



CS4001NI Programming

30% Individual Coursework

2023-24 Autumn

Student Name: SAMARPAN KHADKA

London Met ID: 23047551

College ID: np01nt4a230214

Group: N7

Assignment Due Date: July 28, 2024

Assignment Submission Date: July 27, 2024

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

Table of contents

Contents

Table of contents	2
1 Introduction	4
1.1 Tools Used	5
1.2 TOOLS USED	5
2.Class Diagram.....	7
2.1 Teacher Class Diagram	7
2.2 Lecturer Class Diagram	8
2.3 Tutor Class Diagram	8
2.4 Inheritance Class Diagram.....	9
3.Pseudocode	10
4. Description of all the Methods and Buttons.	14
4.1Method.....	14
4.2Buttons	14
5.Testing	15
5.1 Testing 1	15
5.2Testing 2	16
5.2Testing 3.....	20
6.Error	22
6.1 Syntax Error.....	22
6.2 Sematic Error	23
6.3 Logical Error	25
8. Appendix of code	28

Table of figures

Figure 1teacher class diagram.....	7
Figure 2lecturer class diagram.....	8
Figure 3 Figure 3 tutor class diagram	8
Figure 4inheritance class diagram.....	9
Figure 5 opening through command prompt.....	15
Figure 6 opened through cmd.....	15
Figure 7 error detection	22
Figure 8 error correction.....	23
Figure 9semantic error detection.	23
Figure 10 error correction semantics.	24
Figure 11 logical error detection	25
Figure 12 logical error correction.....	26

1 Introduction

This coursework is an individual project which is about making a GUI for a system that stores details of teachers in an Array List. I should make a class which contains main method. This project helped me to know about array List, Abstract Window toolkit (AWT), Swing, Event Handling, Exception Handling etc.

In the given coursework have created a Window-based application where the information of teacher is stored. This Coursework was same of previous time where 3 classes were made. Teacher was the main class whereas tutor and lecturer were sub-class. Basically, the package or coursework contains a class called Student GUI which is basically used to create a GUI and an array list of Teacher class. This is a more user-friendly application which makes easy to provide information of teacher.

1.1 Tools Used

1.2 TOOLS USED

During this coursework we used various software like BlueJ, MS- Word, Draw.io.

BlueJ



BlueJ is a development environment that allows you to develop Java programs quickly and easily. It was developed mainly for educational purposes, but also suitable for small-scale software development. It was developed by the University of Kent and Deakin University to teach object orientation in a Java development environment.

MS-WORD



Figure 2: picture of MS-word

(<https://office.com>, 2019`)

Microsoft Word is a widely used commercial word processor designed by Microsoft. It's a component of the Microsoft Office suite of productivity software but can also be purchased as a stand-alone product. Microsoft Word can help users create a variety of different types of documents.

Draw.io



Draw.io is a proprietary software designed by Seibert Media for creating diagrams and charts. It's a technology stack for building diagramming applications and is the world's most widely used browser-based end-user diagramming software. It offers a large selection of shapes and hundreds of visual elements to make your diagram or chart unique.

Figure 3: Picture of Draw.io (-, 2020)

2. Class Diagram

Class Diagram is a kind of blueprint for understanding which is mainly used in OOP. Class diagram is visual representation which shows the relationship of class, system their attributes and methods. The convention of class diagram are listed down:

Symbol	Meaning
+	Represent public access modifier
-	Represent private access modifier
#	Represent protected access modifier
<<Constructor>>	Represents constructor of Class

2.1 Teacher Class Diagram

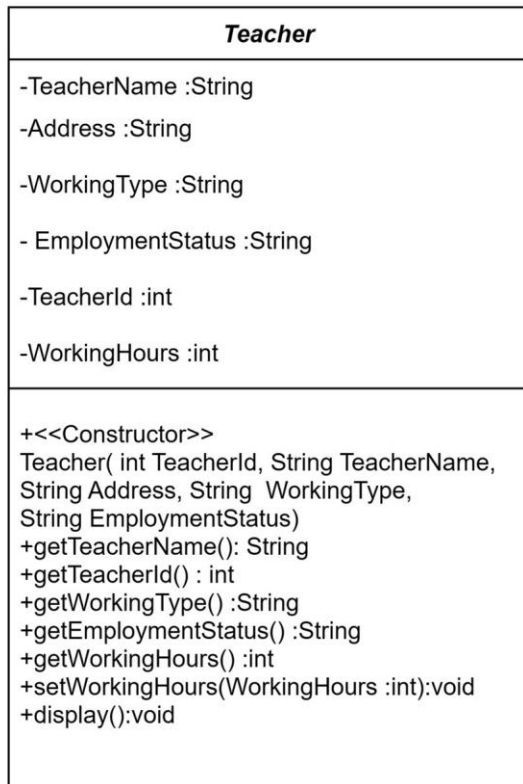


Figure 1 teacher class diagram

2.2 Lecturer Class Diagram

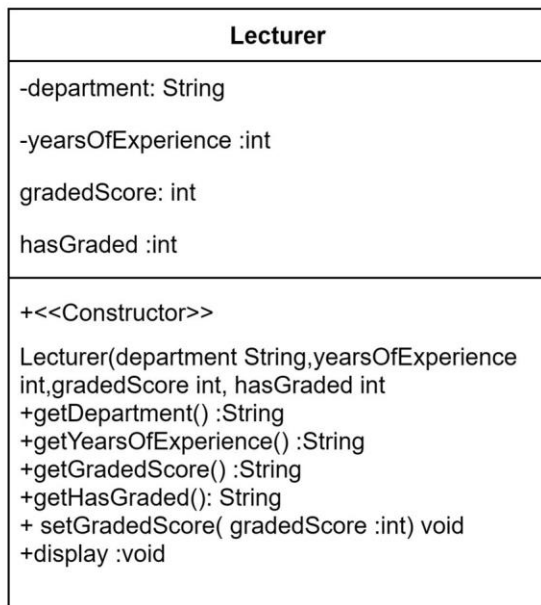


Figure 2lecturer class diagram

2.3 Tutor Class Diagram

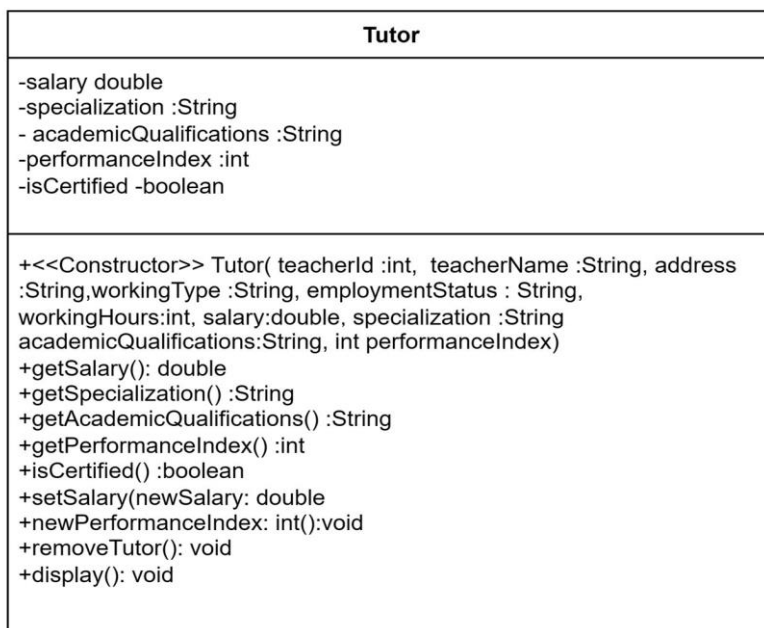


Figure 3 Figure 3 tutor class diagram

2.4 Inheritance Class Diagram.

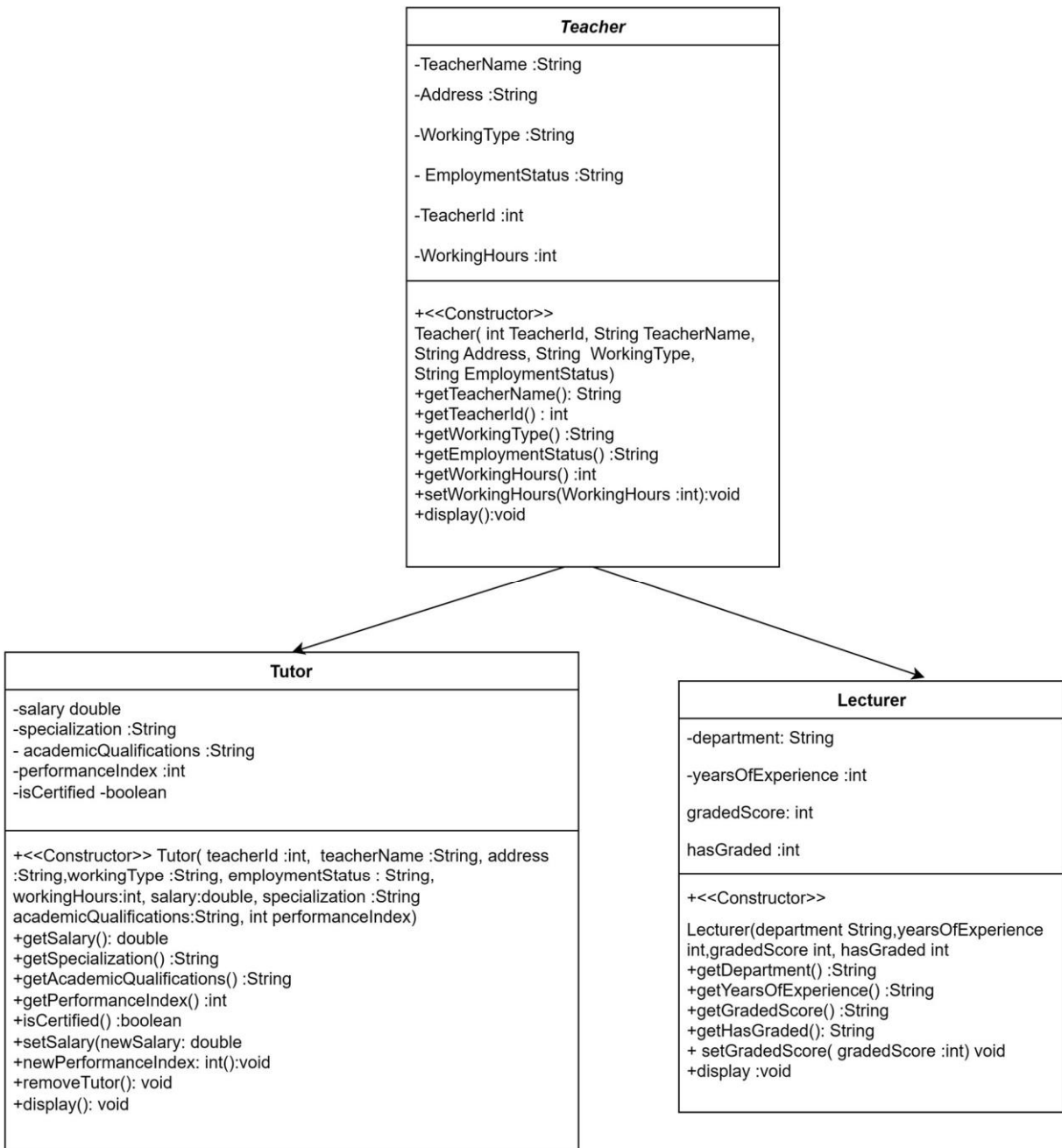


Figure 4 inheritance class diagram

3.Pseudocode

3.1Pseudocode for teachers GUI.

Class TeacherGUI:

// ArrayList to store information about teachers

ArrayList TeacherInfo

// Method to create and display the GUI for adding lecturers

Function sam():

// Create the main JFrame

Create JFrame "TeacherGUI"

Set background color to a specific color

Set size of the JFrame

Set layout to null

// Create a JPanel for GUI components

Create JPanel "GUIPanel"

Set layout to null

Set background color to the same color as JFrame

Set bounds for the JPanel

// Make the JFrame visible

Set JFrame visibility to true

// Add title label

Create JLabel "Teacher GUI"

Set bounds and font for the JLabel

Add the JLabel to the JFrame

// Add section heading label for adding lecturer

Create JLabel "ADDING LECTURER"

Set bounds and font for the JLabel

Add the JLabel to the JFrame

// Add JLabels and JTextFields for lecturer information

Create JLabels and JTextFields for:

Teacher Name

Teacher ID

Teacher Address

Graded Score

Working Type

Employment Status

Years of Experience

Department

Set bounds for each JLabel and JTextField

Add JLabels and JTextFields to the JFrame

// Add JButtons for actions (Add, Grade, Clear, Display, Change to Tutor)

Create JButtons for each action:

Add to Lecturer

Grade Assignment

Clear

Display

Change to Tutor

Set bounds for each JButton

Add action listeners for each JButton

// Method to switch to the tutor interface

Function m():

 // Create JFrame for tutor interface

 Create JFrame "Tutor Record"

 Set background color to a specific color

 Set size of the JFrame

 Set layout to null

 // Make the JFrame visible

 Set JFrame visibility to true

 // Add section heading label for adding tutor

 Create JLabel "ADDING TUTOR"

 Set bounds and font for the JLabel

 Add the JLabel to the JFrame

 // Add JLabels and JTextFields for tutor information

 Create JLabels and JTextFields for:

 Teacher Name

 Teacher ID

 Address

 Working Type

 Employment Status

 Working Hours

 Salary

 Specialization

 Academic Qualification

 Performance Index

Set bounds for each JLabel and JTextField

Add JLabels and JTextFields to the JFrame

// Add JButtons for actions (Clear, Display, Set Salary, Remove, Add, Change to Lecturer)

Create JButtons for each action:

Clear

Display

Set Salary

Remove Tutor

Add Tutor

Change to Lecturer

Set bounds for each JButton

Add action listeners for each JButton

// Main function to create an instance of TeacherGUI and display the lecturer interface

Main function:

// Create TeacherGUI instance tGUI

Create TeacherGUI instance "tGUI"

Call sam() method on tGUI to display lecturer interface

4. Description of all the Methods and Buttons.

4.1 Method

Method is code that helps to perform specific tasks when it is called. Method must be declared within the class and followed by parentheses. Method helps to reuse the code without repeating the same code. Method can be called both by publicly and privately.

Syntax of Method.

```
<Access Modifier><Method Type>(Static)<Return Type><Method Name>(parameters)
{
//Statements//
}
```

S. N	Method Name	Description
1.	Main (String [] args)	Main class and which is used to call the constructor
2	action Performed (Action Event)	Used to perform when an event occurs like clicking a button.

Table 1 method used in program

4.2 Buttons

Buttons used in my coursework is listed down below:

Button	Description
Add to Lecturer (B1)	Adds a new lecturer to the system based on the information entered in the text fields.
Grade Assignment (B2)	Grades assignments based on certain criteria such as years of experience and department
Clear (B8)	Clears all the text fields in both the lecturer and tutor sections of the GUI
Display (B3)	Displays information about the entered lecturer or tutor.
Set Salary (B6)	Sets the salary and performance index of a tutor.
Remove Tutor (B7)	Removes a tutor from the system based on their ID.

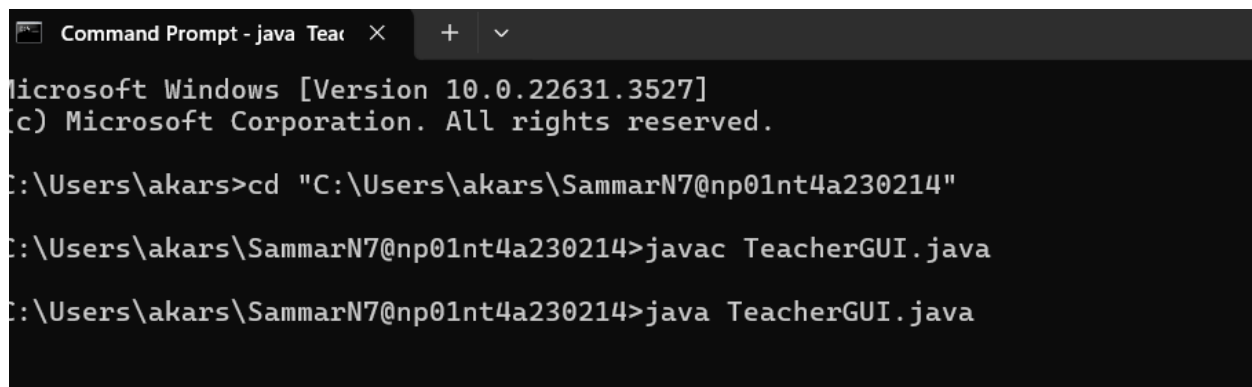
Table 2 buttons used in program.

5. Testing

5.1 Testing 1

Objective	Program should be compiled and run using the cmd command prompt.
Action	File location is directed in cmd prompt. Java file is compiled using the command: Javac TeacherGUI.java command is used for running the program.
Expected Result	GUI should open using cmd prompt.
Actual Result	GUI opened through cmd prompt.
Conclusion	Test was carried successfully.

Table 3testing 1



```
Microsoft Windows [Version 10.0.22631.3527]
(c) Microsoft Corporation. All rights reserved.

C:\Users\akars>cd "C:\Users\akars\SammarN7@np01nt4a230214"

C:\Users\akars\SammarN7@np01nt4a230214>javac TeacherGUI.java

C:\Users\akars\SammarN7@np01nt4a230214>java TeacherGUI.java
```

Figure 5 opening through command prompt

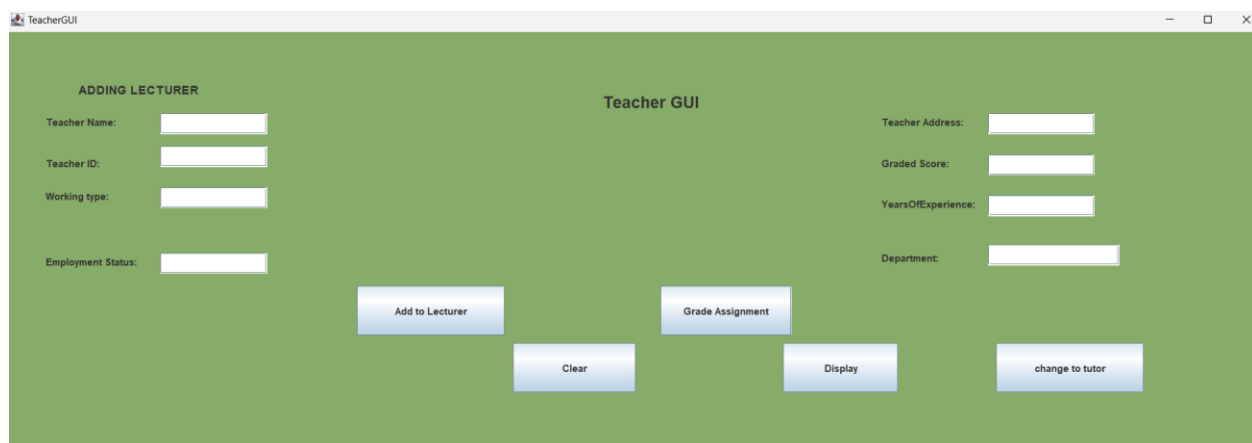


Figure 6 opened through cmd.

5.2 Testing 2

Objective	Filling the given information. Add the lecturer. Add the Tutor Grade Assignments from lecture. Set the salary. Remove the salary
Action	Adding the lecturer. Adding the Tutor Grading Assignments from lecture. Setting the salary. Removing the salary
Expected Result	Lecturer should be added. Tutor should be added. Assignments Should be graded. Salary should be set. Salary should be removed.
Actual Result	Lecturer is added. Tutor is added. Assignment is graded. Salary is set. Salary is removed
Conclusion	Test is successfully runned.

Table 4 testing 2

The image displays two sequential screenshots of a software application titled "Teacher GUI".

Top Screenshot: The interface features a green background. On the left, under the heading "ADDING LECTURER", there are input fields for "Teacher Name" (containing "sammar"), "Teacher ID" (containing "1"), "Working type" (containing "full"), and "Employment Status" (containing "ceo"). On the right, there are input fields for "Teacher Address" (containing "kdm"), "Graded Score" (containing "90"), "YearsOfExperience" (containing "10"), and "Department" (containing "it"). In the center, there are buttons for "Add to Lecturer", "Graded Assignment", "Display", and "change to tutor". A modal dialog box titled "Message" is open in the center, displaying the text "Lecturer is added successfully" with an "OK" button.

Bottom Screenshot: This screenshot shows the same interface after an action. The modal dialog box now displays "Graded Assignment : A" with an "OK" button. The other elements of the interface, including the input fields and buttons, remain the same.

TeacherGUI

Teacher GUI

ADDING LECTURER

Teacher Name:

Teacher ID:

Working type:

Employment Status:

Teacher Address:

Graded Score:

YearsOfExperience:

Department:

Your input successful.

Teacher Name:sammar
Teacher ID:1
Address:ktm
Working Type:full
EmploymentStatus:ceo
YearsOfExperience:10
Department:

tutor record

ADDING TUTOR

Teacher Name:

Teacher ID:

Employment Status:

Working Hours:

Academic qualification:

Address:

Working Type:

Salary:

Specialization:

Performance Index:

Message

Tutor added Successfully

tutor record

ADDING TUTOR

Teacher Name:

Teacher ID:

Employment Status:

Working Hours:

Academic qualification:

Address:

Working Type:

Salary:

Specialization:

Performance Index:

Message

Salary is updated successfully.

tutor record

ADDING TUTOR

Teacher Name: sammar

Teacher ID: 1

Employment Status: ceo

Working Hours: 50

Academic qualification: bsc

Address: ktm

Working Type: full

Salary: 500000

Specialization: networking

Performance Index: 9

Clear

Remove Tutor

Add Tutor

Set Salary

change to tutor

Change to lecturer

Tutor Information successfull

Teacher Name: sammar

Teacher ID: 1

Address: ktm

Working Type: full

Employment Status: ceo

Working Hours: 50

Salary: 500000

Specialization: networking

Academic Qualification: bsc

Performance Index: 9.0

OK

tutor record

ADDING TUTOR

Teacher Name: sammar

Teacher ID: 1

Employment Status: ceo

Working Hours: 50

Academic qualification: bsc

Clear

Remove Tutor

Add Tutor

Set Salary

Message

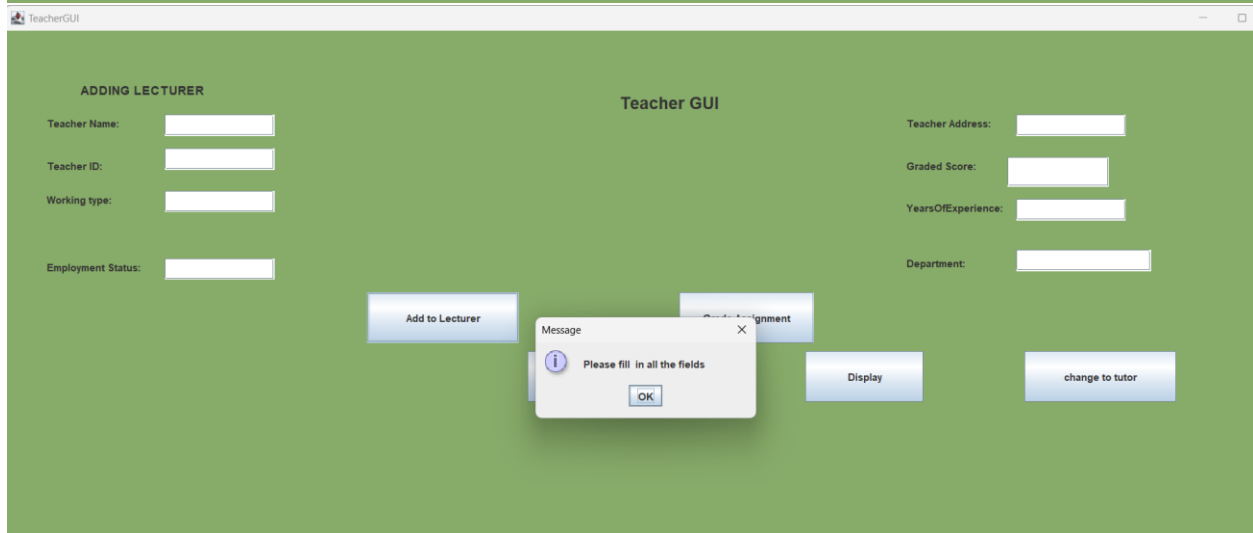
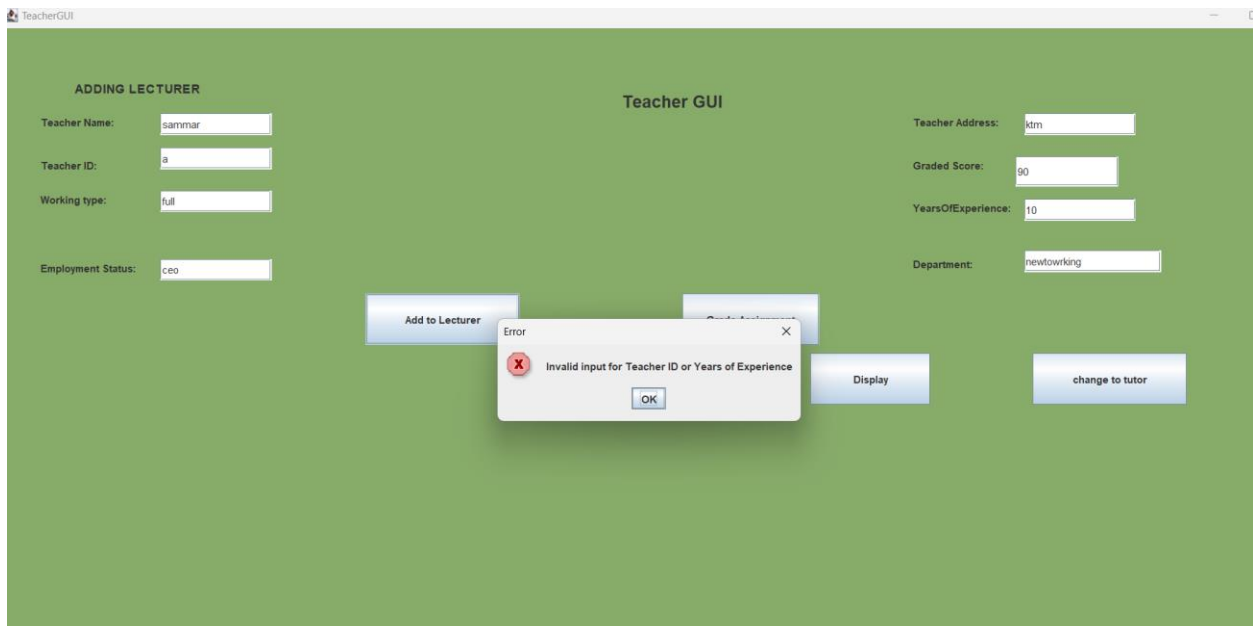
1Removed tutor has been successfully.

OK

5.2 Testing 3

Objective	Error dialogue boxes appears when unsuitable values are entered in Teacher ID
Action	String Value is given to Teacher ID for checking the dialogue box.
Expected Result	Error message should pop up in dialogue box.
Actual Result	Errors message popped up successfully.
Conclusion	Test was successful.

Table 5 Testing 3



ADDING TUTOR

Teacher Name:	<input type="text" value="sammar"/>	Address:	<input type="text" value="ldm"/>
Teacher ID:	<input type="text" value="a"/>	Working Type:	<input type="text" value="full"/>
Employment Status:	<input type="text" value="ceo"/>	Salary:	<input type="text" value="500000"/>
Working Hours:	<input type="text" value="50"/>	Specialization:	<input type="text" value="it"/>
Academic qualification:	<input type="text" value="bsc"/>	Performance Index:	<input type="text" value="9"/>

Clear

Remove Tutor

Message

Invalid input format. Please enter a valid number for teacher ID.

OK

Set Salary

change to tutor

Add Tutor

Change to lecturer

ADDING TUTOR

Teacher Name:	<input type="text"/>	Address:	<input type="text"/>
Teacher ID:	<input type="text"/>	Working Type:	<input type="text"/>
Employment Status:	<input type="text"/>	Salary:	<input type="text"/>
Working Hours:	<input type="text"/>	Specialization:	<input type="text"/>
Academic qualification:	<input type="text"/>	Performance Index:	<input type="text"/>

Clear

Remove Tutor

Message

Please Fill all the fields

OK

Set Salary

change to tutor

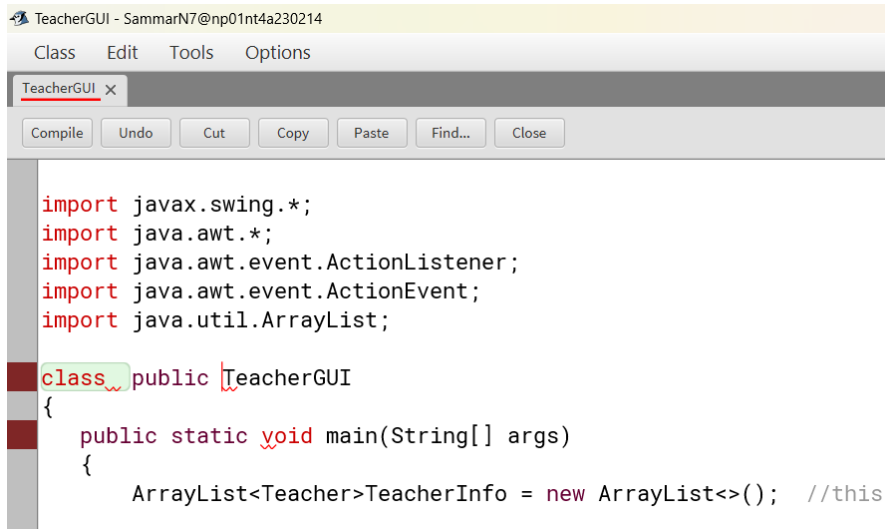
Add Tutor

Change to lecturer

6.Error

6.1 Syntax Error

Java syntax error is the mistakes to made done a programmer in implementing the grammar of the Java programming language. It doesn't cover mistakes in logic of the program itself.



The screenshot shows a Java IDE window titled "TeacherGUI - SammarN7@np01nt4a230214". The menu bar includes "Class", "Edit", "Tools", and "Options". The toolbar has buttons for "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". The code editor displays the following Java code:

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

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

The IDE highlights the word "class" in green and "public" in red, indicating a syntax error. The error is due to the incorrect placement of the access modifier "public" before the keyword "class".

Figure 7 error detection

Error detection: I came to know the reason behind my error was due to incorrect arrangements of access modifier "public" and keyword "Class".

Correction: I declared class in proper syntax after the problem was solved. Access modifier such as (public, private) should be first and class should be after modifier.

```

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

public class TeacherGUI
{
    public static void main(String[] args)
    {
        ArrayList<Teacher>TeacherInfo = new
        JFrame jf = new JFrame("TeacherGUI")
        Color back = Color.decode("#87ab69")
        jf.getContentPane().setBackground(back)
        jf.setSize(1800,1800);
        jf.setLayout(null);
        JPanel GUIPanel = new JPanel();
        GUIPanel.setLayout(null);
        GUIPanel.setBackground(back);
        GUIPanel.setBounds(1800,950,1800,180

        JLabel a1 = new JLabel("Teacher GUI");
        a1.setBounds(730, 45, 200, 80);
        a1.setFont(new Font("Baguet Script"
        jf.add(a1);

        JLabel l1 = new JLabel("ADDING LECT

```

Class compiled - no syntax errors

Figure 8 error correction

6.2 Sematic Error

Semantic error is one in which the code uses a variable that isn't initialized or run properly. While compiling we can find the semantic error in Code.

```

int Id = Integer.parseInt(t12.getText());
int salary = Integer.parseInt(t17.getText());
int workingHours = Integer.parseInt(t16.getText());
int employmentStatus = t15.getText();
int performanceIndex = Integer.parseInt(t20.getText());

Tutor newTutor = new Tutor(Id, t11.getText(), t12.getText(), t14.g

```

Figure 9semantic error detection.

Error detection: I came to know that my error was that in string value I wrote int which showed me error.

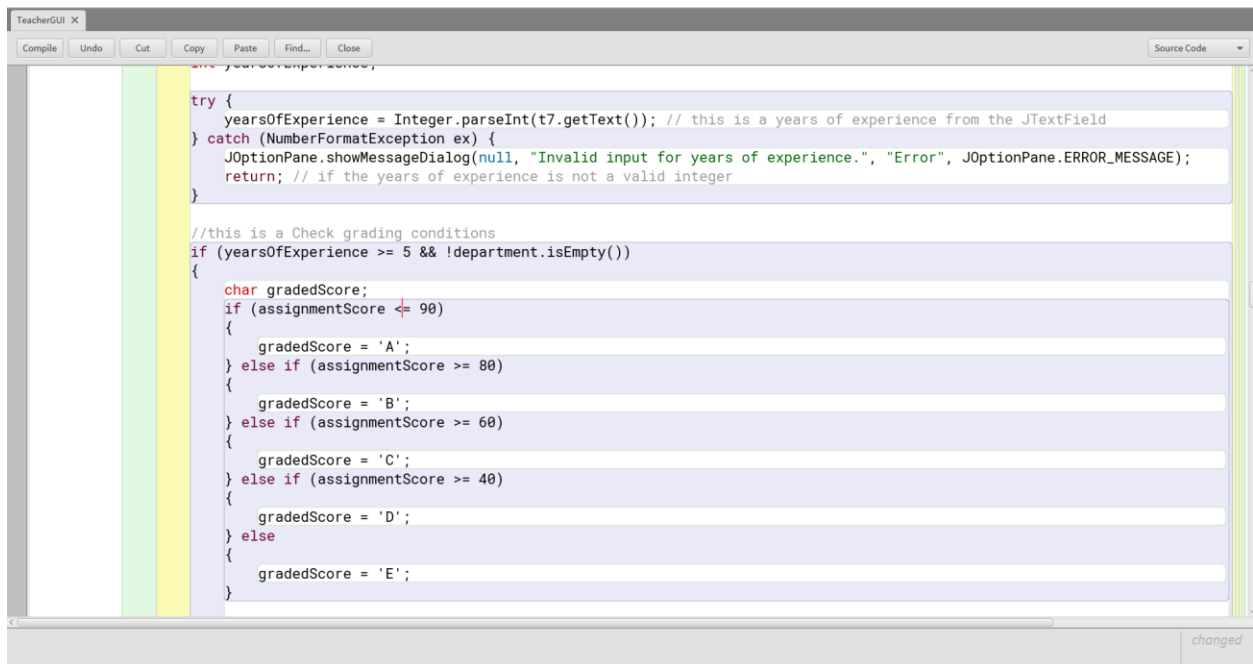
Error Correction: I solved this semantic by replacing the int with String value. As the employment status as string value. After that the problem was solved.

```
int Id = Integer.parseInt(t12.getText());  
int salary = Integer.parseInt(t17.getText());  
int workingHours = Integer.parseInt(t16.getText());  
String employmentStatus = t15.getText();  
int performanceIndex = Integer.parseInt(t20.getText());
```

Figure 10 error correction semantics.

6.3 Logical Error

Logical Error are those errors that are hard to find because the error is not shown in the program, but the output is not of our choice.

A screenshot of a Java IDE window titled "TeacherGUI X". The window has a menu bar with "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". The main area displays Java code. A yellow vertical bar highlights a section of the code. The code includes a try-catch block for parsing years of experience, followed by a comment "//this is a Check grading conditions" and an if-statement. Inside the if-statement, there is a series of if-else if blocks that assign a gradedScore based on assignmentScore. The error is in the first condition: "if (assignmentScore <= 90)". The IDE's status bar at the bottom right shows the word "changed".

```
try {
    yearsOfExperience = Integer.parseInt(t7.getText()); // this is a years of experience from the JTextField
} catch (NumberFormatException ex) {
    JOptionPane.showMessageDialog(null, "Invalid input for years of experience.", "Error", JOptionPane.ERROR_MESSAGE);
    return; // if the years of experience is not a valid integer
}

//this is a Check grading conditions
if (yearsOfExperience >= 5 && !department.isEmpty())
{
    char gradedScore;
    if (assignmentScore <= 90)
    {
        gradedScore = 'A';
    } else if (assignmentScore >= 80)
    {
        gradedScore = 'B';
    } else if (assignmentScore >= 60)
    {
        gradedScore = 'C';
    } else if (assignmentScore >= 40)
    {
        gradedScore = 'D';
    } else
    {
        gradedScore = 'E';
    }
}
```

Figure 11 logical error detection

Error detection: I came to know that my error was that in assignment score I mistakenly typed lower than sign (<) which showed me error.

Error Correction: I solved this logical error by replacing (<) sign with (>) sign. After that the problem was solved.

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

//this is a Check grading conditions
if (yearsOfExperience >= 5 && !department.isEmpty())
{
    char gradedScore;
    if (assignmentScore >= 90)
    {
        gradedScore = 'A';
    } else if (assignmentScore >= 80)
    {
        gradedScore = 'B';
    } else if (assignmentScore >= 60)
    {
        gradedScore = 'C';
    } else if (assignmentScore >= 40)
    {
        gradedScore = 'D';
    } else
    {
        gradedScore = 'E';
    }
}
```

Figure 12 logical error correction

7.Conclusion

From this coursework, I learned the main concepts of making Window based application with proper functionality. This course work taught me about the idea of Swing, Abstract Window Toolkit (AWT), Exception and Event Handling, Object Casting as well gained the concept of OOP. I faced a lot of difficulties while attempting my coursework. I learned about JAVA in depth. I learned to make tables from Draw. Io as well. This coursework helped me to boost my confidence towards JAVA. In upcoming days, I can do various projects which are carried through JAVA. I got to know the idea on building desktop applications, database application.

As an individual undertaking this coursework, I got an opportunity to test my skills, ideas, and understanding of programming skills. I faced a lot of logical and syntax errors while developing an application for a system that stores the details of Teacher (Tutor and Lecturer). Before this coursework I only knew the basic of OOP, but after this project I came to know how to use it in real- world problems.

I like to give special Credit to my Teacher (Astha Sharma Mam) who made me capable of doing this project. I like to give special thanks to my colleagues who helped to solve various problems that had arrived while doing this project.

8. Appendix of code

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

class TeacherGUI {

    ArrayList<Teacher> TeacherInfo = new ArrayList<> ();

    public void sam() {

        JFrame jf = new JFrame("TeacherGUI");
        Color back = Color.decode("#87ab69"); // color code
        jf.getContentPane().setBackground(back);
        jf.setSize(1800,1800);
        jf.setLayout(null);
        JPanel GUIPanel = new JPanel();
        GUIPanel.setLayout(null);
        GUIPanel.setBackground(back);
        GUIPanel.setBounds(1800,950,1800,1800);

        JLabel a1 = new JLabel("Teacher GUI");
        a1.setBounds(730, 45, 200, 80);
        a1.setFont(new Font("Baguet Script", Font.BOLD, 20));
        jf.add(a1);
```

```
JLabel l1 = new JLabel("ADDING LECTURER");  
l1.setBounds(90,45,150,50);  
l1.setFont(new Font("Baguet Script", Font.BOLD, 15));  
jf.add(l1);
```

```
JLabel l2 = new JLabel("Teacher Name:");  
l2.setBounds(50, 90, 100, 40);  
jf.add(l2);
```

```
JTextField t2 = new JTextField();  
t2.setBounds(190, 100, 130, 25);  
jf.add(t2);
```

```
JLabel l3 = new JLabel("Teacher ID:");  
l3.setBounds(50, 140, 100, 40);  
jf.add(l3);
```

```
JTextField t3 = new JTextField();  
t3.setBounds(190, 140, 130, 25);  
jf.add(t3);
```

```
JLabel l8 = new JLabel("Graded Score:");  
l8.setBounds(1070,140,100,40);  
jf.add(l8);
```

```
JTextField t8 = new JTextField();  
t8.setBounds(1190,150,120,35);
```

```
jf.add(t8);
```

```
JLabel l4 = new JLabel("Teacher Address:");  
l4.setBounds(1070, 90, 100, 40);  
jf.add(l4);
```

```
JTextField t4 = new JTextField();  
t4.setBounds(1200, 100, 130, 25);  
jf.add(t4);
```

```
JLabel l5 = new JLabel("Working type:");  
l5.setBounds(50,180,100,40);  
jf.add(l5);
```

```
JTextField t5 = new JTextField();  
t5.setBounds(190,190,130,25);  
jf.add(t5);
```

```
JLabel l6 = new JLabel("Employment Status:");  
l6.setBounds(50,260,160,40);  
jf.add(l6);
```

```
JTextField t6 = new JTextField();  
t6.setBounds(190,270,130,25);  
jf.add(t6);
```

```
JLabel l7 = new JLabel("YearsOfExperience:");  
l7.setBounds(1070,190,140,40);  
jf.add(l7);
```

```
JTextField t7 = new JTextField();  
t7.setBounds(1200,200,130,25);  
jf.add(t7);
```

```
JLabel l9 = new JLabel("Department:");  
l9.setBounds(1070,250,80,50);  
jf.add(l9);
```

```
JTextField t9 = new JTextField();  
t9.setBounds(1200,260,160,25);  
jf.add(t9);
```

```
JButton B1 = new JButton("Add to Lecturer");  
B1.setBounds(430,310,180,60);  
jf.add(B1);
```

```
JButton B2 = new JButton("Grade Assignment");  
B2.setBounds(800,310,160,60);  
jf.add(B2);
```

```
JButton B8 = new JButton("Clear");
```

```
B8.setBounds(620,380,150,60);  
jf.add(B8);
```

```
JButton B3 = new JButton("Display");  
B3.setBounds(950,380,140,60);  
jf.add(B3);
```

```
JButton B51 = new JButton("change to tutor");  
B51.setBounds(1210,380,180,60);  
jf.add(B51);  
jf.setVisible(true);
```

```
B1.addActionListener(new ActionListener() //adding lecturer action listener  
{  
    public void actionPerformed(ActionEvent ae)  
    {  
        if (t2.getText().isEmpty() || t3.getText().isEmpty() || t4.getText().isEmpty() ||  
t7.getText().isEmpty())  
        {  
            JOptionPane.showMessageDialog(null, "Please fill in all the fields");  
        } else  
        {  
            try  
            {  
                int tld = Integer.parseInt(t3.getText());  
                int years = Integer.parseInt(t7.getText());  
  
                Lecturer newLecturer = new Lecturer(  
                    tld,
```



```

        t2.getText(),
        t4.getText(),
        t5.getText(),
        t6.getText(),
        t9.getText(),
        years
    );

```

```

        TeacherInfo.add(newLecturer);
        JOptionPane.showMessageDialog(null, "Lecturer is added
successfully"); // a message dialog box
    } catch (NumberFormatException ex) //this is a exception
    {
        JOptionPane.showMessageDialog(null, "Invalid input for Teacher ID or
Years of Experience", "Error", JOptionPane.ERROR_MESSAGE);
    } catch (Exception ex)
    {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(null, "An error occurred while
adding the lecturer", "Error", JOptionPane.ERROR_MESSAGE);
    }
}
}
});

```

B3.addActionListener(new ActionListener() //this is a display action listener in lecturer section

```

{
    public void actionPerformed(ActionEvent ae)
    {

```

```

for(Teacher teacher : TeacherInfo)
{
    if(teacher instanceof Lecturer)
    {
        Lecturer lecturer = (Lecturer) teacher;
        lecturer.display();

        String TeacherName = t2.getText();
        String TeacherID = t3.getText();
        String Address = t4.getText();
        String WorkingType = t5.getText();
        String EmploymentStatus = t6.getText();
        String YearsOfExperience = t7.getText();
        String Department = t9.getText();

        String Message = "Teaher Name:" + TeacherName + "\n" + "Teacher
ID:" + TeacherID + "\n" + "Address:" + Address + "\n" +
        " Working Type:" + WorkingType + "\n" + "EmploymentStatus:" +
EmploymentStatus + "\n" + "YearsOfExperience:" + YearsOfExperience + "\n"
        + "Department:" + "\n";

        JOptionPane.showMessageDialog(null, Message, "Your input
successfull.", JOptionPane.INFORMATION_MESSAGE);

        JOptionPane.showMessageDialog(null,"Added Lecturer
Successfully"); // message dialog box

    }
}
});

```

B2.addActionListener(new ActionListener())// this is a grade assignment action listener function of button

```
{
    public void actionPerformed(ActionEvent ae)
    {
        // this is a variables assignmentScore and studentDepartment
        int assignmentScore;
        String department = t9.getText(); //this is a department is entered in a
JTextField

        try
        {
            assignmentScore = Integer.parseInt(t8.getText()); // this is a assignment
score from the JTextField
        } catch (NumberFormatException ex)
        {
            JOptionPane.showMessageDialog(null, "Invalid input for assignment
score.", "Error", JOptionPane.ERROR_MESSAGE);
            return; // if the assignment score is not a valid integer
        }

        // this is a yearsOfExperience is defined and parsed
        int yearsOfExperience;

        try {
            yearsOfExperience = Integer.parseInt(t7.getText()); // this is a years of
experience from the JTextField
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(null, "Invalid input for years of
experience.", "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

```
        return; // if the years of experience is not a valid integer
    }
```

```
//this is a Check grading conditions
```

```
if (yearsOfExperience >= 5 && !department.isEmpty())
```

```
{
    char gradedScore;
    if (assignmentScore >= 90)
    {
        gradedScore = 'A';
    } else if (assignmentScore >= 80)
    {
        gradedScore = 'B';
    } else if (assignmentScore >= 60)
    {
        gradedScore = 'C';
    } else if (assignmentScore >= 40)
    {
        gradedScore = 'D';
    } else
    {
        gradedScore = 'E';
    }
}
```

```
//this is a Display the graded score in a dialog box
```

```
JOptionPane.showMessageDialog(null, "Graded Assignment : " +
gradedScore, "Graded Assignment", JOptionPane.INFORMATION_MESSAGE);
```

```
} else
```

```
{
```

```

        //this is a Display error message if grading conditions are not met
        JOptionPane.showMessageDialog(null, "Lecturer does not meet criteria
for grading.", "Error", JOptionPane.ERROR_MESSAGE);
    }
}
});

```

B8.addActionListener(new ActionListener())//this is a clear action listener which is clear all jtextfield

```

{
    public void actionPerformed(ActionEvent ae)
    {
        t2.setText("");
        t3.setText("");
        t4.setText("");
        t5.setText("");
        t6.setText("");
        t7.setText("");
        t8.setText("");
        t9.setText("");

        JOptionPane.showMessageDialog(null,"Cleared!"); //this is a joptionpane
which is show message dialog box
    }
}
});

```

B51.addActionListener(new ActionListener())

```

{
    public void actionPerformed(ActionEvent sth) {
        TeacherGUI hi=new TeacherGUI();
    }
}

```

```
        hi.m();  
    }  
});
```

```
}
```

```
//calling tutorclass
```

```
public void m() {
```

```
    JFrame jf = new JFrame ("tutor record");  
    Color back = Color.decode("#87ab69"); // color code  
    jf.getContentPane().setBackground(back);  
    jf.setSize(1800,1800);  
    jf.setLayout(null);  
    jf.setVisible (true);
```

```
    JLabel l10 = new JLabel("ADDING TUTOR");  
    l10.setBounds(15,15,250,100);  
    l10.setFont(new Font("ADLaM Display", Font.BOLD, 25));  
    jf.add(l10);
```

```
    JLabel l11 = new JLabel("Teacher Name:");  
    l11.setBounds(50, 90, 100, 40);  
    jf.add(l11);
```

```
    JTextField t11 = new JTextField();
```

```
t11.setBounds(190, 100, 130, 25);  
jf.add(t11);
```

```
JLabel l12 = new JLabel("Teacher ID:");  
l12.setBounds(50,140,100,40);  
jf.add(l12);  
JTextField t12 = new JTextField();  
t12.setBounds(190,140,130,25);  
jf.add(t12);
```

```
JLabel l13 = new JLabel("Address:");  
l13.setBounds(1070,90,100,40);  
jf.add(l13);  
JTextField t13 = new JTextField();  
t13.setBounds(1200,100,130,25);  
jf.add(t13);
```

```
JLabel l14 = new JLabel("Working Type:");  
l14.setBounds(1070,140,100,40);  
jf.add(l14);  
JTextField t14 = new JTextField();  
t14.setBounds(1200,150,130,25);  
jf.add(t14);
```

```
JLabel l15 = new JLabel("Employment Status:");  
l15.setBounds(50,180,130,35);  
jf.add(l15);  
JTextField t15 = new JTextField();  
t15.setBounds(190,190,130,25);  
jf.add(t15);
```

```
JLabel l16 = new JLabel("Working Hours:");
l16.setBounds(50,260,140,40);
jf.add(l16);
JTextField t16 = new JTextField();
t16.setBounds(190,270,130,25);
jf.add(t16);
JLabel l17 = new JLabel("Salary:");
l17.setBounds(1070,190,140,40);
jf.add(l17);
JTextField t17 = new JTextField();
t17.setBounds(1200,200,130,25);
jf.add(t17);
JLabel l18 = new JLabel("Specialization:");
l18.setBounds(1070,250,140,40);
jf.add(l18);
JTextField t18 = new JTextField();
t18.setBounds(1200,260,130,25);
jf.add(t18);
JLabel l19 = new JLabel("Academic qualification:");
l19.setBounds(50,310,140,40);
jf.add(l19);
JTextField t19 = new JTextField();
t19.setBounds(190,310,130,25);
jf.add(t19);
JLabel l20 = new JLabel("Performance Index:");
l20.setBounds(1070,310,140,40);
jf.add(l20);
JTextField t20 = new JTextField();
```



```
t20.setBounds(1200,320,130,25);  
jf.add(t20);
```

```
 JButton B4 = new JButton("Clear");  
B4.setBounds(430,350,160,60);  
jf.add(B4);
```

```
 JButton B5 = new JButton("Display");  
B5.setBounds(710,350,160,60);  
jf.add(B5);
```

```
 JButton B6 = new JButton("Set Salary");  
B6.setBounds(1020,350,160,60);  
jf.add(B6);
```

```
 JButton B7 = new JButton("Remove Tutor");  
B7.setBounds(480,480,160,60);  
jf.add(B7);
```

```
 JButton B9 = new JButton("Add Tutor");  
B9.setBounds(900,480,160,60);  
jf.add(B9);
```

```
 JButton B10 = new JButton("Change to lecturer");  
B10.setBounds(1200,480,160,60);  
jf.add(B10);
```

```

B10.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent sth) {
        TeacherGUI hi=new TeacherGUI();
        hi.sam();
    }
});

```

```

B5.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent ae)
    {
        for (Teacher teacher : TeacherInfo)
        {
            if (teacher instanceof Tutor)
            {
                Tutor tutor = (Tutor) teacher;
                tutor.display();

                String TeacherName = tutor.getTeacherName();
                String TeacherID = Integer.toString(tutor.getTeacherId());
                String Address = tutor.getAddress();
                String WorkingType = tutor.getWorkingType();
                String EmploymentStatus = tutor.getEmploymentStatus();
                String WorkingHours = Integer.toString(tutor.getWorkingHours());
                int Salary = (int) Double.parseDouble(t17.getText());
                String Specialization = t18.getText();
                String Academicqualification = t19.getText();
            }
        }
    }
});

```

```
double PerformanceIndex = Double.parseDouble(t20.getText());
```

```
String message = "Teacher Name: "+TeacherName +"\n"+  
    "Teacher ID: "+TeacherID +"\n"+  
    "Address: "+Address +"\n"+  
    "Working Type: "+WorkingType +"\n" +  
    "Employment Status: "+EmploymentStatus +"\n"+  
    "Working Hours: "+WorkingHours+"\n"+  
    "Salary: " + Salary+"\n"+  
    "Specialization: "+Specialization+"\n"+  
    "Academic Qualification: "+Academicqualification+"\n"+  
    "Performance Index: "+PerformanceIndex +"\n";
```

```
// Display message
```

```
JOptionPane.showMessageDialog(null, message, "Tutor Information  
successfull", JOptionPane.INFORMATION_MESSAGE);
```

```
}
```

```
}
```

```
}
```

```
});
```

```
B6.addActionListener(new ActionListener()
```

```
{
```

```
    public void actionPerformed(ActionEvent ae)
```

```
    {
```

```
        try {
```

```
            int teacherIdToSetSalary = Integer.parseInt(t12.getText());
```

```
        int newSalary = Integer.parseInt(JOptionPane.showInputDialog("Enter  
new salary:"));
```

```
        int newPerformanceIndex =  
Integer.parseInt(JOptionPane.showInputDialog("Enter new salary:"));
```

```
        for (Teacher teacher : TeacherInfo)  
        {  
            if (teacher instanceof Tutor && teacher.getTeacherId() ==  
teacherIdToSetSalary)  
            {  
                Tutor tutorToSetSalary = (Tutor) teacher;  
                tutorToSetSalary.setSalary(newSalary, newPerformanceIndex);  
                JOptionPane.showMessageDialog(null, "Salary is updated  
successfully.");  
                return;  
            }  
        }  
    }
```

```
        JOptionPane.showMessageDialog(null, "Tutor ID not found: " +  
teacherIdToSetSalary);
```

```
    } catch (NumberFormatException e) {  
        JOptionPane.showMessageDialog(null, "Invalid input format. Please  
enter a valid number for teacher ID.");  
    }  
}  
});
```

```
B9.addActionListener(new ActionListener()  
{
```

```

public void actionPerformed(ActionEvent ae)
{
    if (t12.getText().isEmpty() || t11.getText().isEmpty() || t13.getText().isEmpty()
||
t14.getText().isEmpty() || t15.getText().isEmpty() || t16.getText().isEmpty() ||
t17.getText().isEmpty() || t18.getText().isEmpty() || t19.getText().isEmpty() ||
t20.getText().isEmpty())
    {
        JOptionPane.showMessageDialog(null, "Please Fill all the fields");
    } else
    {
        int Id = Integer.parseInt(t12.getText());
        int salary = Integer.parseInt(t17.getText());
        int workingHours = Integer.parseInt(t16.getText());
        String employmentStatus = t15.getText();
        int performanceIndex = Integer.parseInt(t20.getText());

        Tutor newTutor = new Tutor(Id, t11.getText(), t13.getText(), t14.getText(),
            t15.getText(), workingHours, salary, t18.getText(), t19.getText(),
performanceIndex);

        TeacherInfo.add(newTutor);

        JOptionPane.showMessageDialog(null, "Tutor added Successfully");

    }
}

```

```
});
```

```
B4.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent ae)
    {
        t11.setText("");
        t12.setText("");
        t13.setText("");
        t14.setText("");
        t15.setText("");
        t16.setText("");
        t17.setText("");
        t18.setText("");
        t19.setText("");
        t20.setText("");
    }
});
```

```
B7.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent ae)
    {
        try {

            int teacherIdToRemove = Integer.parseInt(t12.getText());
```

```

        for (Teacher teacher : TeacherInfo)
        {
            if (teacher instanceof Tutor && teacher.getTeacherId() ==
teacherIdToRemove)
            {

                Tutor tutorToRemove = (Tutor) teacher;
                TeacherInfo.remove(tutorToRemove);

                JOptionPane.showMessageDialog(null, teacherIdToRemove +
"Removed tutor has been successfully.");

                return;
            }
        }

        JOptionPane.showMessageDialog(null, "Tutor ID was not found: " +
teacherIdToRemove);
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(null, "Invalid input format. Please
enter a valid number for teacher ID to remove.");
    }
}

});
}

public static void main(String[] args) {
    TeacherGUI tGUI = new TeacherGUI();
    tGUI.sam();
}

```

}

}