)  
**SET** tb3's background color to GREEN

```
    SET tb3's font to Times new Roman, plain, size 14
ADD ActionListener to tb3:
    DECLARE ActionListener:
        actionPerformed method(ae):
            FOR EACH teacher in tlist:
                TRY:
                    SET newSalary = ConvertToInt(tns.getText())
                    SET newPerformanceIndex = ConvertToInt(tnp.getText())
                    IF teacher is an instance of Tutor:
                        SET T1 = downcast teacher to Tutor
                        SET T1.setSalary(newSalary, newPerformanceIndex)
                        DISPLAY MessageDialog(null, "!!!Thank You!!! ")
                CATCH Exception e:
                    JOptionPane.showMessageDialog(null, "Enter Correct
Information")
            END FOR
        END actionPerformed
    END ActionListener
END ADD
```

```
    DECLARE JButton tb4 for displaying
    SET tb4's bounds to (20,270,120,30)
    SET tb4's background color to GREEN
    SET tb4's font to Times new Roman, plain, size 14
    ADD an ActionListener to tb4 to handle displaying
```

**ADD** JLabels, JTextFields, and JButtons to p2

**END** main method

**END** class TeacherGUI

**END**

#### 4.Method description

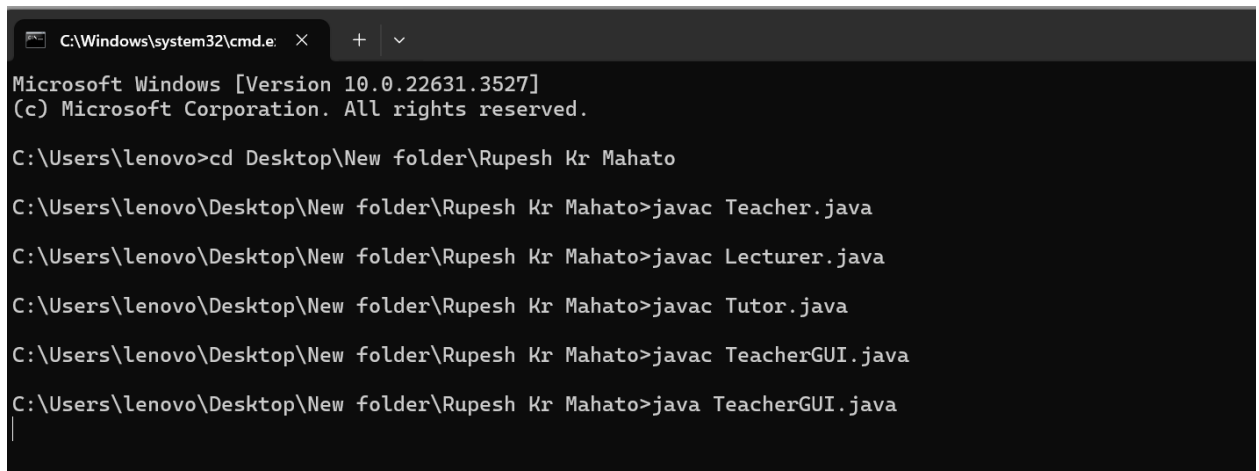
- Add lecturer: To operate this button user have to enter all the required fields like teacher id, teacher name, working type, working hours, address, department, years of experience. After filling all the fields if the button is clicked then teacher is added and suitable message is displayed.
- Clear: To operate this button user must fill atleast one of the field and when this button is clicked it displays confirmation message and if user click yes then the textfield is clear.
- Display: In both Lecturer and Tutor to display all the details of lecturer and tutor the required fields are filled and display button is clicked. When the button is clicked it calls display method from their required class.
- Grade assignment: To operate this button the fields like teacher id, department, graded score, and years of experience are filled and the button is clicked it display information dialog containing all the information given for grading teacher and if user click yes grade method from Lecturer is called and then teacher is graded and its grade is shown in terminal.
- Set salary: To operate this button all the fields like teacher id, new performance index, new salary are entered and when the salary button is clicked it calls set salary method from tutor and sets the salary.
- Remove: To operate this button the teacher id is entered when this button is pressed it compare the entered id with existing id if matches found it removes the Tutor of the same id.

## 5. Testing

### Test 1

Test	1
Action performed	Program file was given to command prompt for compilation and run.
Expected result	All file should be compiled and run.
Actual result	All file was compiled and run successfully.
Conclusion	Hence, the test was successful

*Table 1: Test 1*



```

C:\Windows\system32\cmd.e  X  +  v
Microsoft Windows [Version 10.0.22631.3527]
(c) Microsoft Corporation. All rights reserved.

C:\Users\lenovo>cd Desktop\New folder\Rupesh Kr Mahato
C:\Users\lenovo\Desktop\New folder\Rupesh Kr Mahato>javac Teacher.java
C:\Users\lenovo\Desktop\New folder\Rupesh Kr Mahato>javac Lecturer.java
C:\Users\lenovo\Desktop\New folder\Rupesh Kr Mahato>javac Tutor.java
C:\Users\lenovo\Desktop\New folder\Rupesh Kr Mahato>javac TeacherGUI.java
C:\Users\lenovo\Desktop\New folder\Rupesh Kr Mahato>java TeacherGUI.java
|

```

*Figure 2: compile and run using command prompt*

## Test 2

Test	2
Action performed	All the required fields are filled and then lecturer, tutor were added assignment was graded and salary was set .
Expected result	Lecturer and Tutor should be added. Grade should be assigned, Tutor should be removed salary should be set.
Actual result	Lecturer and Tutor were added. Grade was assigned, Tutor was removed and salary was set.
Conclusion	Hence, the test was successful.

Table 2: test2

## ➤ Adding Lecturer

```

BlueJ: Terminal Window - Rupesh Kr Mahato

Options

Teacher Id = 123
Teacher Name = rupesh
address= kathmandu
Working Type= onsite
Employment Status = fulltime
Working hour is not assigned
Department: IT
Years of Experience: 6
Assignment has not been graded yet.

```

Figure 3:Added lecturer

Figure 4: output after adding lecturer

## ➤ Adding Tutor

The screenshot shows the 'LECTURER' form with fields for Teacher ID (123), Name (rupesh), Address (kathmandu), Working Type (onsite), Employment Status (fulltime), Working Hours (24), Year Of Experience (6), and Department (IT). The 'Add' button is highlighted in green. A message box in the center says 'Tutor added successfully!' with an 'OK' button.

Figure 6: Adding tutor

BlueJ: Terminal Window - Rupesh Kr Mahato

## Options

Teacher Id = 321  
 Teacher Name = ram  
 Address= kathmandu  
 Working Type= onsite  
 Employment Status = fulltime  
 Working hours = 25  
 Certigied status not certi

Figure 5: output after adding tutor

## ➤ Grade assignment from lecturer

The screenshot shows the 'LECTURER' form with fields for Teacher ID (123), Name (ram), Address (ktm), Working Type (onsite), Employment Status (fulltime), Working Hours (25), Year Of Experience (6), and Department (kkk). The 'Grade' button is highlighted in pink. A message box in the center shows the graded score: 'Teacher ID: 123, Graded Score: 70, Department: kkk, Years of Experience: 6' with an 'OK' button.

Figure 7: Input for grade assignment

The screenshot shows the 'LECTURER' form with fields for Teacher ID (123), Name (ram), Address (ktm), Working Type (onsite), Employment Status (fulltime), Working Hours (25), Year Of Experience (6), and Department (kkk). The 'Grade' button is highlighted in pink. A message box in the center says '!!!Thank You!!! Graded Score Successful' with an 'OK' button.

Figure 8: Lecturer was graded

BlueJ: Terminal Window - Rupesh Kr Mahato

## Options

Graded score = A  
 Teacher Id = 123  
 Teacher Name = ram  
 Address= ktm  
 Working Type= onsite  
 Employment Status = fulltime  
 working hour is not assigned  
 Department: kkk  
 Years of Experience: 6  
 Graded Score: 0

Figure 9: output after lecturer is graded

## ➤ Set salary

The screenshot shows a Java Swing application window titled "LECTURER". It contains several input fields for teacher information. A "Set Salary" section is active, showing "Teacher ID: 123", "New Salary: 30000", and "New performance index: 7". A "Thank You!!" dialog box is displayed in the center.

Figure 11: set salary

BlueJ: Terminal Window - Rupesh Kr Mahato

Options

Teacher Id = 123  
 Teacher Name = ram  
 Address= ktm  
 Working Type= onsite  
 Employment Status = fulltime  
 Working hours = 24  
 Salary: 23000.0  
 Specialization: IT  
 Academic Qualifications: master  
 Performance Index: 7

Figure 10: output after setting salary

## Test 3



Test	3
Action Performed	String value was entered in teacher ID text field.
Expected result	Dialog box displaying invalid input for numeric fields should appear and ask user to enter valid integer.
Actual result	Dialog box displaying invalid input for numeric fields was appeared asked user to enter valid integer.
Conclusion	Hence, the test was successful.

Table 3:Test3

The screenshot shows a web application window titled "coursework" with a "LECTURER" form. The form has two sections: "LECTURER" and "Grade Assignment".

**LECTURER Section:**

- Teacher ID:
- Working Type:
- Department:
- Teacher Name:
- Employment Status:
- Address:
- Working Hours:
- Year Of Experience:
- Buttons: "Add" (green), "clear" (yellow)

**Grade Assignment Section:**

- Department:
- Graded Score:
- Year Of Experience:
- Teacher ID:
- Buttons: "Display" (green), "Grade" (pink)

**Message Dialog Box:**

Message [X]  
 i Invalid input for numeric fields. Please enter valid integers.  
 OK

Figure 12: Dialog box test

## 5. Error detection and correction

### 1. Syntax error

Syntax errors are errors in the source code that include things like spelling and punctuation mistakes, wrong labels, etc. These errors lead to error messages from the compiler. They're displayed in a separate window, showing the error type and line number, making it easier to correct them in the editing window. (Attaway, 2019)

- Error detection1:  
Paranthesis was missing while creating the arraylist.

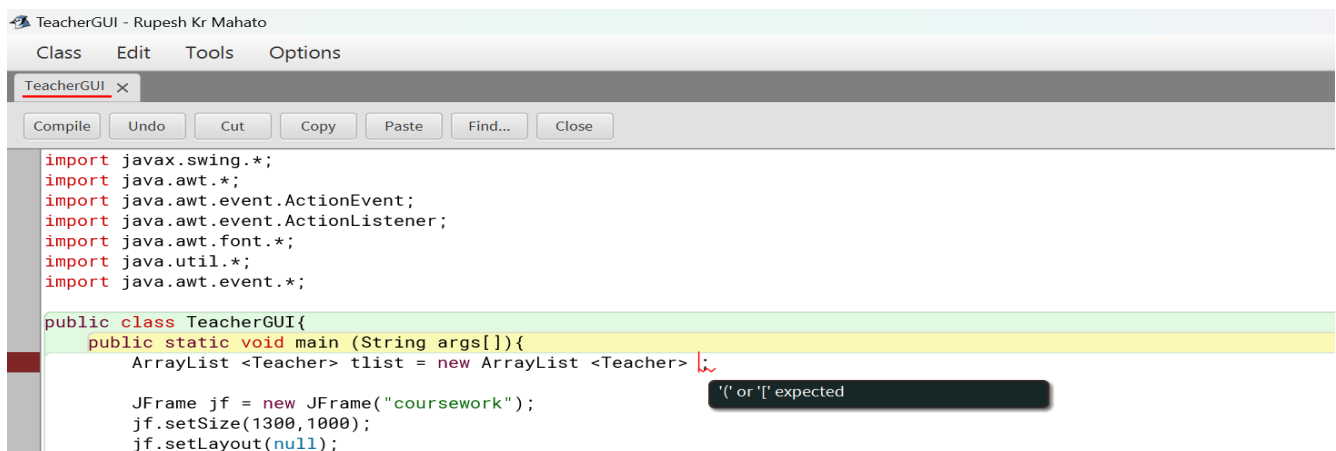


Figure 13:Error detection 1(missing paranthesis)

- Error correction 1  
Paranthesis was added while creating arraylist to correct the error.

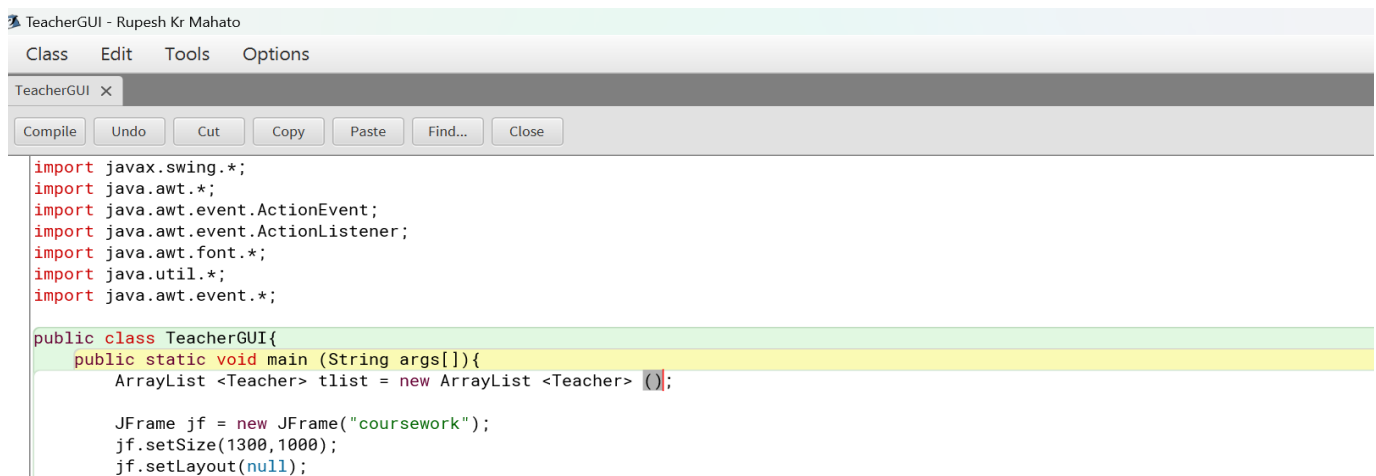


Figure 14: Correction of error 1

- Error detection 2  
Semicolon was missing at the end of line.

```

JButton b1 = new JButton("Grade");
b1.setBounds(840, 270, 120, 30);
b1.setBackground(Color.MAGENTA);
b1.setFont(new Font("Times new Roman", Font.PLAIN, 14));
b1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
        try {
            String teacherID = t13.getText(); // Assuming t13 is your JTextField for Teacher ID
            String gradedScoreStr = t11.getText();
            String department = t10.getText();

```

Figure 15: Error 2 (missing semicolon)

- Error correction 2  
Semicolon was added to correct the error.

```

JButton b1 = new JButton("Grade");
b1.setBounds(840, 270, 120, 30);
b1.setBackground(Color.MAGENTA);
b1.setFont(new Font("Times new Roman", Font.PLAIN, 14));
b1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
        try {
            String teacherID = t13.getText(); // Assuming t13 is your JTextField for Teacher ID
            String gradedScoreStr = t11.getText();
            String department = t10.getText();
            String yearsOfExperienceStr = t12.getText();

```

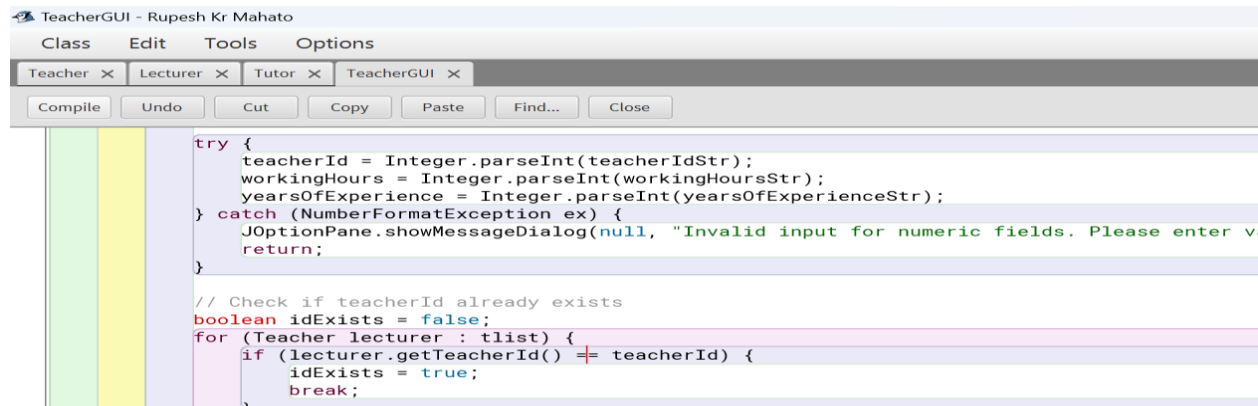
Figure 16: Correction of error 2

## 2. Semantic error

A semantic error occurs when a programmer misinterprets how the programming language functions and writes code that doesn't align with the language's rules. Although the code might be structurally correct, it goes against the rules or "semantics" of the language. (stackOverflow, n.d.)

- Error detection 1

if (lecturer.getTeacherId() != teacherId in place double equals to it was set to not equal to.



```

try {
    teacherId = Integer.parseInt(teacherIdStr);
    workingHours = Integer.parseInt(workingHoursStr);
    yearsOfExperience = Integer.parseInt(yearsOfExperienceStr);
} catch (NumberFormatException ex) {
    JOptionPane.showMessageDialog(null, "Invalid input for numeric fields. Please enter v
    return;
}

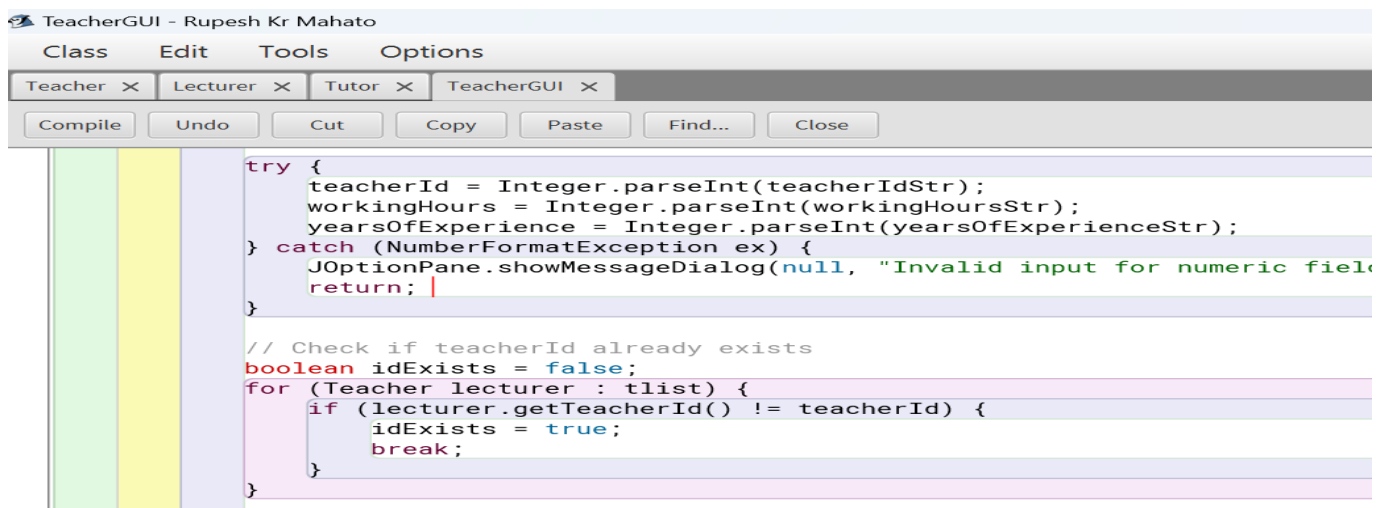
// Check if teacherId already exists
boolean idExists = false;
for (Teacher lecturer : tlist) {
    if (lecturer.getTeacherId() != teacherId) {
        idExists = true;
        break;
    }
}

```

Figure 17: Semantic error 1

- Error correction 1

Double equals was added to correct.  
if (lecturer.getTeacherId() == teacherId



```

try {
    teacherId = Integer.parseInt(teacherIdStr);
    workingHours = Integer.parseInt(workingHoursStr);
    yearsOfExperience = Integer.parseInt(yearsOfExperienceStr);
} catch (NumberFormatException ex) {
    JOptionPane.showMessageDialog(null, "Invalid input for numeric field
    return;
}

// Check if teacherId already exists
boolean idExists = false;
for (Teacher lecturer : tlist) {
    if (lecturer.getTeacherId() == teacherId) {
        idExists = true;
        break;
    }
}

```

Figure 18: Semantic error correction 1

## 6.Conclusion

This Java programming course was a significant learning experience for me as it required individual assignments. Through this coursework, I gained valuable skills in researching and utilizing various tools essential for development. Specifically, I became proficient in using tools like BlueJ for coding, Microsoft Word for documentation, and Draw.io for drawing class diagrams. While BlueJ facilitated the coding process, Microsoft Word helped me in creating comprehensive documentation, and Draw.io proved instrumental in visualizing class structures. Moreover, the coursework provided me with insights into GUI components and event handling in Java. One of the key takeaways was learning to handle errors effectively, including exceptions. This course not only enhanced my technical skills but also sharpened my ability to organize and present information efficiently.

Throughout this project, I faced several challenges that needed to be overcome. At first, I had difficulty with accuracy, dealing with many syntax and logical errors. However, with perseverance and guidance from my tutor, along with help from various websites, I successfully tackled these issues. Despite the setbacks, I was able to work through the problems and finish the coursework on time. This experience not only improved my problem-solving abilities but also emphasized the importance of seeking help and making the most of available resources when encountering difficulties.

I had a fantastic experience working on this coursework. It taught me the fundamentals of Java programming including GUI components and event handling, which I believe will be valuable in my future career. Additionally, it enhanced my research skills, and now I feel confident in my understanding of basic Java programming after completing this coursework.

## Bibliography

Attaway, S., 2019. *ScienceDirect*. [Online]

Available at: <https://www.sciencedirect.com/topics/engineering/syntax-error#:~:text=Syntax%20errors%20are%20mistakes%20in%20the%20source%20code%2C%20such%20as,corrected%20in%20the%20edit%20window.>

[Accessed 8 May 2024].

geeksforgeeks, 2024. *geeksforgeeks*. [Online]

Available at: <https://www.geeksforgeeks.org/introduction-to-java/>

[Accessed 7 May 2024].

Paraschiv, L., 2023. *FOTC*. [Online]

Available at: <https://fotc.com/blog/draw-io-online-guide/>

[Accessed 8 May 2024].

Rouse, M., 2013. *Techopedia*. [Online]

Available at: <https://www.techopedia.com/definition/29530/bluej>

[Accessed 8 May 2024].

stackoverflow, n.d. *stackoverflow*. [Online]

Available at: <https://stackoverflow.com/questions/77391755/differentiating-semantic-error-logic-error-and-runtime-error#:~:text=A%20semantic%20error%20is%20when,%E2%80%9Csemantics%E2%80%9D%20of%20the%20language.>

[Accessed 8 May 2024].

Urwin, M., 2024. *Builtib*. [Online]

Available at: <https://builtin.com/data-science/pseudocode>

[Accessed 8 May 2024].

## Appendix

### Teacher

```
public class Teacher{  
    // variables declaration  
    private int teacherId;  
    private String teacherName;  
    private String address;  
    private String workingType;  
    private String employmentStatus;  
    private int workingHours;  
  
    // parameter constructor  
    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;  
    }  
  
    // accessor method for each attributes  
    public int getTeacherId(){  
        return teacherId;  
    }  
    public String getTeacherName(){  
        return teacherName;  
    }  
}
```

```
public String getAddress(){
    return address;
}

public String getWorkingType(){
    return workingType;
}

public String getEmploymentStatus(){
    return employmentStatus;
}

public int getWorkingHours(){
    return workingHours;
}

// mutator method to set working hours
public void setWorkingHours(int newWorkingHours){
    this.workingHours = newWorkingHours;
}

public void display(){
    System.out.println("Teacher Id = " + teacherId);
    System.out.println("Teacher Name = " + teacherName);
    System.out.println("Address= " + address);
    System.out.println("Working Type= " + workingType);
    System.out.println("Employment Status = " + employmentStatus);
    if(workingHours > 0){

        System.out.println("Working hours = " + workingHours);}
}
```



```
        else{
            System.out.println("working hour is not assigned");
        }
    }

}
```

### Lecturer

```
public class Lecturer extends Teacher{
    // variables declaration
    private String department;
    private int yearsOfExperience;
    private int gradedScore;
    private boolean hasGraded;

    // Constructor
    public Lecturer(int teacherId, String teacherName, String address, String
workingType, int workingHours, String employmentStatus,
        String department, int yearsOfExperience) {
        super(teacherId, teacherName, address, workingType, employmentStatus);
        this.department = department;
        this.yearsOfExperience = yearsOfExperience;
        this.gradedScore = 0; // Initialize gradedScore to 0
        this.hasGraded = false; // Initialize hasGraded to false
    }

    // Accessor methods
    public String getDepartment() {
```

```
        return department;
    }

    public int getYearsOfExperience() {
        return yearsOfExperience;
    }

    public int getGradedScore() {
        return gradedScore;
    }

    public boolean hasGraded() {
        return hasGraded;
    }

    // Mutator method for gradedScore
    public void setGradedScore(int newGradedScore) {
        gradedScore = newGradedScore;
    }

    public void gradeAssignment(int gradedScore, String department , int
yearsOfExperience) {
        if (!hasGraded) {
            if (yearsOfExperience >= 5 && department.equals(department)) {
                if (gradedScore >= 70) {
                    System.out.println("Graded score = A");
                } else if (gradedScore >= 60) {
                    System.out.println("Graded score = B");
                } else if (gradedScore >= 50) {
                    System.out.println("Graded score = C");
                }
            }
        }
    }
}
```

```
        } else if (gradedScore >= 40) {  
            System.out.println("Graded score = D");  
        } else {  
            System.out.println("Graded score = E");  
        }  
        hasGraded = true;  
    } else {  
        System.out.println("Lecturer cannot grade assignment for this student.");  
    }  
} else {  
    System.out.println("Assignment has already been graded.");  
}  
}
```

// Display method

```
public void display() {  
    super.display(); // Call the display method in the Teacher class  
  
    System.out.println("Department: " + department);  
    System.out.println("Years of Experience: " + yearsOfExperience);  
  
    if (hasGraded) {  
        System.out.println("Graded Score: " + gradedScore);  
    } else {  
        System.out.println("Assignment has not been graded yet.");  
    }  
}
```

```
}
```

## **Tutor**

```
public class Tutor extends Teacher {  
    // variables declaration  
    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);  
        this.setWorkingHours(workingHours);  
        this.salary = salary;  
        this.specialization = specialization;  
        this.academicQualifications = academicQualifications;  
        this.performanceIndex = performanceIndex;  
        this.isCertified = false;  
    }  
  
    // Accessor methods  
    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;  
}
```

```
// Mutator method for salary
```

```
public void setSalary(int newSalary, int newPerformanceIndex) {  
    if (newPerformanceIndex > 5 && getWorkingHours() > 20) {  
        double appraisalPercentage;  
        if (newPerformanceIndex >= 5 && newPerformanceIndex <= 7) {  
            appraisalPercentage = 0.05;  
        } else if (newPerformanceIndex >= 8 && newPerformanceIndex <= 9) {  
            appraisalPercentage = 0.10;  
        } else { // newPerformanceIndex = 10  
            appraisalPercentage = 0.20;  
        }  
    }  
}
```

```
    }  
    double appraisalAmount = newSalary * appraisalPercentage;  
    this.salary += appraisalAmount;  
    this.isCertified = true;  
    this.performanceIndex = newPerformanceIndex;  
} else {  
    System.out.println("Salary cannot be approved. Tutor is not certified.");  
}  
}  
  
// Method to remove Tutor  
public void removeTutor() {  
    if (!isCertified) {  
        this.salary = 0;  
        this.specialization = "";  
        this.academicQualifications = "";  
        this.performanceIndex = 0;  
        this.isCertified = false;  
    }  
}  
  
// Display method for Tutor  
public void display() {  
    super.display();  
    if (isCertified) {  
        System.out.println("Salary: " + salary);  
        System.out.println("Specialization: " + specialization);  
        System.out.println("Academic Qualifications: " + academicQualifications);  
    }  
}
```

```

        System.out.println("Performance Index: " + performanceIndex);
    } else {
        System.out.println("Certigied status not certi " ); // Call display method in the
parent class
    }
}
}
}

```

### TeacherGUI

```

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.font.*;
import java.util.*;
import java.awt.event.*;

public class TeacherGUI{
    public static void main (String args[]){
        ArrayList <Teacher> tlist = new ArrayList <Teacher> ();

        JFrame jf = new JFrame("coursework");
        jf.setSize(1300,1000);
        jf.setLayout(null);
        jf.setVisible(true);

        JPanel p1 = new JPanel();
    }
}

```

```
p1.setBounds(0,0,1300,315);  
p1.setBackground(Color.LIGHT_GRAY);  
p1.setLayout(null);  
jf.add(p1);
```

```
JLabel heading = new JLabel("LECTURER");  
heading.setFont(new Font("Times new Roman", Font.PLAIN, 35));  
heading.setBounds(475,5,230,30);
```

```
JLabel l1 = new JLabel("Teacher ID: ");  
l1.setFont(new Font("Times new Roman", Font.PLAIN, 16));  
l1.setBounds(10,60,115,30);  
JTextField t1 = new JTextField();  
t1.setBounds(138,65,130,25);
```

```
JLabel l2 = new JLabel("Teacher Name: ");  
l2.setFont(new Font("Times new Roman", Font.PLAIN, 16));  
l2.setBounds(10,100,130,30);  
JTextField t2 = new JTextField();  
t2.setBounds(138,103,130,25);
```

```
JLabel l3 = new JLabel("Address: ");  
l3.setFont(new Font("Times new Roman", Font.PLAIN, 16));  
l3.setBounds(10,140,100,30);  
JTextField t3 = new JTextField();  
t3.setBounds(138,143,130,25);
```



```
JLabel l4 = new JLabel("Working Type: ");  
l4.setFont(new Font("Times new Roman", Font.PLAIN, 16));  
l4.setBounds(350,60,140,30);  
JTextField t4 = new JTextField();  
t4.setBounds(516,65,130,25);
```

```
JLabel l5 = new JLabel("Employment Status: ");  
l5.setFont(new Font("Times new Roman", Font.PLAIN, 16));  
l5.setBounds(350,100,180,30);  
JTextField t5 = new JTextField();  
t5.setBounds(516,103,130,25);
```

```
JLabel l6 = new JLabel("Working Hours: ");  
l6.setFont(new Font("Times new Roman", Font.PLAIN, 16));  
l6.setBounds(350,140,180,30);  
JTextField t6 = new JTextField();  
t6.setBounds(516,143,130,25);
```

```
JLabel l7 = new JLabel("Department: ");  
l7.setFont(new Font("Times new Roman", Font.PLAIN, 16));  
l7.setBounds(685,60,130,30);  
JTextField t7 = new JTextField();  
t7.setBounds(870,65,130,25);
```

```
/*JLabel l8 = new JLabel("Graded Score: ");  
l8.setFont(new Font("Times new Roman", Font.PLAIN, 16));  
l8.setBounds(685,100,130,30);  
JTextField t8 = new JTextField();  
t8.setBounds(870,103,130,25);*/
```

```
JLabel l9 = new JLabel("Year Of Experience: ");  
l9.setFont(new Font("Times new Roman", Font.PLAIN, 16));  
l9.setBounds(685,140,140,30);  
JTextField t9 = new JTextField();  
t9.setBounds(870,143,143,25);
```

```
JLabel sh1 = new JLabel("Grade Assignment ");  
sh1.setFont(new Font("Times new Roman", Font.PLAIN, 25));  
sh1.setBounds(535,180,200,30);
```

```
JLabel l10 = new JLabel("Department: ");  
l10.setFont(new Font("Times new Roman", Font.PLAIN, 16));  
l10.setBounds(10,230,130,30);  
JTextField t10 = new JTextField();  
t10.setBounds(138,233,130,25);
```

```
JLabel l11 = new JLabel("Graded Score: ");  
l11.setFont(new Font("Times new Roman", Font.PLAIN, 16));  
l11.setBounds(350,230,130,30);  
JTextField t11 = new JTextField();  
t11.setBounds(500,233,130,25);
```

```
JLabel l12 = new JLabel("Year Of Experience: ");  
l12.setFont(new Font("Times new Roman", Font.PLAIN, 16));  
l12.setBounds(685,230,140,30);  
JTextField t12 = new JTextField();  
t12.setBounds(855,233,143,25);
```

```
JLabel l13 = new JLabel("Teacher ID: ");  
l13.setFont(new Font("Times new Roman", Font.PLAIN, 16));  
l13.setBounds(1020,230,140,30);  
JTextField t13 = new JTextField();  
t13.setBounds(1120,233,130,25);
```

```
JButton b1 = new JButton("Grade");  
b1.setBounds(840, 270, 120, 30);  
b1.setBackground(Color.MAGENTA);
```

```
b1.setFont(new Font("Times new Roman", Font.PLAIN, 14));
b1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
        try {
            String teacherID = t13.getText(); // Assuming t13 is your JTextField for Teacher
ID
            String gradedScoreStr = t11.getText();
            String department = t10.getText();
            String yearsOfExperienceStr = t12.getText();

            // Check if any field is empty
            if (teacherID.isEmpty() || gradedScoreStr.isEmpty() || department.isEmpty() ||
yearsOfExperienceStr.isEmpty()) {
                JOptionPane.showMessageDialog(null, "Please fill all the fields.");
                return;
            }

            int gradedScore = 0;
            int yearsOfExperience = 0;

            try {
                gradedScore = Integer.parseInt(gradedScoreStr);
                yearsOfExperience = Integer.parseInt(yearsOfExperienceStr);
            } catch (NumberFormatException e) {
                JOptionPane.showMessageDialog(null, "Enter correct numerical values for
Graded Score and Years of Experience");
                return;
            }
        }
    }
});
```

```
boolean teacherFound = false;
for (Teacher t : tlist) {
    if (t13.getText().equals(t1.getText())) {
        if (t instanceof Lecturer) {
            //downcasting
            Lecturer L1 = (Lecturer) t;
            L1.gradeAssignment(gradedScore, department, yearsOfExperience);
            teacherFound = true;

            JOptionPane.showMessageDialog(null, "Teacher ID: " + teacherID +
"\nGraded Score: " + gradedScore + "\nDepartment: " + department + "\nYears of
Experience: " + yearsOfExperience, "Information",
JOptionPane.INFORMATION_MESSAGE);

            break;
            // No need to continue if teacher is found
        }
    }
}

if (teacherFound) {
    JOptionPane.showMessageDialog(null, "!!!Thank You!!! Graded Score
Successful");
} else {
    JOptionPane.showMessageDialog(null, "Teacher not found with the given
ID");
}
} catch (Exception e) {
    JOptionPane.showMessageDialog(null, "Enter Correct Information to Add
Graded Score");
}
```

```
    }  
});
```

```
        JButton b2 = new JButton("Display");  
        b2.setBounds(20,270,120,30);  
        b2.setBackground(Color.GREEN);  
        b2.setFont(new Font("Times new Roman", Font.PLAIN, 14));  
        b2.addActionListener(new ActionListener() {  
            public void actionPerformed(ActionEvent ae) {  
                for(Teacher t : tlist)  
                {  
                    if(t instanceof Lecturer)  
                    {  
                        t.display();  
                    }  
                }  
            }  
        });
```

```
JButton b3 = new JButton("Add");  
b3.setBounds(1040, 70, 120, 30);  
b3.setBackground(Color.GREEN);  
b3.setFont(new Font("Times new Roman", Font.PLAIN, 14));
```

```
b3.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent ae) {  
        try {  
            String teacherIdStr = t1.getText();  
            String teacherName = t2.getText();  
            String address = t3.getText();  
            String workingType = t4.getText();  
            String employmentStatus = t5.getText();  
            String workingHoursStr = t6.getText();  
            String department = t7.getText();  
            String yearsOfExperienceStr = t9.getText();  
  
            // Check if any field is empty  
            if (teacherIdStr.isEmpty() || teacherName.isEmpty() || address.isEmpty() ||  
                workingType.isEmpty() || employmentStatus.isEmpty() ||  
                workingHoursStr.isEmpty() ||  
                department.isEmpty() || yearsOfExperienceStr.isEmpty()) {  
                JOptionPane.showMessageDialog(null, "Please fill all the fields.");  
                return;  
            }  
        }  
  
        int teacherId = 0;  
        int workingHours = 0;  
        int yearsOfExperience = 0;  
  
        try {  
            teacherId = Integer.parseInt(teacherIdStr);  
            workingHours = Integer.parseInt(workingHoursStr);  
            yearsOfExperience = Integer.parseInt(yearsOfExperienceStr);
```

```
    } catch (NumberFormatException ex) {  
        JOptionPane.showMessageDialog(null, "Invalid input for numeric fields.  
Please enter valid integers.");  
        return;  
    }  
  
    // Check if teacherId already exists  
    boolean idExists = false;  
    for (Teacher lecturer : tlist) {  
        if (lecturer.getTeacherId() == teacherId) {  
            idExists = true;  
            break;  
        }  
    }  
  
    if (idExists) {  
        JOptionPane.showMessageDialog(null, "Teacher ID already exists! Please  
enter a unique ID.");  
    } else {  
        Lecturer lecturer = new Lecturer(teacherId, teacherName, address,  
workingType, workingHours, employmentStatus, department, yearsOfExperience);  
  
        tlist.add(lecturer);  
  
        JOptionPane.showMessageDialog(null, "Lecturer added successfully!");  
    }  
} catch (Exception ex) {  
    JOptionPane.showMessageDialog(null, "Error occurred. Please make sure you  
have filled all the fields correctly.");  
}
```



```
    }  
});
```

```
        JButton b4 = new JButton("clear");  
        b4.setBounds(1040,120,120,30);  
        b4.setBackground(Color.ORANGE);  
        b4.setFont(new Font("Times new Roman", Font.PLAIN, 14));  
        b4.addActionListener(new ActionListener() {  
            public void actionPerformed(ActionEvent e) {  
                int choice = JOptionPane.showConfirmDialog(null, "Do you want to clear?");  
                if (choice == JOptionPane.YES_OPTION){  
  
                    t1.setText("");  
                    t2.setText("");  
                    t3.setText("");  
                    t4.setText("");  
                    t5.setText("");  
                    t6.setText("");  
                    t7.setText("");  
                    //t8.setText("");  
                    t9.setText("");  
                    t10.setText("");  
                    t11.setText("");  
                    t12.setText("");  
                    t13.setText("");  
  
                }  
            }  
        });
```

```
    else{  
  
        }  
    }  
});
```

```
p1.add(t1);  
p1.add(l1);  
p1.add(l2);  
p1.add(t2);  
p1.add(l3);  
p1.add(t3);  
p1.add(l4);  
p1.add(t4);  
p1.add(l5);  
p1.add(t5);  
p1.add(l6);  
p1.add(t6);  
p1.add(l7);  
p1.add(t7);  
//p1.add(l8);  
//p1.add(t8);  
p1.add(l9);  
p1.add(t9);  
p1.add(b1);  
p1.add(b2);
```

```
p1.add(heading);
p1.add(b3);
p1.add(b4);
p1.add(sh1);
p1.add(l10);
p1.add(l11);
p1.add(l12);
p1.add(t10);
p1.add(t11);
p1.add(t12);
p1.add(l13);
p1.add(t13);
```

```
JPanel p2 = new JPanel();
p2.setBounds(0,325,1300,320);
p2.setBackground(Color.LIGHT_GRAY);
p2.setLayout(null);
jf.add(p2);
```

```
JLabel heading1 = new JLabel("TUTOR");
heading1.setFont(new Font("Times new Roman", Font.PLAIN, 35));
heading1.setBounds(475,5,230,30);
```

```
JLabel tl1 = new JLabel("Teacher ID: ");
tl1.setFont(new Font("Times new Roman", Font.PLAIN, 16));
tl1.setBounds(10,60,115,30);
JTextField tt1 = new JTextField();
```

```
tt1.setBounds(138,65,130,25);
```

```
JLabel tl2 = new JLabel("Teacher Name: ");  
tl2.setFont(new Font("Times new Roman", Font.PLAIN, 16));  
tl2.setBounds(10,100,130,30);  
JTextField tt2 = new JTextField();  
tt2.setBounds(138,103,130,25);
```

```
JLabel tl3 = new JLabel("Address: ");  
tl3.setFont(new Font("Times new Roman", Font.PLAIN, 16));  
tl3.setBounds(10,140,100,30);  
JTextField tt3 = new JTextField();  
tt3.setBounds(138,143,130,25);
```

```
JLabel tl4 = new JLabel("Working Type: ");  
tl4.setFont(new Font("Times new Roman", Font.PLAIN, 16));  
tl4.setBounds(350,60,140,30);  
JTextField tt4 = new JTextField();  
tt4.setBounds(542,65,130,25);
```

```
JLabel tl5 = new JLabel("Employment Status: ");  
tl5.setFont(new Font("Times new Roman", Font.PLAIN, 16));  
tl5.setBounds(350,100,180,30);  
JTextField tt5 = new JTextField();
```

```
tt5.setBounds(542,103,130,25);
```

```
JLabel tl6 = new JLabel("Working Hours: ");  
tl6.setFont(new Font("Times new Roman", Font.PLAIN, 16));  
tl6.setBounds(350,140,180,30);  
JTextField tt6 = new JTextField();  
tt6.setBounds(542,143,130,25);
```

```
JLabel tl7 = new JLabel("Salary: ");  
tl7.setFont(new Font("Times new Roman", Font.PLAIN, 16));  
tl7.setBounds(10,180,130,30);  
JTextField tt7 = new JTextField();  
tt7.setBounds(138,183,130,25);
```

```
JLabel tl8 = new JLabel("Specialization: ");  
tl8.setFont(new Font("Times new Roman", Font.PLAIN, 16));  
tl8.setBounds(695,60,130,30);  
JTextField tt8 = new JTextField();  
tt8.setBounds(875,63,130,25);
```

```
JLabel tl9 = new JLabel("Academic Qualification: ");  
tl9.setFont(new Font("Times new Roman", Font.PLAIN, 16));  
tl9.setBounds(695,100,204,30);
```

```
JTextField tt9 = new JTextField();  
tt9.setBounds(875,103,130,25);
```

```
JLabel tl10 = new JLabel("Performance Index: ");  
tl10.setFont(new Font("Times new Roman", Font.PLAIN, 16));  
tl10.setBounds(695,140,180,30);  
JTextField tt10 = new JTextField();  
tt10.setBounds(875,143,130,25);
```

```
JLabel sh2 = new JLabel("Set Salary");  
sh2.setFont(new Font("Times new Roman", Font.PLAIN, 25));  
sh2.setBounds(540,200,180,30);
```

```
JLabel sh3 = new JLabel("Remove Tutor");  
sh3.setFont(new Font("Times new Roman", Font.PLAIN, 25));  
sh3.setBounds(1050,200,180,30);
```

```
JLabel tl11 = new JLabel("New Salary: ");  
tl11.setFont(new Font("Times new Roman", Font.PLAIN, 16));  
tl11.setBounds(240,240,130,30);  
JTextField tns = new JTextField();  
tns.setBounds(340,243,130,25);
```

```
JLabel tl12 = new JLabel("New performance index : ");  
tl12.setFont(new Font("Times new Roman", Font.PLAIN, 16));
```

```
tl12.setBounds(500,240,175,30);  
JTextField tnp = new JTextField();  
tnp.setBounds(690,243,130,25);
```

```
JLabel tl13 = new JLabel("Teacher ID: ");  
tl13.setFont(new Font("Times new Roman", Font.PLAIN, 16));  
tl13.setBounds(10,240,100,30);  
JTextField tt13 = new JTextField();  
tt13.setBounds(100,243,130,25);
```

```
JLabel tl14 = new JLabel("Teacher ID: ");  
tl14.setFont(new Font("Times new Roman", Font.PLAIN, 16));  
tl14.setBounds(1000,240,100,30);  
JTextField tt14 = new JTextField();  
tt14.setBounds(1100,243,130,25);
```

```
JButton tb1 = new JButton("Add a tutor");  
tb1.setBounds(1050, 70, 120, 30);  
tb1.setBackground(Color.GREEN);  
tb1.setFont(new Font("Times new Roman", Font.PLAIN, 14));  
tb1.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent ae) {  
        try {  
            // Check if any field is empty  
            if (tt1.getText().isEmpty() || tt2.getText().isEmpty() || tt3.getText().isEmpty() ||
```

```
tt4.getText().isEmpty() || tt5.getText().isEmpty() || tt6.getText().isEmpty() ||  
tt7.getText().isEmpty() || tt8.getText().isEmpty() || tt9.getText().isEmpty() ||  
tt10.getText().isEmpty()) {  
    JOptionPane.showMessageDialog(null, "Please fill all the fields.");  
    return;  
}
```

```
int teacherId = Integer.parseInt(tt1.getText());  
// Check if teacherId already exists  
boolean idExists = false;  
for (Teacher tutor : tlist) {  
    if (tutor.getTeacherId() == teacherId) {  
        idExists = true;  
        break;  
    }  
}  
if (idExists) {  
    JOptionPane.showMessageDialog(null, "Teacher ID already exists. Please  
enter a new ID.");  
    return; // Exit the method  
}
```

```
String teacherName = tt2.getText();  
String address = tt3.getText();  
String workingType = tt4.getText();  
String employmentStatus = tt5.getText();  
int workingHours = Integer.parseInt(tt6.getText());  
int salary = Integer.parseInt(tt7.getText());  
String specialization = tt8.getText();
```



```
String academicQualifications = tt9.getText();
int performanceIndex = Integer.parseInt(tt10.getText());

// Create a new Tutor object
Tutor tutor1 = new Tutor(teacherId, teacherName, address, workingType,
employmentStatus, workingHours, salary, specialization, academicQualifications,
performanceIndex);

// Add the new tutor to the list
tlist.add(tutor1);

JOptionPane.showMessageDialog(null, "Tutor added successfully!");
} catch (NumberFormatException ex) {
    JOptionPane.showMessageDialog(null, "Invalid input. Please enter valid integer
values for working hours, salary, and performance index.");
}
}
});

JButton tb2 = new JButton("clear");
tb2.setBounds(1050,120,120,30);
tb2.setBackground(Color.ORANGE);
tb2.setFont(new Font("Times new Roman", Font.PLAIN, 14));
tb2.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int choice = JOptionPane.showConfirmDialog(null, "Do you want to clear?");
        if (choice == JOptionPane.YES_OPTION){
```

```
        tt1.setText("");
        tt2.setText("");
        tt3.setText("");
        tt4.setText("");
        tt5.setText("");
        tt6.setText("");
        tt7.setText("");
        tt8.setText("");
        tt9.setText("");
        tt10.setText("");
        tns.setText("");
        tnp.setText("");
        tt13.setText("");
        tt14.setText("");

    }
    else{

    }

    }

});
```

```
JButton tb3 = new JButton("Salary");
tb3.setBounds(200, 280, 150, 30);
```

```

tb3.setBackground(Color.MAGENTA);
tb3.setFont(new Font("Times new Roman", Font.PLAIN, 14));
tb3.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
        for (Teacher t : tlist){
            try{
                int newSalary = Integer.parseInt(tns.getText());
                int newPerformanceIndex = Integer.parseInt(tnp.getText());
                Tutor T1 = (Tutor) t;
                T1.setSalary(newSalary,newPerformanceIndex);
                JOptionPane.showMessageDialog(null, "!!!Thank You!!! ");
            } catch(Exception e){
                JOptionPane.showMessageDialog(null, "Enter Correct
Information");
            }
        }
    }
});

```

```

JButton tb4 = new JButton("Display");
tb4.setBounds(500,280,150,30);

```

```
tb4.setBackground(Color.GREEN);
tb4.setFont(new Font("Times new Roman", Font.PLAIN, 14));
tb4.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
        for(Teacher t : tlist)
        {
            if(t instanceof Tutor)
            {
                t.display();
            }
        }
    }
});
```

```
JButton tb5 = new JButton("Remove");
tb5.setBounds(1050, 280, 150, 30);
tb5.setBackground(Color.RED);
tb5.setFont(new Font("Times new Roman", Font.PLAIN, 14));

tb5.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
        // Get the teacher ID entered by the user
        int teacherId = Integer.parseInt(tt14.getText());
```

```
// Check if the teacher ID exists in the ArrayList
boolean removed = false;
for (Teacher t : tlist) {
    if (t.getTeacherId() == teacherId) {
        Tutor tt1 = (Tutor) t;
        tt1.removeTutor();
        removed = true;
        break; // Exit loop since we found and removed the tutor
    }
}

// If tutor was removed, notify the user
if (removed) {
    JOptionPane.showMessageDialog(null, "Tutor with teacher ID " + tt14.getText()
+ " removed successfully.");
} else {
    JOptionPane.showMessageDialog(null, "Tutor with teacher ID " + tt14.getText()
+ " not found.");
}
});
```

```
p2.add(tt1);
p2.add(tl1);
p2.add(tl2);
p2.add(tt2);
p2.add(tl3);
```

```
p2.add(tt3);
p2.add(tl4);
p2.add(tt4);
p2.add(tl5);
p2.add(tt5);
p2.add(tl6);
p2.add(tt6);
p2.add(tl7);
p2.add(tt7);
p2.add(tl8);
p2.add(tt8);
p2.add(tl9);
p2.add(tt9);
p2.add(tb1);
p2.add(tb2);
p2.add(heading1);
p2.add(tl10);
p2.add(tt10);
p2.add(tb3);
p2.add(tb4);
p2.add(tb5);
p2.add(sh2);
p2.add(tns);
p2.add(tnp);
p2.add(tl11);
p2.add(tl12);
p2.add(tl13);
p2.add(tt13);
```

```
p2.add(tl14);
```

```
p2.add(tt14);
```

```
p2.add(sh3);
```

```
}
```

```
}
```