CS4001NI Programming

30% Individual Coursework

2023-24 Autumn

Student Name: Rohan Prasad Adhikari

London Met ID: 23047505

College ID: NP01NT4A230177

Group: N7

Assignment Due Date: Friday, May 10, 2024

Assignment Submission Date: Friday, May 10, 2024

Table of Contents

Table of Tables

Table Of Figures

1. Introduction

This project or coursework focuses on the application of Java's object-oriented concept to real-world problem solving. Object-oriented programming is a style of programming that focuses on building objects and classes to build programmes and software. In this course, the Teacher class is the superclass or parent class, whereas Lecturer and Tutor are subclasses of the Teacher class. To reduce method, duplicate and code reuse, lecturers and tutors inherit the characteristics and methods of the Teacher class. Before beginning this course, we must have an excellent knowledge of the inheritance and encapsulation concepts of object-oriented programming. Encapsulation is the process of concealing sensitive or critical information from users. To access and modify the value of instance variables in the parent class, we must define getter and setter methods. We also used the concept of constructors, which are similar to methods, to initialise an object's attributes when it is first formed. To design a constructor, we should keep the class and constructor names the same, with no return type, including void.

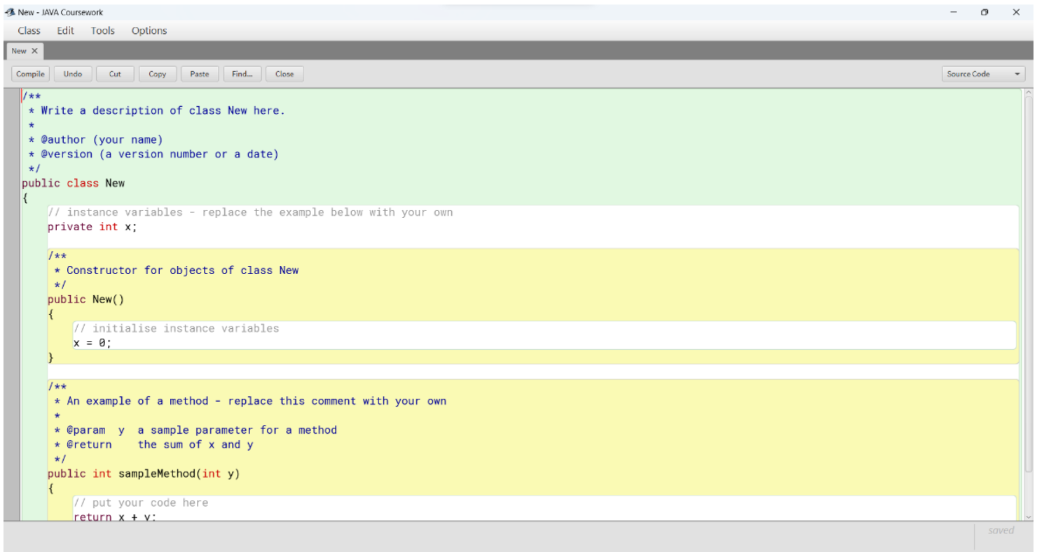
In the context of this coursework, the main goal of this coursework is to create a Graphical User Interface (GUI) for a system that uses an Array List to hold teacher (lecturer and tutor) information. I have to create a class named TeacherGUI, and it's kept in the same files as the previous assignments. I had to be familiar in Object-Oriented Programming, Array Lists, Swing, the Abstract Window Toolkit (AWT), Event Handling, Exception Handling, Type Casting, and other related concepts before starting this coursework. I created a window-based application that works as a database for teacher (lecturer and tutor) data. The teacher class in this coursework is the superclass, or parent class, and the tutor and lecturer are its subclasses. In order to prevent method duplication and code repetition, Lecturer and Tutor inherit the properties and methods of the Teacher class. A class named TeacherGUI, which is essentially used to create a GUI, and an array list of instructor classes, which store objects from both Lecturer and Tutor classes, are included in the package. This application is more user-friendly, making it simple to enter teacher information, calculate salary appraisals, grade assignments, remove tutors, and so on.

I have created frames, panels, labels, buttons, and text fields using the pre-built package Swing. One Java GUI toolkit included in the Java Foundation Classes (JFC) is called Swing. It has lightweight components like JFrame, JPanel, JDialog, JLabel, JButton, and others and is platform independent. It has the ability to change graphical components' appearance and feel interactively. It is constructed on top of AWT and has more components than AWT.

In the end, this coursework has been highly helpful for me because it taught me how to fully understand and use concepts like OOP, Collection Frameworks, Graphical User Interface, and Exception Handling, which will help me obtain an upper-level job in the future. Also, I learned how to resolve a real-world problem.

1.1. Bluej

Bluej is an integrated development environment (IDE) that runs the Java programming language. It is useful for creating projects and coding in JAVA. It is a very familiar IDE for beginners. We may check the value of objects, call methods, and pass arguments. Bluej can be run on Windows, Mac OS, Linux, and a variety of other systems. It is created specifically for use in colleges and universities. It helps in detecting our faults, whether in syntax or semantics. (Kölling & Rosenberg, 20 June 2023)



```
/**
 * Write a description of class New here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class New
{
    // instance variables - replace the example below with your own
    private int x;

    /**
     * Constructor for objects of class New
     */
    public New()
    {
        // initialise instance variables
        x = 0;
    }

    /**
     * An example of a method - replace this comment with your own
     *
     * @param  y  a sample parameter for a method
     * @return    the sum of x and y
     */
    public int sampleMethod(int y)
    {
        // put your code here
        return x + y;
```

Figure 1 : Bluej and Structure of Java

1.2. Microsoft Word

Microsoft Word is a word processing programme that was created in 1983 and is now used every day by millions of people all over the world. It is used for creating both simple and complicated documents. This programme is compatible with several platforms, including Windows and macOS. Word allows us to do a variety of things, including create headers and footers, alter font styles, and format pages. We can also open and edit external PDF files of our choosing. We can also insert photographs, videos, and so forth. MS Word can be used to produce letters, business cards, and bills. (Anon., 2023)

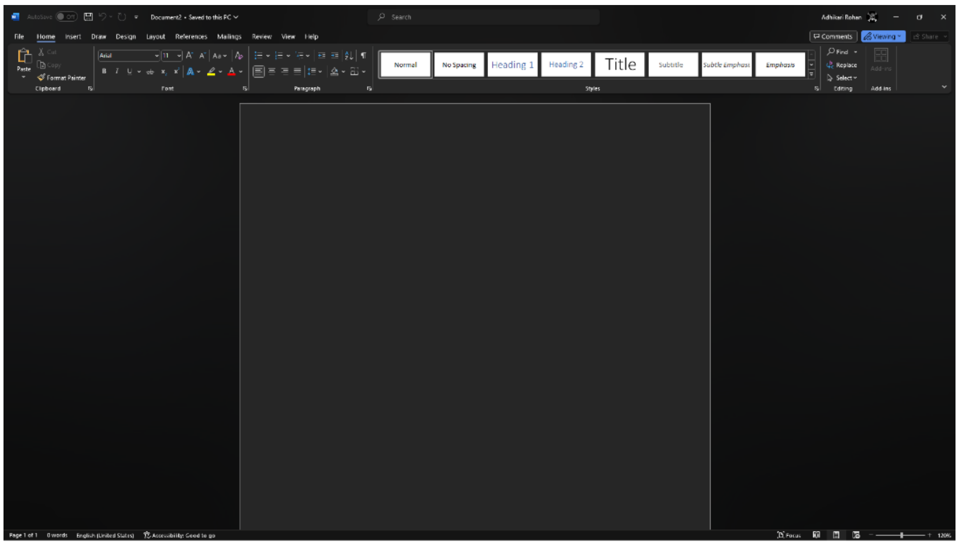


Figure 2 : Microsoft Word

1.3. Draw.io

Draw.io is a popular diagramming tool that can be used to create a variety of diagrams such as flowcharts, class diagrams, ER-Diagrams, and more. It's completely free and can be used to make our projects and courses appear better. Draw.io is widely used by individuals, teams, and organisations to create diagrams for a variety of applications, including software design and project management (Williams, February 2020). (Williams, February 2020)

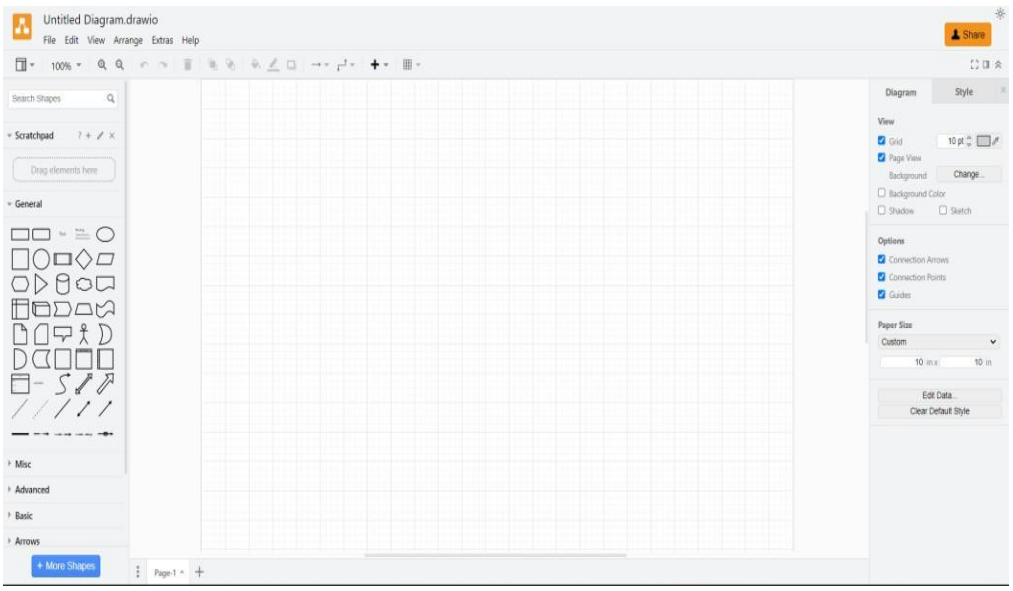


Figure 3 : Draw.io

2. Class Diagram

Class diagrams are a type of design used in object-oriented programming to help people understand and build objects. A class diagram is a visual representation of the structure and relationships of a system's classes, as well as their properties and methods. In a class diagram, a rectangular box is built using three divisions. The top section contains the class name, the middle division contains the attributes (instance variables), and the bottom division contains the methods of the class. We should understand the class diagram conventions, which are as follow below:

Symbol	Meaning
+	Represents public access modifier
-	Represents private access modifier
#	Represents protected access modifier
<<constructor>>	Represents constructor of class

Table 1 : Convention of Class Diagram

2.1. Teacher Class

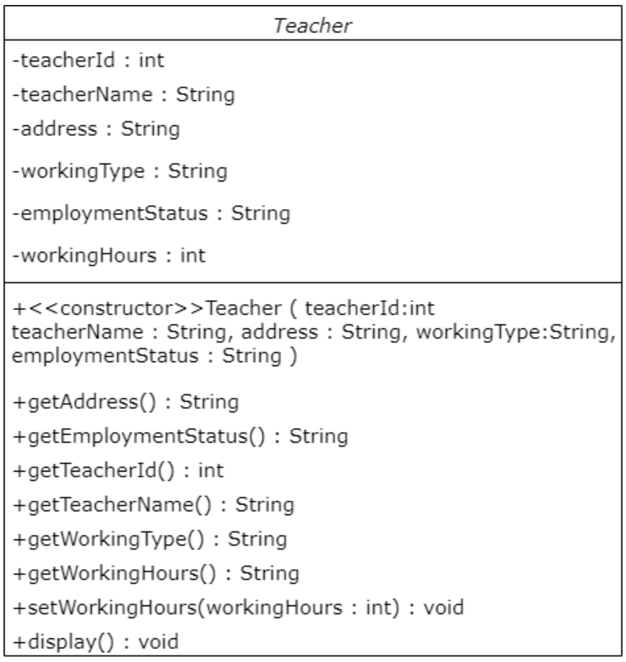

Teacher
-teacherId : int
-teacherName : String
-address : String
-workingType : String
-employmentStatus : String
-workingHours : int
+<<constructor>>Teacher (teacherId:int teacherName : String, address : String, workingType:String, employmentStatus : String)
+getAddress() : String
+getEmploymentStatus() : String
+getTeacherId() : int
+getTeacherName() : String
+getWorkingType() : String
+getWorkingHours() : String
+setWorkingHours(workingHours : int) : void
+display() : void

Figure 4 : Class Diagram of Teacher Class

2.2. Lecturer Class

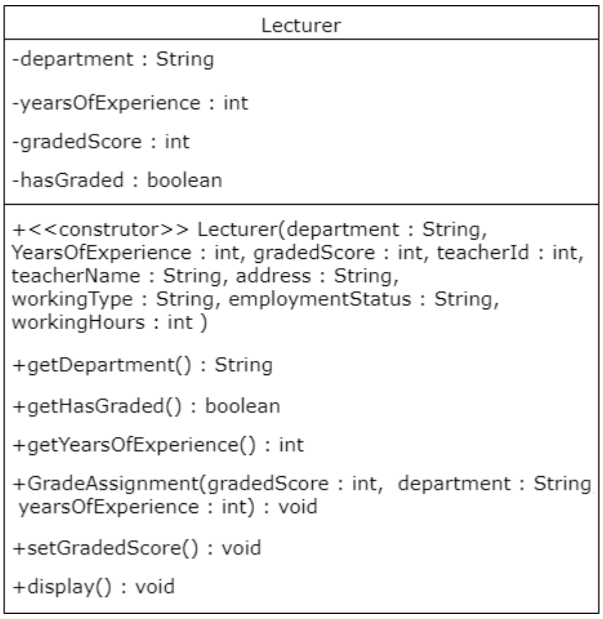
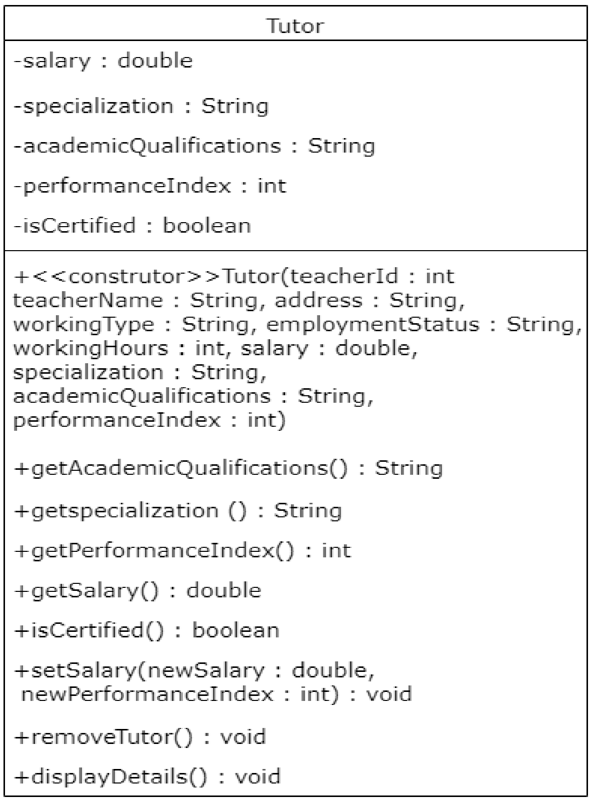

Lecturer
-department : String
-yearsOfExperience : int
-gradedScore : int
-hasGraded : boolean
+<<construtor>> Lecturer(department : String, YearsOfExperience : int, gradedScore : int, teacherId : int, teacherName : String, address : String, workingType : String, employmentStatus : String, workingHours : int)
+getDepartment() : String
+getHasGraded() : boolean
+getYearsOfExperience() : int
+GradeAssignment(gradedScore : int, department : String yearsOfExperience : int) : void
+setGradedScore() : void
+display() : void

Figure 5 : Class Diagram of Lecturer Class

2.3. Tutor Class



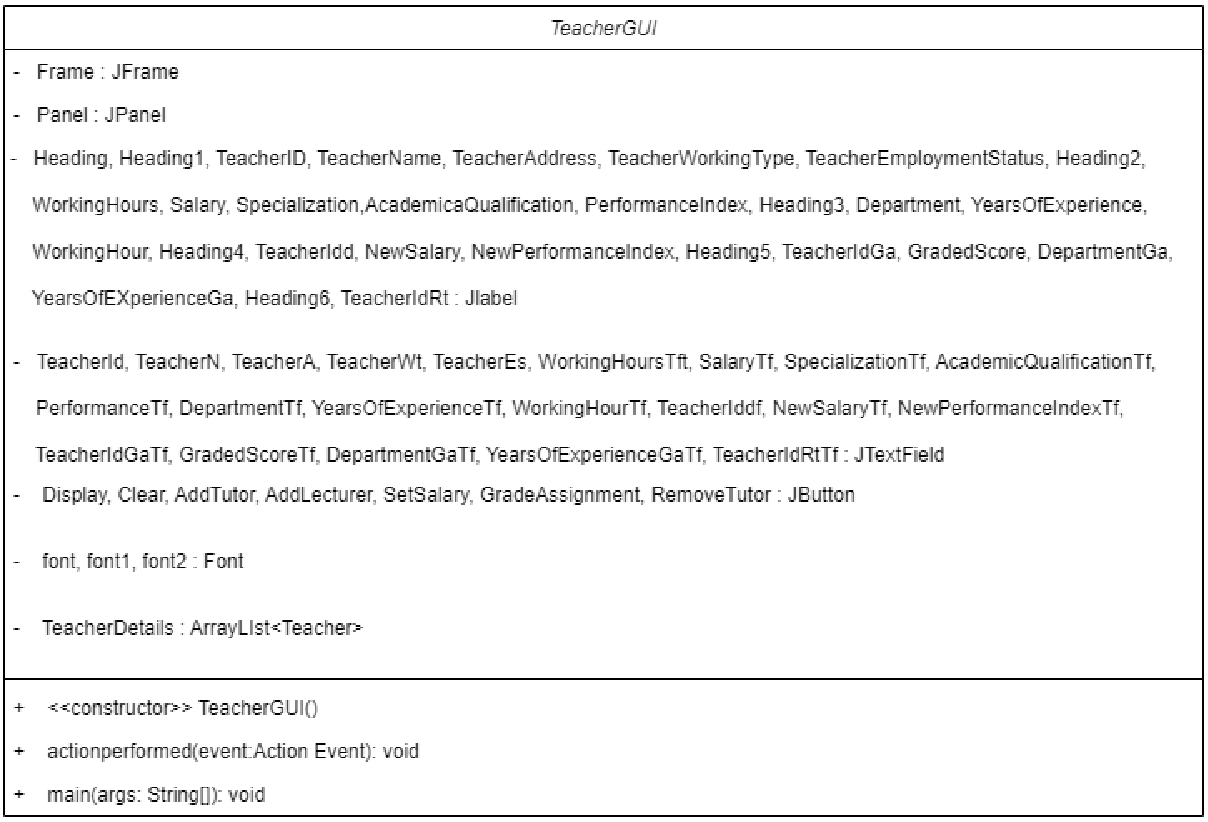
Tutor

-salary : double
-specialization : String
-academicQualifications : String
-performanceIndex : int
-isCertified : boolean

+<<construtor>>Tutor(teacherId : int teacherName : String, address : String, workingType : String, employmentStatus : String, workingHours : int, salary : double, specialization : String, academicQualifications : String, performanceIndex : int)
+getAcademicQualifications() : String
+getspecialization () : String
+getPerformanceIndex() : int
+getSalary() : double
+isCertified() : boolean
+setSalary(newSalary : double, newPerformanceIndex : int) : void
+removeTutor() : void
+displayDetails() : void

Figure 6 : Class Diagram of Tutor Class

2.4. TeacherGUI



TeacherGUI

- Frame : JFrame
- Panel : JPanel
- Heading, Heading1, TeacherID, TeacherName, TeacherAddress, TeacherWorkingType, TeacherEmploymentStatus, Heading2, WorkingHours, Salary, Specialization,AcademicaQualification, PerformanceIndex, Heading3, Department, YearsOfExperience, WorkingHour, Heading4, TeacherIdd, NewSalary, NewPerformanceIndex, Heading5, TeacherIdGa, GradedScore, DepartmentGa, YearsOfEXperienceGa, Heading6, TeacherIdRt : Jlabel
- TeacherId, TeacherN, TeacherA, TeacherWt, TeacherEs, WorkingHoursTft, SalaryTf, SpecializationTf, AcademicQualificationTf, PerformanceTf, DepartmentTf, YearsOfExperienceTf, WorkingHourTf, TeacherIddf, NewSalaryTf, NewPerformanceIndexTf, TeacherIdGaTf, GradedScoreTf, DepartmentGaTf, YearsOfExperienceGaTf, TeacherIdRtTf : JTextField
- Display, Clear, AddTutor, AddLecturer, SetSalary, GradeAssignment, RemoveTutor : JButton
- font, font1, font2 : Font
- TeacherDetails : ArrayLIst<Teacher>

+ <<constructor>> TeacherGUI()
+ actionperformed(event:Action Event): void
+ main(args: String[]): void

Figure 7 Class Diagram of TeacherGUI Class

2.5. Inheritance Class Diagram

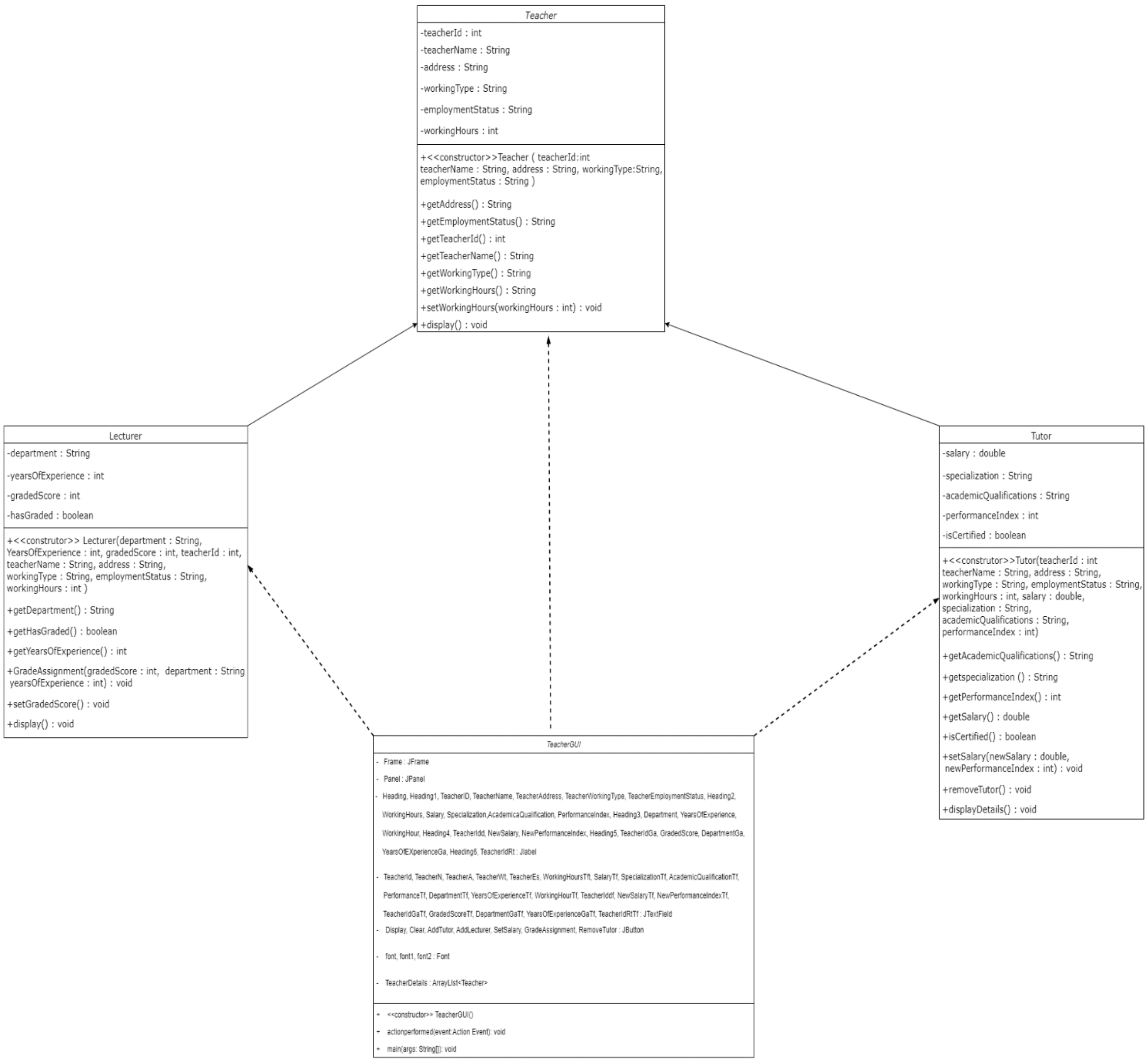

Teacher
-teacherId : int
-teacherName : String
-address : String
-workingType : String
-employmentStatus : String
-workingHours : int
+<<constructor>>Teacher (teacherId:int teacherName : String, address : String, workingType:String, employmentStatus : String)
+getAddress() : String
+getEmploymentStatus() : String
+getTeacherId() : int
+getTeacherName() : String
+getWorkingType() : String
+getWorkingHours() : String
+setWorkingHours(workingHours : int) : void
+display() : void
Lecturer
-department : String
-yearsOfExperience : int
-gradedScore : int
-hasGraded : boolean
+<<construtor>> Lecturer(department : String, YearsOfExperience : int, gradedScore : int, teacherId : int, teacherName : String, address : String, workingType : String, employmentStatus : String, workingHours : int)
+getDepartment() : String
+getHasGraded() : boolean
+getYearsOfExperience() : int
+GradeAssignment(gradedScore : int, department : String yearsOfExperience : int) : void
+setGradedScore() : void
+display() : void
Tutor
-salary : double
-specialization : String
-academicQualifications : String
-performanceIndex : int
-isCertified : boolean
+<<construtor>>Tutor(teacherId : int teacherName : String, address : String, workingType : String, employmentStatus : String, workingHours : int, salary : double, specialization : String, academicQualifications : String, performanceIndex : int)
+getAcademicQualifications() : String
+getspecialization () : String
+getPerformanceIndex() : int
+getSalary() : double
+isCertified() : boolean
+setSalary(newSalary : double, newPerformanceIndex : int) : void
+removeTutor() : void
+displayDetails() : void
TeacherGUI
- Frame : JFrame
- Panel : JPanel
- Heading, Heading1, TeacherID, TeacherName, TeacherAddress, TeacherWorkingType, TeacherEmploymentStatus, Heading2, WorkingHours, Salary, Specialization,AcademicaQualification, PerformanceIndex, Heading3, Department, YearsOfExperience, WorkingHour, Heading4, TeacherIdd, NewSalary, NewPerformanceIndex, Heading5, TeacherIdGa, GradedScore, DepartmentGa, YearsOfEXperienceGa, Heading6, TeacherIdRt : Jlabel
- TeacherId, TeacherN, TeacherA, TeacherWt, TeacherEs, WorkingHoursTft, SalaryTf, SpecializationTf, AcademicQualificationTf, PerformanceTf, DepartmentTf, YearsOfExperienceTf, WorkingHourTf, TeacherIddf, NewSalaryTf, NewPerformanceIndexTf, TeacherIdGaTf, GradedScoreTf, DepartmentGaTf, YearsOfExperienceGaTf, TeacherIdRtTf : JTextField
- Display, Clear, AddTutor, AddLecturer, SetSalary, GradeAssignment, RemoveTutor : JButton
- font, font1, font2 : Font
- TeacherDetails : ArrayList<Teacher>
+ <<constructor>> TeacherGUI()
+ actionperformed(event:Action Event): void
+ main(args: String[]): void

Figure 8 : Inheritance Class Diagram of Teacher GUI

3. Pseudocode

Pseudocode is a method of describing actual code using normal language and programming language-like syntax. It is usually written in plain English, with suitable indentation and structured statements to indicate control flow and logic. Pseudocode describes how a computer programme works when created in an Integrated Development Environment (IDE)

3.1. Pseudocode of TeacherGUI Class

```
CREATE a class TeacherGUI
Do
    DECLARE instances of ArrayList
    CREATE a constructor TeacherGUI()
    Do
        DO
            CREATE the required JPanel as Panel
            SET the required bounds to the Panel
            SET the background color to the Panel
        END DO

        DO
            CREATE the required JFrame as Frame
            SET the required size of the Frame
            SET the Jframe Layout to null
            SET the Jframe visible to true
            ADD the given JPanel as Panel
        END DO
```

```
DO
    Add the Font as font for main heading
    Add the Font as font1 for sub heading one
    Add the Font as font2 for sub heading two
END DO

DO
    CREATE the required JLabel for Heading
    SET the required Font Font in Heading
    SET the required Bounds for Heading
    ADD the Heading to Panel
END DO

DO
    CREATE the required JLabel for Heading1
    SET the required Font font one in Heading1
    SET the required Bounds for Heading1
    ADD Heading1 to Frame
END DO

DO
    CREATE the required JLabel for all Teacher components
    SET the Font font two for all Teacher components
    SET the required Bounds for all Teacher components
    ADD TeacherID to Frame for all the Teacher components
END DO
```

```
        DO
            CREATE the required JTextField for all Teacher components
            SET the Font font two for all Teacher components
            SET the required Bounds for all Teacher components
            ADD TeacherId to Frame
        END DO

        DO
            CREATE the required JButton as Display
            SET Font font two for Display
            SET the required Bounds for Display
            SET the background color gray for Display
            ADD TeacherID to Display
        END DO

        DO
            CREATE the required JButton as Clear
            SET Font font two for Clear
            SET the required Bounds for Clear
            SET the background color gray for Clear
            ADD TeacherID to Clear
        END DO

    CREATE an actionPerformed(event as ActionEvent) with return type void
    DO
        FOR instance of Teacher to TeacherDetails
        IF Teacher Instance of Tutor
        DO
            DECLARE a variable Teacher ID equals to TeacherId field text
            DECLARE a variable Teacher Name equals to TeacherName field text
            DECLARE a variable Teacher Address equals to Address field text
            DECLARE a variable Working Type equals to WorkingType field text
            DECLARE a variable Employment Status equals to EmploymentStatus field text
            DECLARE a variable Working Hours equals to WorkingHoursfield text
            DECLARE a variable Salary equals to Salary_field text
            DECLARE a variable Specialization equals to Specialization_field text
            DECLARE a variable Academic Qualification equals to AcademicQualification_field text
            DECLARE a variable Performance Index equals to PerformanceIndex_field text
            DISPLAY Tutor Details
            DISPLAY message Tutor box Information
        END DO
        END IF
        ELSE (instance of Lecturer)
        DO
            DECLARE a variable Teacher ID equals to TeacherId field text
            DECLARE a variable Teacher Name equals to TeacherName field text
            DECLARE a variable Teacher Address equals to Address field text
            DECLARE a variable Working Type equals to WorkingType field text
            DECLARE a variable Employment Status equals to EmploymentStatus field text
            DECLARE a variable Department equals to Department field text
            DECLARE a variable Years of Experience equals to Years of Experience field text
            DECLARE a variable Working Hours equals to Working Hours
            field text
            DISPLAY lecturer
            DISPLAY message box Lecturer Information

END DO
ELSE END
END DO

DO
        CREATE the required JLabel as Heading2 for Tutor
        SET font font one for Heading2
        SET the required Bounds for Heading2
        ADD Heading2 to Frame
END DO

DO
        CREATE the required JLabel for all Tutor components
        SET the Font font two for all Tutor components
        SET the required Bounds for all Tutor components
        ADD all the Tutor components to Frame
END DO

DO
        CREATE the required JTextField for all Tutor components
        SET the Font font two for all Tutor components
        SET the required Bounds for all Tutor components
        ADD all components of Tutor to Frame
END DO
```

```
DO
    CREATE the required JButton for AddTutor
    SET Font font two of AddTutor
    SET the required Bounds for AddTutor
    ADD AddTutor to Frame
END DO

CREATE an actionPerformed(event as ActionEvent) with return type void
DO
Try
    DO
        DECLARE a variable TeacherID equals to parse integer with TeacherId_field text as an arguments
        DECLARE a variable TeacherName equals to TeacherN_field text as an arguments
        DECLARE a variable TeacherAddress equals to TeacherA_field text as an arguments
        DECLARE a variable TeacherWorkingType equals to TeacherWt_field text as an arguments
        DECLARE a variable TeacherEmploymentStatus equals to TeacherEs_field text as an arguments
        DECLARE a variable WorkingHours equals to parse integer with WorkingHoursTfT_field text as an arguments
        DECLARE a variable Salary equals to parse boolean with SalaryTf_field text as an arguments
        DECLARE a variable Specialization equals to SpecializationTf _field text as an arguments
        DECLARE a variable AcademicQualification equals to AcademicQualificationTf _field text as an arguments
        DECLARE a variable PerformanceIndex equals to parse integer with PerformanceIndexTf_field text as an arguments
```

```
                FOR
                DO
                    IF (If teacherId equals to teacherId)
                        DO
                            DISPLAY message Teacher ID exists. Please add correct ID
                        END DO
                    END IF
                END DO
                END FOR

                CREATE a constructor(TeacherID, TeacherName, TeacherAddress, TeacherWorkingType, TeacherEmploymentStatus, WorkingHours, Salary, Specialization, AcademicQualification, PerformanceIndex)
                ADD Tutorvalue to TeacherDetails
                DISPLAY message Successfully! Added Tutor
            END DO

            CATCH (NumberFormatException e)
            DO
                DISPLAY message Please Kindly fill the NUMERIC VALUE
            END DO

            IF (textfield is empty)
                DO
                    DISPLAY message Please Kindly fill the VALUE
                END DO
            END IF
        END DO
```

```
DO
    CREATE the required JLabel as Heading3 for Lecturer
    SET Font font one of Heading3
    SET the required Bounds of Heading3
    ADD Heading3 to Frame
END DO

    DO
        CREATE the required JLabel for all Lecturer components
        SET the Font font two for all Lecturer components
        SET the required Bounds for all Lecturer components
        ADD all the Lecturer components to Frame
    END DO

    DO
        CREATE the required JTextField for all Lecturer components
        SET the Font font two for all Lecturer components
        SET the required Bounds for all Lecturer components
        ADD all the Lecturer components to Frame
    END DO


    DO
        CREATE the required JButton as AddLecturer
        SET font font two of AddLecturer
        SET the required Bounds for AddLecturer
        ADD AddLecturer to Frame
    END DO
```

```
CREATE an actionPerformed(event as ActionEvent) with return type void
DO
Try
    DO
        DECLARE a variable Department equals to DepartmentTf_field text as an arguments
        DECLARE a variable YearsOfExperience equals to parse integer with YearsOfExperienceTf_field text as an arguments
        DECLARE a variable TeacherID equals to parse integer with TeacherId_field text as an arguments
        DECLARE a variable TeacherName equals to TeacherN_field text as an arguments
        DECLARE a variable TeacherAddress equals to TeacherA_field text as an arguments
        DECLARE a variable TeacherWorkingType equals to TeacherWt_field text as an arguments
        DECLARE a variable TeacherEmploymentStatus equals to TeacherEs_field text as an arguments
        DECLARE a variable WorkingHours equals to parse integer with WorkingHoursTf_field text as an arguments
        FOR
        DO
            IF (If teacherId equals to teacherId)
                DO
                    DISPLAY message Teacher ID exists. Please add correct ID
                END DO
            END IF
        END DO
        END FOR

        CREATE a constructor (Department, YearsOfExperience, TeacherID, TeacherName, TeacherAddress,
                TeacherWorkingType, TeacherEmploymentStatus,
                WorkingHours)
                **ADD** Lect to TeacherDetails
                **DISPLAY** message Successfully! Added Lecturer
        **END DO**

        **CATCH** (NumberFormatException e)
        **DO**
                **DISPLAY** message Please Kindly fill the NUMERIC
                VALUE
        **END DO**

        **IF** (textfield is empty)
                **DO**
                        DISPLAY message Please Kindly fill the VALUE
                **END DO**
        **END IF**
**END DO**
```

```
DO
        CREATE the required JLabel as Heading4 for Salary
        SET the font font two for Heading4
        SET the required Bounds of Heading4
        ADD Heading4 to Frame
END DO

DO
        CREATE the required JLabel for all salary components
        SET the Font font two for all salary components
        SET the required Bounds for all salary components
        ADD  all the salary components to the Frame
END DO

DO
        CREATE the required JTextField for all salary components
        SET the Font font two for all salary components
        SET the required Bounds for all salary components
        ADD all the salary components to the Frame
END DO

DO
        CREATE the required JButton as SetSalary
        SET font font two of SetSalary
        SET the required Bounds for SetSalary
        ADD SetSalary to Frame
END DO
```

```
CREATE an actionPerformed (event as ActionEvent) with return type void
DO
Try
    DO
        DECLARE a variable teacherId equals to TeacherIddTf_field text as an arguments
        DECLARE a variable newSalary equals to parse boolean with NewSalaryTf_field text as an arguments
        DECLARE a variable newPerformanceIndex equals to parse integer with newPerformanceIndex_field text as an arguments
        UPDATE Tutor to tutorToUpdate equals to null
            FOR
            DO
                IF (instance of tutor and TeacherId equals to teacherId)
                    DO
                        UPDATE tutorToUpdate equals to teacher
                        BREAK
                    END Do
                END IF
            END DO
            END FOR

        IF (tutorToUpdate equals to null)
            DO
                DISPLAY message Invalid Teacher ID! No matching Tutor found
            END DO
            END IF
            UPDATE tutorToUpdate to setSalary (newSalary, newPerformanceIndex)
```

```
                **DISPLAY** message Salary and Performance Index
                updated successfully!
        END DO

        CATCH (NumberFormatException e)
        DO
            DISPLAY message Please Kindly fill the NUMERIC
            VALUE
        END DO

        CATCH (Exception e)
        DO
            DISPLAY message Error
        END DO
    END DO



    DO
        CREATE the required JLabel as Heading5 for Grade Assingment
        SET font font one for Heading5
        SET the required Bounds of Heading5
        ADD Heading5 to Frame
    END DO
```

```
DO
    CREATE the required JLabel for all Grade Assignment components
    SET the Font font two for all Grade Assignment components
    SET the required Bounds for all Grade Assignment components
    ADD all the Grade Assingment components to Frame
END DO

DO
    CREATE the required JTextField for all Grade Assingment components
    SET the Font font two for all Grade Assingment components
    SET the required Bounds for all Grade Assingment components
    ADD all the Grade Assingment components to Frame
END DO

DO
    CREATE the required JButton as GradeAssignments
    SET font font two for GradeAssignments
    SET the required Bounds for GradeAssignments
    ADD GradeAssignments button  to Frame
END DO

CREATE an actionPerformed (event as ActionEvent) with return type void
DO
Try
    DO
        DECLARE a variable teacherId equals to TeacherIddTf_field text as an arguments
        DECLARE a variable gradedScore equals to parse boolean with GradedScoreTf_field text as an arguments
        DECLARE a variable department equals to
        DepartmentGaTf_field text as an arguments

        DECLARE a variable yearsOfExperience equals to parse
        boolean with yearsOfExperience_field text as an
        arguments


        UPDATE Lecturer to lecturerToUpdate equals to null

            FOR (instance of teacher to TeacherDetails)

            DO

                IF (instance of Lecturer and TeacherId
                equals to teacherId)

                    DO

                        UPDATE lecturerToUpdate
                        equals to teacher

                        UPDATE lecturerToUpdate to
                        GradeAssignment
                        (gradedScore, department,
                        yearsOfExperience)

                        BREAK

                    END DO

                END IF

            END DO

            END FOR


            IF (lecturerToUpdate equals to null)

            DO

                DISPLAY message Invalid Teacher ID! No
                matching Lecturer found

            END DO

            END IF

            IF (lecturerToUpdate doesn’t equals to department)

            DO
```

```
                        DISPLAY message Department or Years of
                        Experience do not match the Lecturer's
                        details

                        DISPLAY message Graded Score updated
                        successfully!
                    END DO
                    END IF

            END DO

            CATCH (NumberFormatException e)
            DO
                DISPLAY message Please Kindly fill the NUMERIC
                VALUE
            END DO

            CATCH (Exception e)
            DO
                DISPLAY message Error
            END DO
        END DO

        DO
            CREATE the reqired JLabel as Heading6 for Remove Tutor
            SET Font font one for Heading6
            SET  the required Bounds of Heading6
            ADD Heading6 to Frame
        END DO
```

```
DO
          CREATE the required JLabel for all Remove Tutor components
          SET the Font font two for all Remove Tutor components
          SET the required Bounds for all Remove tutor components
          ADD all the Remove Tutor components to Frame
     END DO

     DO
          CREATE the required JTextField for all Remove tutor components
          SET the Font font two for all Remove tutor components
          SET the required Bounds for all Remove tutor components
          ADD all the Remove Tutor components to Frame
     END DO
     DO
          CREATE the required JButton as RemoveTutor
          SET font font two of RemoveTutor
          SET the required Bounds of RemoveTutor
          ADD the required button RemoveTutor to Frame
     END DO

     CREATE an actionPerformed (event as ActionEvent) with return type void
     DO
     Try
          DO
               DECLARE a variable teacherId equals to TeacherIdRtTf_field text as an arguments

               UPDATE Tutor to tutorToRemove equals to null
                    FOR (instance of teacher to TeacherDetails)
```

```
        DO
            IF (instance of tutor and TeacherId equals to teacherId)
                DO
                    tutorToRemove equals to teacher
                    BREAK
                END DO
            END IF
        END DO
        END FOR

        IF (lecturerToUpdate equals to null)
        DO
            DISPLAY message Invalid Teacher ID! No matching Tutor found
        END DO
        END IF
        UPDATE teacherdetails to remove tutorToRemove
        DISPLAY message Tutor removed successfully!

END DO

CATCH (NumberFormatException e)
DO
    DISPLAY message Please Kindly fill the NUMERIC VALUE
END DO
CATCH (Exception e)
DO
    DISPLAY message Error
END DO
```

```
        END DO
        CREATE an actionPerformed (event as ActionEvent) with return type
        void
        DO
            SET TeacherId _field with empty string
            SET TeacherN _field with empty string
            SET TeacherA _field with empty string
            SET TeacherWt _field with empty string
            SET TeacherEs _field with empty string
            SET WorkingHoursTfT _field with empty string
            SET SalaryTf _field with empty string
            SET SpecializationTf _field with empty string
            SET AcademicQualificationTf _field with empty string
            SET PerformanceIndexTf _field with empty string
            SET DepartmentTf _field with empty string
            SET YearsOfExperienceTf _field with empty string
            SET WorkingHourTf _field with empty string
            SET TeacherIddTf _field with empty string
            SET NewSalaryTf _field with empty string
            SET NewPerformanceIndexTf _field with empty string
            SET TeacherIdGaTf _field with empty string
            SET GradedScoreTf _field with empty string
            SET DepartmentGaTf _field with empty string
            SET YearsOfExperienceGaTf _field with empty string
            SET TeacherIdRtTf _field with empty string
            DISPLAY message All Fields are CLEARED
        END DO
    END DO
END DO
```

4. Methods and Buttons

A method is a part of code that performs certain tasks when applied. It enables code reuse without having to duplicate the same code many times. A method must be declared within the class, followed by brackets. We can pass parameters into the method and set the arguments when it is executed.

Syntax of Method:

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

4.1. Methods of Teacher GUI

S. N.	Method Name	Description
1.	main (String [] args)	It is the main of the class and used to call the constructor
2.	actionPerformed(Action Event event)	It is used to performed when an event occurs like clicking a button.

Table 2 : Methods of Teacher GUI

4.2. Buttons

S.N.	Buttons	Description
1.	Display button of Jframe to add Display	When this button is clicked, the code checks whether the ArrayList 'teacher' is empty using isEmpty() method that return either true or false. If the ArrayList is empty, a warning message dialog is displayed to the user using JOptionPane.showMessageDialog(). If the ArrayList is not empty, the further code proceeds to loop through each object in the ArrayList and checks whether the objects stored in the ArrayList is instance of Tutor and Lecturer. If the object is instance of Tutor and Lecturer, the code casts object of teacher to a Tutor and Lecturer object and assign it to a new variable teacher1. Then, display () method of regular class is called and print all the information of student in the terminal.
2.	Clear button of Jframe to clear text fields	When this button is clicked, the code sets the empty string to all the respective fields of the program/GUI.
3.	AddTutor button of Jframe to add Add Tutor	When this button is clicked, the program checks if any of the text fields are empty using getText() which is used to return the text of the fields and isEmpty() method that basically return Boolean value either true or false. If any of text field is empty, a warning message dialog box will be displayed to the user using JOptionPane.showMessageDialog(). After filling all the text fields, the program proceeds to handle expected exceptions (NumberFormatException) which may occur during the conversion of data. After that, the program extracts the value of Tutor

		all textfields and stores the value accordingly in the array list. It converts the value of working hours, salary and performance index into the string using parseInt() method. If all fields are fields correctly it throws message which displays message that Tutor added successfully with all teacher details.
4.	AddLecturer button of Jframe to add Add Lecturer	When this button is clicked, the program checks if any of the text fields are empty using getText() which is used to return the text of the fields and isEmpty() method that basically return Boolean value either true or false. If any of text field is empty, a warning message dialog box will be displayed to the user using JOptionPane.showMessageDialog(). After filling all the text fields, the program proceeds to handle expected exceptions (NumberFormatException) which may occur during the conversion of data. After that, the program extracts the value of Lecturer all textfields and stores the value accordingly in the array list. It converts the value of working hours, salary and performance index into the string using parseInt() method. If all fields are fields correctly it throws message which displays message that Lecturer added successfully with all teacher details.
5.	SetSalary button of Jframe to add SetSalary	When this button is clicked, If all the fields are empty it will throw a message in a dialog box that fields are empty with the help of isEmpty() method and JOptionPane.showMessageDialog().After filling all the text fields, the code proceeds to handle expected exceptions

		(NumberFormatException) which may occur during the conversion of data. After that, the code extracts the value of Teacher ID, New Salary and New Performance Index convert it into integer using parseInt() method. After the input of all three text fields then in the backend it will calculate the salary of the tutor on the basis od new salary and new performance index. The calculation of the new salary is salary multiple by appraisal percentage. And at last, it will throw a message in a dialog box that new salary has been set.
6.	GradeAssignment button of Jframe to add Grade Assignment	When this button is clicked, If all the fields are empty it will throw a message in a dialog box that fields are empty with the help of isEmpty() method and JOptionPane.showMessageDialog(). After filling all the text fields, the code proceeds to handle expected exceptions (NumberFormatException) which may occur during the conversion of data. After that, the code extracts the value of Teacher ID, Graded Score and Year of experience convert it into integer using parseInt() method. After the input of all three text fields then in the backend it will calculate the Grade of the student on the basis of Grade has been assigned. their obtained mark i.e. Graded score. After the calculation it will display a message on the console their corresponding Grade. And at last, it will throw a message in a dialog box that Grade has been assigned.

7.	RemoveTutor button of Jframe to add Remove Tutor	When this button is clicked, If all the fields are empty it will throw a message in a dialog box that fields are empty with the help of isEmpty() method and JOptionPane.showMessageDialog(). After filling all the text fields, the code proceeds to handle expected exceptions (NumberFormatException) which may occur during the conversion of data. After that, the code extracts the value of Teacher ID, convert it into integer using parseInt() method. After the input of text fields then in the backend it will checks the teacher id matches or not which input in the teacher Id above if not it will throw a new message in the dialog box that invalid teacher id else it matches then it will remove the tutor with all details that has been entered.

Table 3 : Buttons of the TeacherGUI

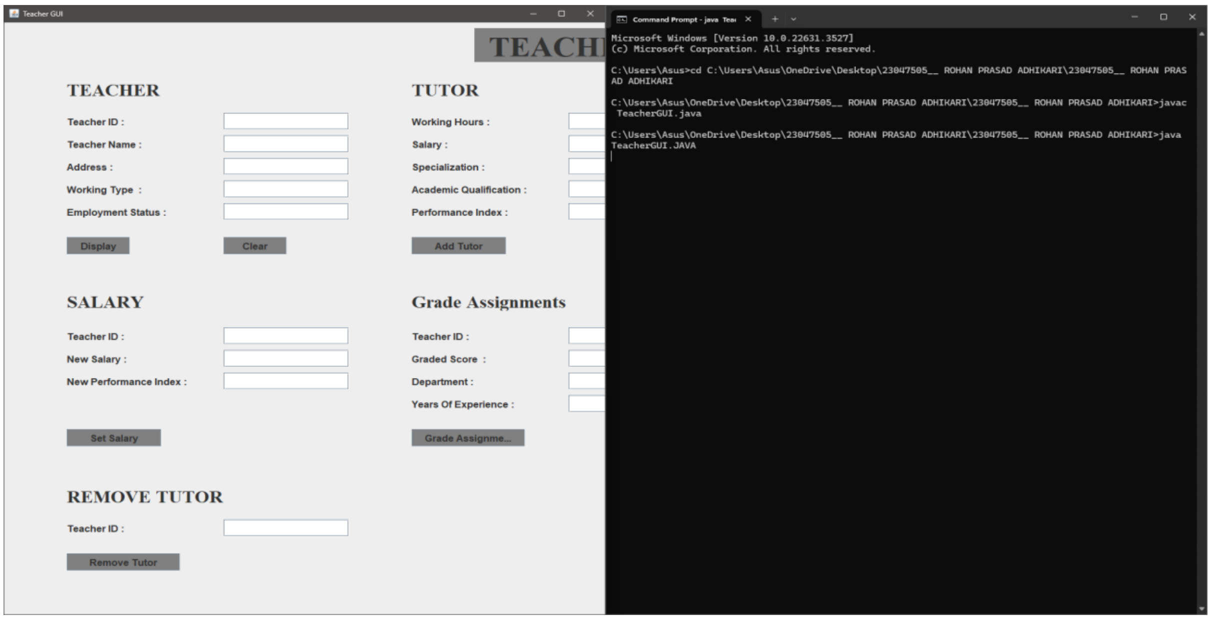
5. Testing

5.1. Test 1

5.1.1. Test 1: Using command prompt to run the program

Objective	**To Test the program which can be compiled and run using the command prompt**
Action	• The command prompt was run. • After opening the command prompt, the java file was compiled using syntax "javac TeacherGUI.java".
Expected result	After opening the command prompt from the folder, the TeacherGUI.java file should be compiled and should run through command prompt.
Actual result	After opening the command prompt, the TeacherGUI.java file was compiled and ran through cmd.
Conclusion	Test successful

Table 4 : Using command prompt to run the program.



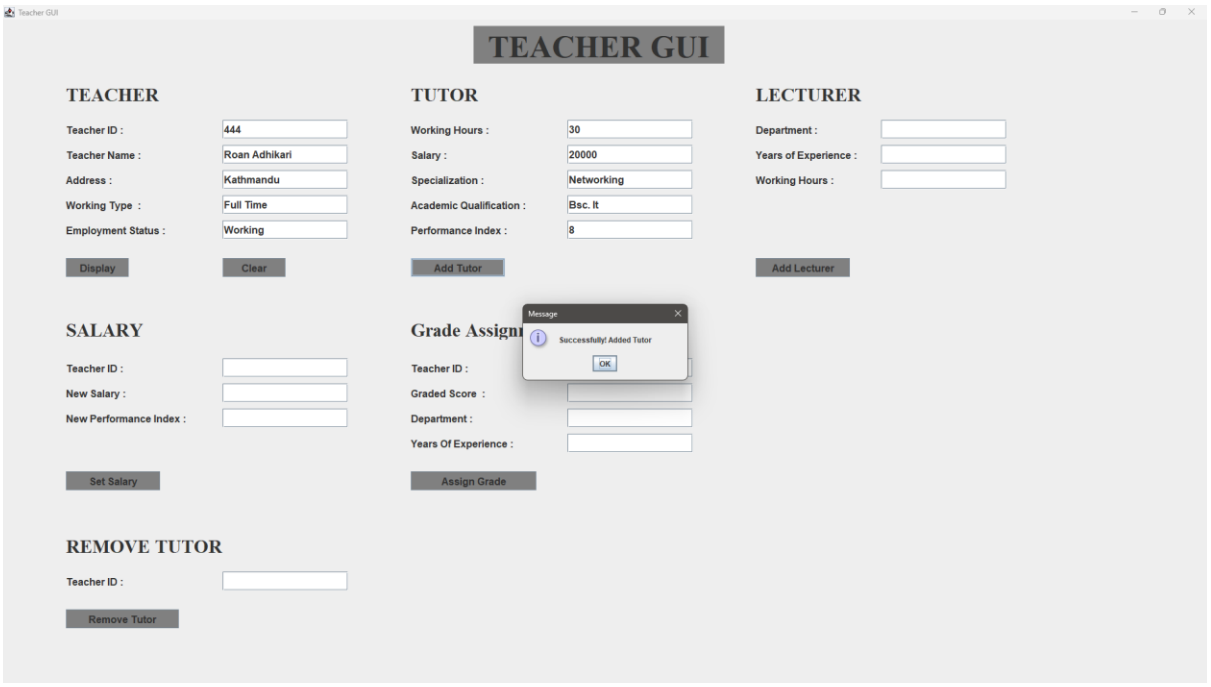
```
Microsoft Windows [Version 10.0.22631.3527]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Asus>cd C:\Users\Asus\OneDrive\Desktop\23047505__ ROHAN PRASAD ADHIKARI\23047505__ ROHAN PRAS
AD ADHIKARI

C:\Users\Asus\OneDrive\Desktop\23047505__ ROHAN PRASAD ADHIKARI\23047505__ ROHAN PRASAD ADHIKARI>javac
 TeacherGUI.java

C:\Users\Asus\OneDrive\Desktop\23047505__ ROHAN PRASAD ADHIKARI\23047505__ ROHAN PRASAD ADHIKARI>java
TeacherGUI.JAVA
```

Figure 9 : Using command prompt to run the program.

5.2. Test 2:

5.2.1. Test to Add Tutor

Objective	**To add the Tutor when Add Tutor button is clicked and a dialog appears when button is clicked**
Action	The TeacherGUI class was compiled and Add Tutor button of main frame was clicked. The following information for Tutor was entered: • Teacher – Teacher ID: 444 – Teacher Name: Roan Adhikari – Address: Kathmandu – Working Type: Full Time – Employment Status: Working • Tutor – Working Hours: 30 – Salary: 20000 – Specialization: Networking – Academic Qualification: BSc. It – Performance Index: 8 • And at last, Add Tutor button is clicked
Expected result	After entering all the details of Tutor, the values should be added correctly and a dialog box appeared that “Tutor added Successfully” and while display button clicked all the information of Tutor has to be in the dialog box.
Actual result	An information dialog appeared with “Tutor added Successfully” and also displayed all information of the Tutor.
Conclusion	Test Successful



Teacher ID : 444
Teacher Name : Roan Adhikari
Address : Kathmandu
Working Type : Full Time
Employment Status : Working

Working Hours : 30
Salary : 20000
Specialization : Networking
Academic Qualification : Bsc. It
Performance Index : 8

Department :
Years of Experience :
Working Hours :

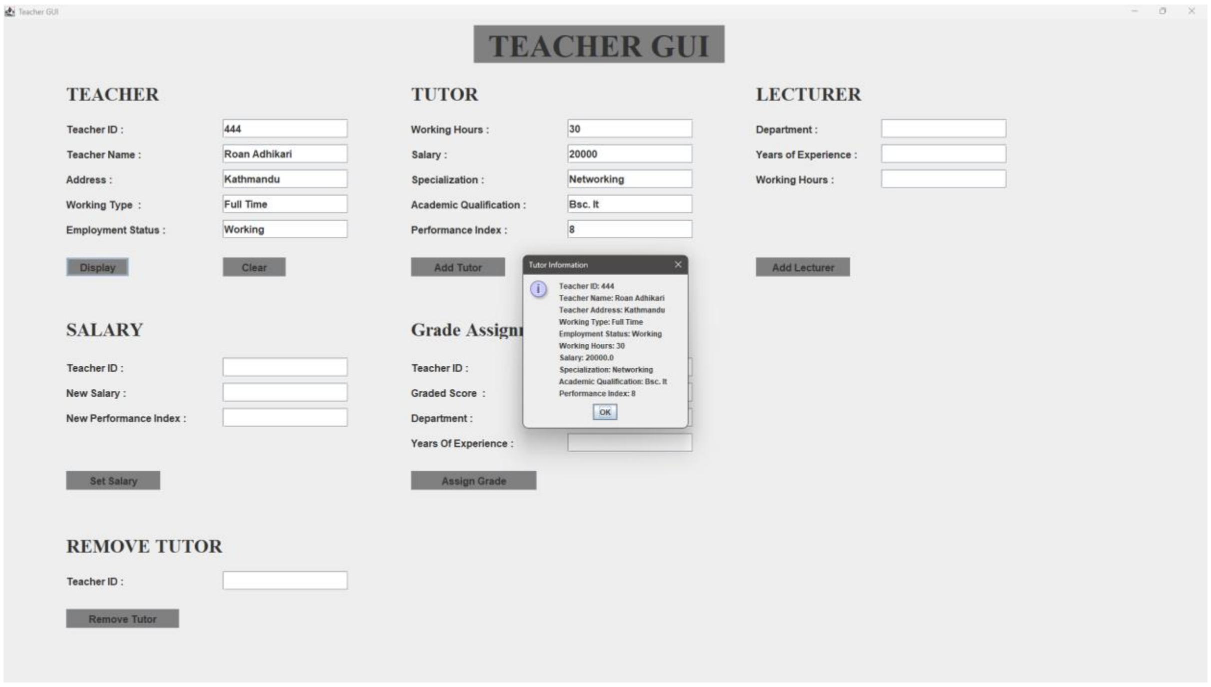
Message
Successfully! Added Tutor
OK

Teacher ID :
New Salary :
New Performance Index :

Teacher ID :
Graded Score :
Department :
Years Of Experience :

Teacher ID :

Figure 10 : Tutor Added



Teacher ID : 444
Teacher Name : Roan Adhikari
Address : Kathmandu
Working Type : Full Time
Employment Status : Working

Working Hours : 30
Salary : 20000
Specialization : Networking
Academic Qualification : Bsc. It
Performance Index : 8

Department :
Years of Experience :
Working Hours :

Tutor Information
Teacher ID: 444
Teacher Name: Roan Adhikari
Teacher Address: Kathmandu
Working Type: Full Time
Employment Status: Working
Working Hours: 30
Salary: 20000.0
Specialization: Networking
Academic Qualification: Bsc. It
Performance Index: 8
OK

Teacher ID :
New Salary :
New Performance Index :

Teacher ID :
Graded Score :
Department :
Years Of Experience :

Teacher ID :

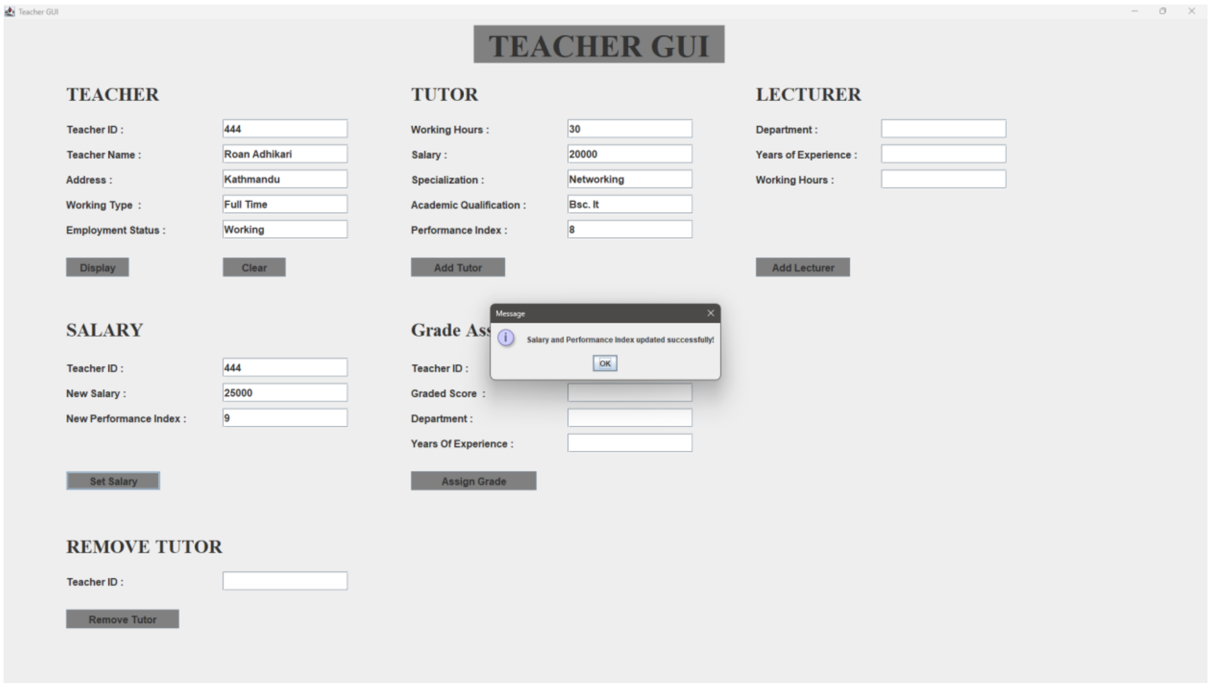
Figure 11 : Tutor Information Displayed in Dialog Box

5.2.2. Test 2: Setting Salary

Objective	**To set the new salary when Set Salary button is clicked and a dialog appears when button is clicked**
Action	• The TeacherGUI class was compiled and Add Lecturer button of main • frame was clicked. • The following information for Tutor was entered: • Teacher – Teacher ID: 444 – Teacher Name: Roan Adhikari – Address: Kathmandu – Working Type: Full Time – Employment Status: Working • Tutor – Working Hours: 30 – Salary: 20000 – Specialization: Networking – Academic Qualification: BSc. It – Performance Index: 8 • Salary – Teacher Id: – New Salary: 25000 – New Performance Index: 9 • And at last, Set Salary button is clicked
Expected result	After entering all the details of Tutor, the values should be added correctly and a dialog box appeared that “Salary has been set” and while display button clicked all the information of Tutor has to be in the dialog box with new salary which is calculated on the basis of the salary and appraisal percentage. Also, while all the information should be also displayed in terminal

Actual result	An information dialog appeared with “Salary has been set” and also displayed all information of the Tutor with new salary along with information displayed in terminal.
Conclusion	Test Successful

Table 5 : Setting Salary



TEACHER

Teacher ID : 444
Teacher Name : Roan Adhikari
Address : Kathmandu
Working Type : Full Time
Employment Status : Working

TUTOR

Working Hours : 30
Salary : 20000
Specialization : Networking
Academic Qualification : Bsc. It
Performance Index : 8

LECTURER

Department :
Years of Experience :
Working Hours :

SALARY

Teacher ID : 444
New Salary : 25000
New Performance Index : 9

Grade Ass

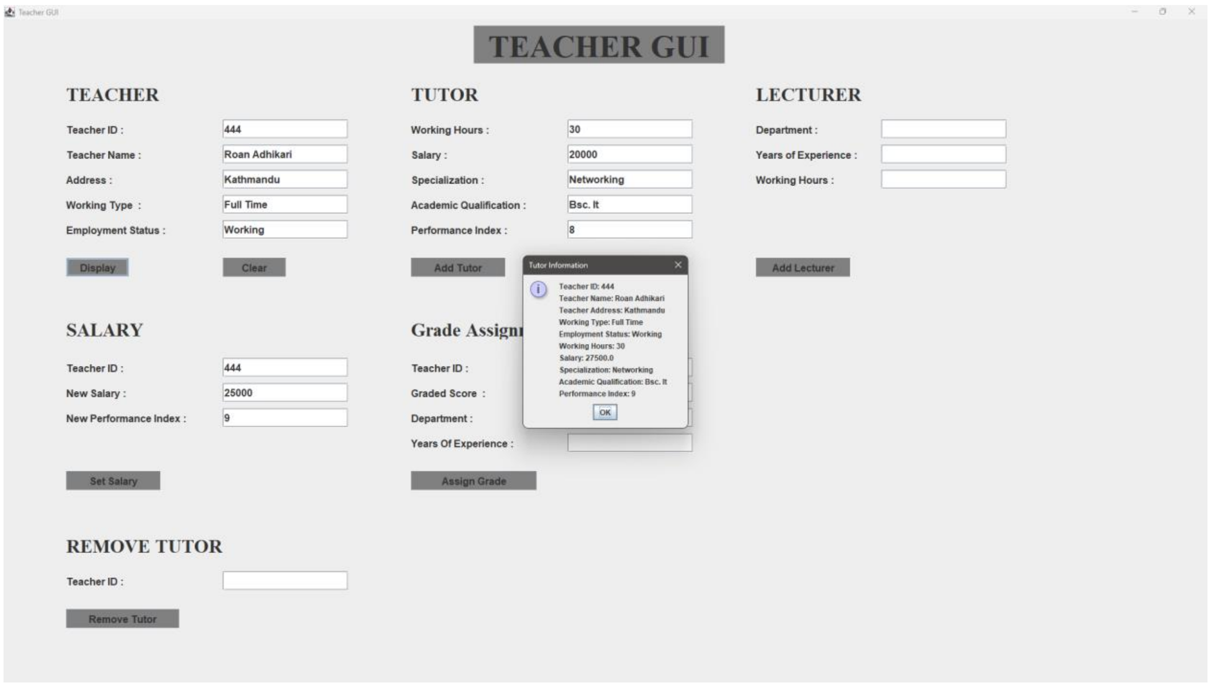
Teacher ID :
Graded Score :
Department :
Years Of Experience :

REMOVE TUTOR

Teacher ID :

Salary and Performance Index updated successfully!

Figure 12 : Updating the New Salary and Performance Index



TEACHER

Teacher ID : 444
Teacher Name : Roan Adhikari
Address : Kathmandu
Working Type : Full Time
Employment Status : Working

TUTOR

Working Hours : 30
Salary : 20000
Specialization : Networking
Academic Qualification : Bsc. It
Performance Index : 8

LECTURER

Department :
Years of Experience :
Working Hours :

SALARY

Teacher ID : 444
New Salary : 25000
New Performance Index : 9

Grade Assign

Teacher ID :
Graded Score :
Department :
Years Of Experience :

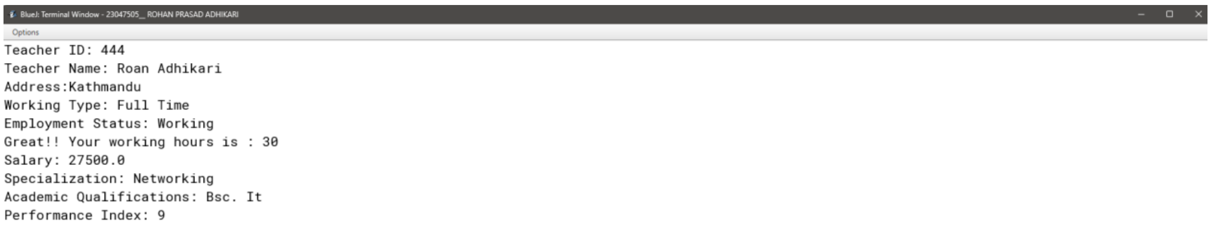
REMOVE TUTOR

Teacher ID :

Tutor Information

Teacher ID: 444
Teacher Name: Roan Adhikari
Teacher Address: Kathmandu
Working Type: Full Time
Employment Status: Working
Working Hours: 30
Salary: 27500.0
Specialization: Networking
Academic Qualification: Bsc. It
Performance Index: 9

Figure 13 : New salary and Performance Index Updated



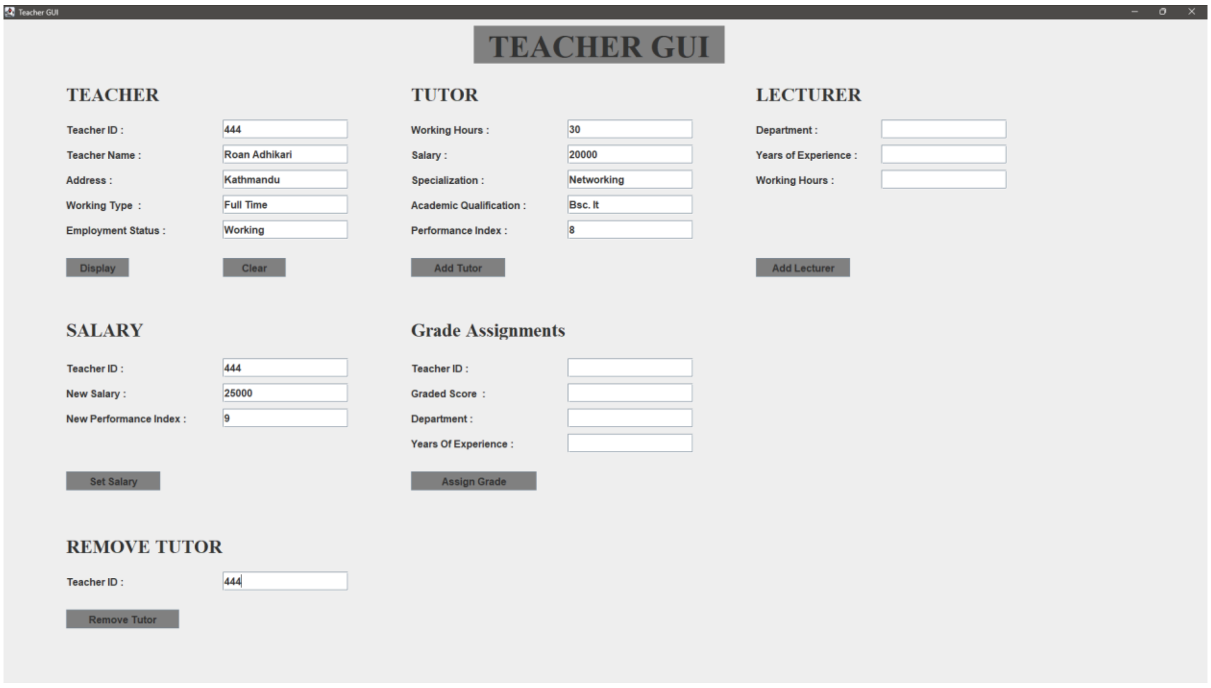
```
Teacher ID: 444
Teacher Name: Roan Adhikari
Address:Kathmandu
Working Type: Full Time
Employment Status: Working
Great!! Your working hours is : 30
Salary: 27500.0
Specialization: Networking
Academic Qualifications: Bsc. It
Performance Index: 9
```

Figure 14 : Information of Tutor with updated salary and performance index

5.2.3. Removing Tutor

Objective	**To remove the Tutor when Remove Tutor button is clicked and a dialog appears when button is clicked**
Action	• The TeacherGUI class was compiled and Remove Tutor button of main frame was clicked. • The following information for Tutor was entered: • Teacher – Teacher ID: 444 – Teacher Name: Roan Adhikari – Address: Kathmandu – Working Type: Full Time – Employment Status: Working • Tutor – Working Hours: 30 – Salary: 20000 – Specialization: Networking – Academic Qualification: BSc. It – Performance Index: 8 • Remove Tutor – Teacher Id: 444 • And at last, Remove Tutor button is clicked
Expected result	After entering all the details of Tutor, then it will remove the tutor of the entered tutor id from the program and a dialog box appeared with message of “Tutor Removed Successfully”
Actual result	Tutor removed and a dialog box appeared with message of “Tutor Removed Successfully”
Conclusion	Test Successful

Table 6 : Removing Tutor



Teacher GUI

TEACHER GUI

TEACHER

Teacher ID : 444
Teacher Name : Roan Adhikari
Address : Kathmandu
Working Type : Full Time
Employment Status : Working

TUTOR

Working Hours : 30
Salary : 20000
Specialization : Networking
Academic Qualification : Bsc. It
Performance Index : 8

LECTURER

Department :
Years of Experience :
Working Hours :

SALARY

Teacher ID : 444
New Salary : 25000
New Performance Index : 9

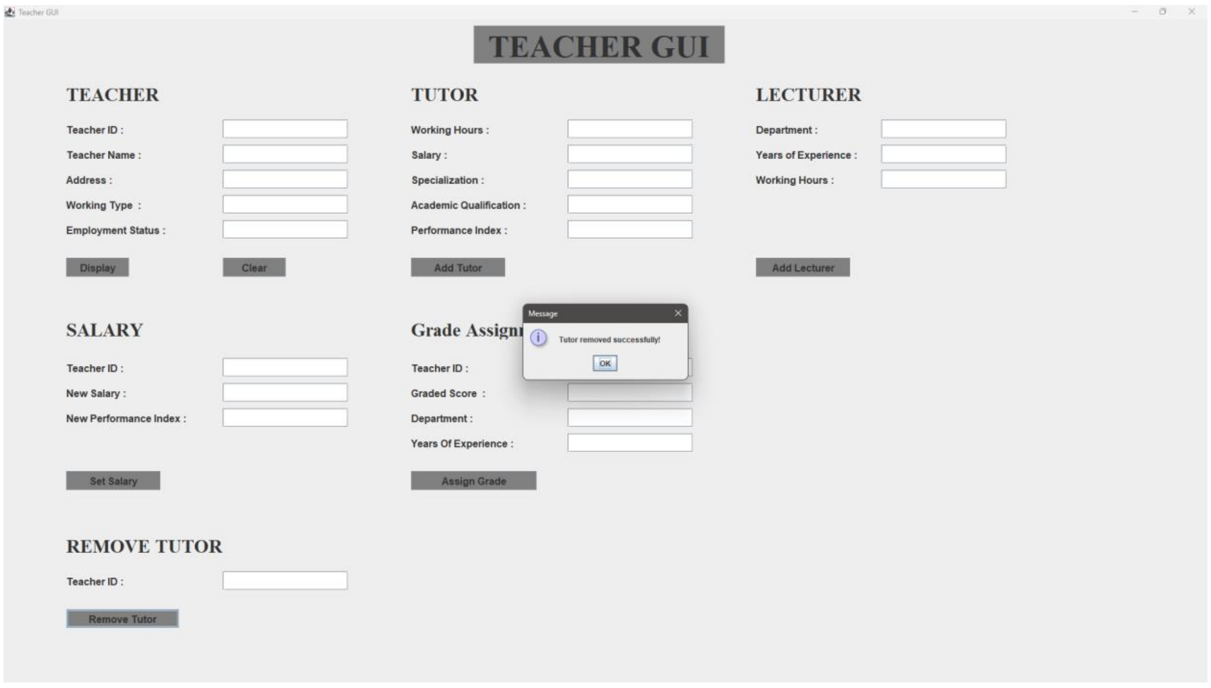
Grade Assignments

Teacher ID :
Graded Score :
Department :
Years Of Experience :

REMOVE TUTOR

Teacher ID : 444

Figure 15 : Removing Tutor



Teacher GUI

TEACHER GUI

TEACHER

Teacher ID :
Teacher Name :
Address :
Working Type :
Employment Status :

TUTOR

Working Hours :
Salary :
Specialization :
Academic Qualification :
Performance Index :

LECTURER

Department :
Years of Experience :
Working Hours :

SALARY

Teacher ID :
New Salary :
New Performance Index :

Grade Assignments

Teacher ID :
Graded Score :
Department :
Years Of Experience :

REMOVE TUTOR

Teacher ID :

Message

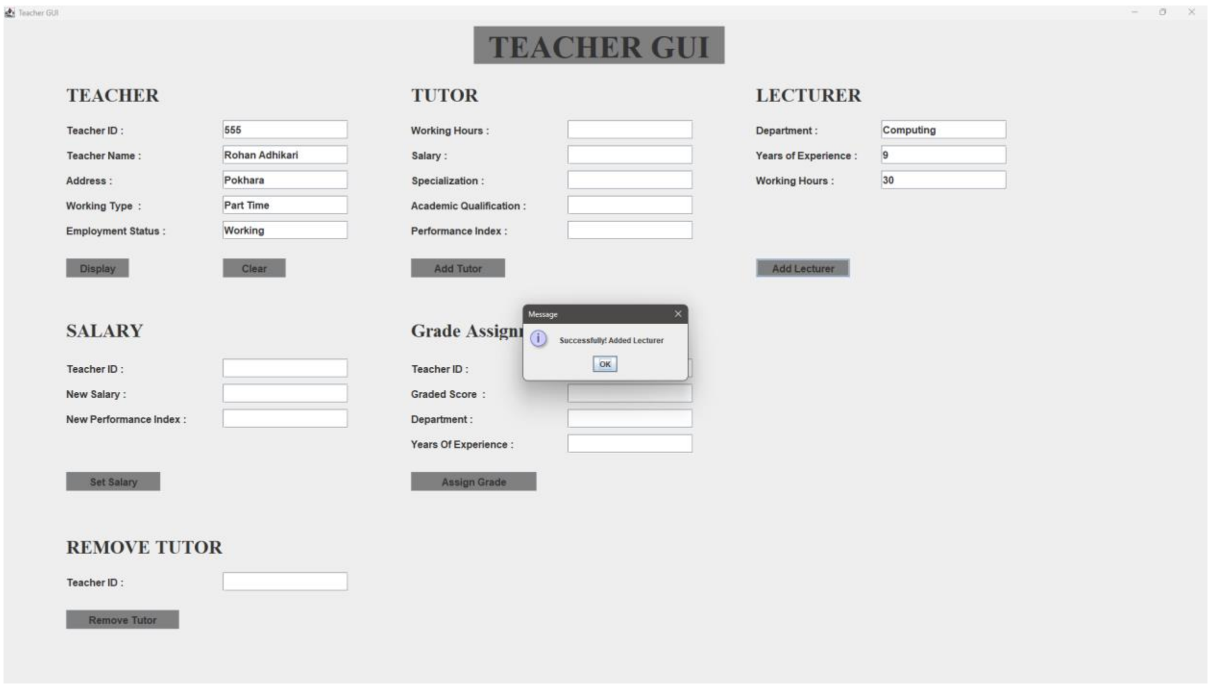
Tutor removed successfully!

Figure 16 : Tutor Removed

5.2.4. Adding Lecturer

Objective	**To add the Lecturer when Add Lecturer button is clicked and a dialog appears when button is clicked**
Action	• The TeacherGUI class was compiled and Add Lecturer button of main frame was clicked. • The following information for Lecturer was entered: • Teacher – Teacher ID: 555 – Teacher Name: Rohan Adhikari – Address: Pokhara – Working Type: Part Time – Employment Status: Working • Lecturer – Department: Computing – Years of Experience: 9 – Working Hours: 30 • And at last, Add Lecturer button is clicked
Expected result	After entering all the details of Lecturer, the values should be added correctly and a dialog box appeared that “Lecturer added Successfully” and while display button clicked all the information of Lecturer has to be in the dialog box
Actual result	An information dialog appeared with “Lecturer added Successfully” and also displayed all information of the Lecturer.
Conclusion	Test Successful

Table 7 : Adding Lecturer



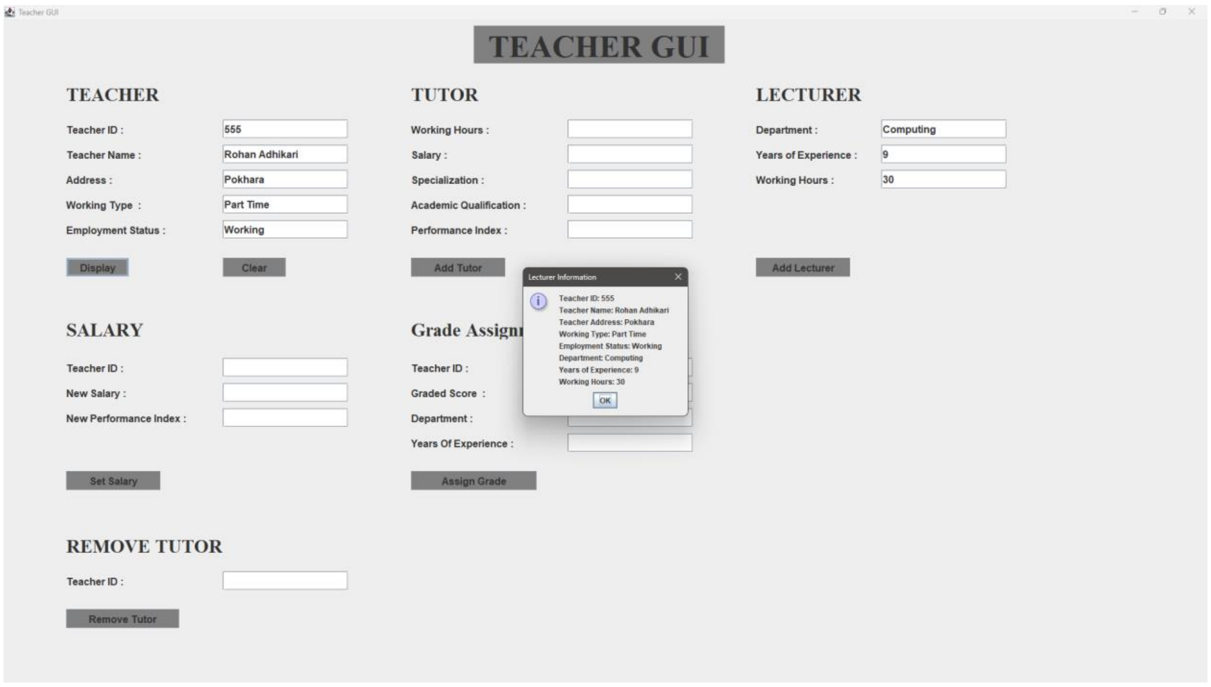
Teacher ID :	555
Teacher Name :	Rohan Adhikari
Address :	Pokhara
Working Type :	Part Time
Employment Status :	Working
Department :	Computing
Years of Experience :	9
Working Hours :	30

Message

Successfully! Added Lecturer

OK

Figure 17 : Lecturer Added



Teacher ID :	555
Teacher Name :	Rohan Adhikari
Address :	Pokhara
Working Type :	Part Time
Employment Status :	Working
Department :	Computing
Years of Experience :	9
Working Hours :	30

Lecturer Information

```
Teacher ID: 555
Teacher Name: Rohan Adhikari
Teacher Address: Pokhara
Working Type: Part Time
Employment Status: Working
Department: Computing
Years of Experience: 9
Working Hours: 30
```

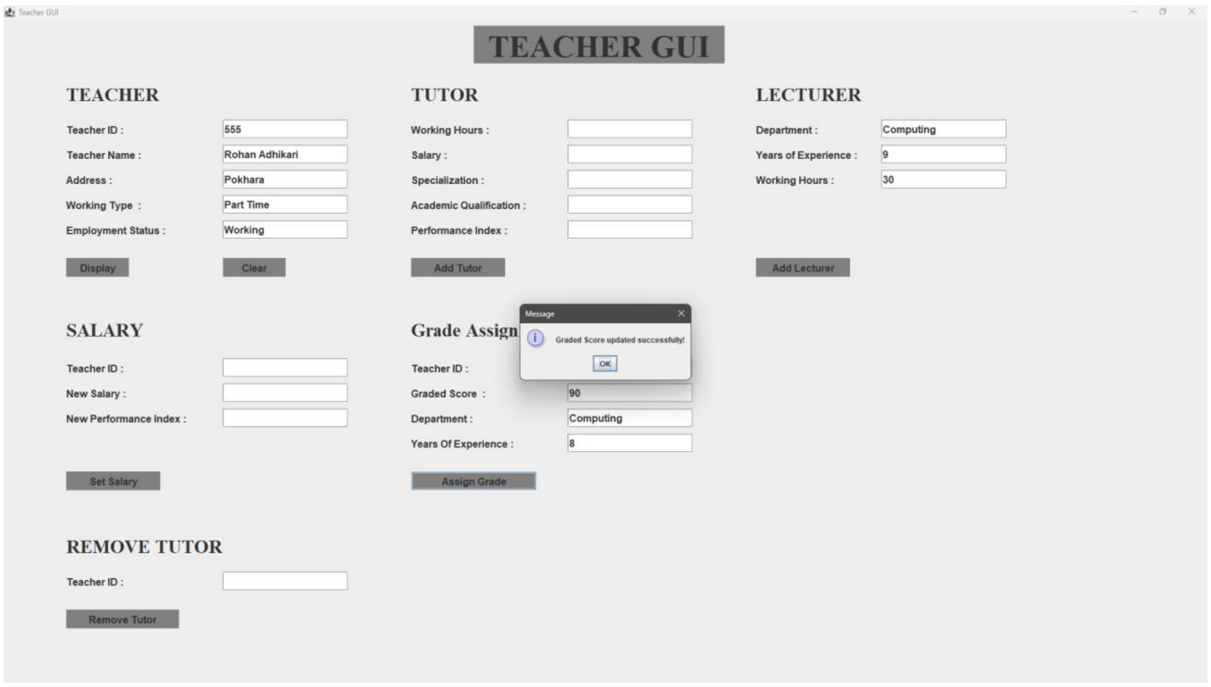
OK

Figure 18 : Lecturer information displayed in dialog box

5.2.5. Assing the Grade

Objective	**To assign the grade when Grade Assignment button is clicked and a dialog appears when button is clicked**
Action	• The TeacherGUI class was compiled and Add Lecturer button of main frame was clicked. • The following information for Lecturer was entered: • Teacher – Teacher ID: 555 – Teacher Name: Rohan Adhikari – Address: Pokhara – Working Type: Part Time – Employment Status: Working • Lecturer – Department: Computing – Years of Experience: 8 – Working Hours: 35 • Garde assignment – Teacher Id: 555 – Graded Score: 90 – Department: Computing – Years of experience: 9 • And at last, Add Lecturer button is clicked
Expected result	After entering all the details of Tutor, then it will remove the tutor of the entered tutor id from the program and a dialog box appeared with message of “Tutor Removed Successfully”
Actual result	Tutor removed and a dialog box appeared with message of “Tutor Removed Successfully”
Conclusion	Test Successful

Table 8 : Assigning the Grade



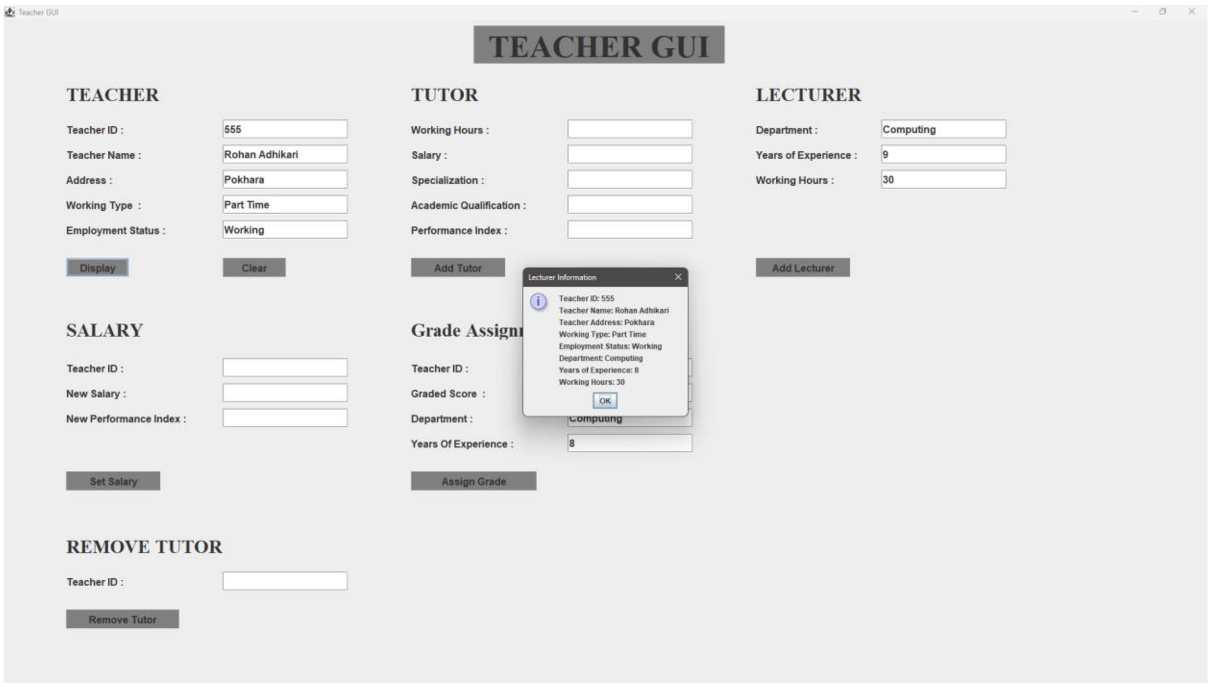
Teacher ID :	555
Teacher Name :	Rohan Adhikari
Address :	Pokhara
Working Type :	Part Time
Employment Status :	Working
Working Hours :	
Salary :	
Specialization :	
Academic Qualification :	
Performance Index :	
Department :	Computing
Years of Experience :	9
Working Hours :	30
Teacher ID :	
New Salary :	
New Performance Index :	
Teacher ID :	
Graded Score :	90
Department :	Computing
Years Of Experience :	8
Teacher ID :	

Message

Graded Score updated successfully!

OK

Figure 19 : Assigning the Garde



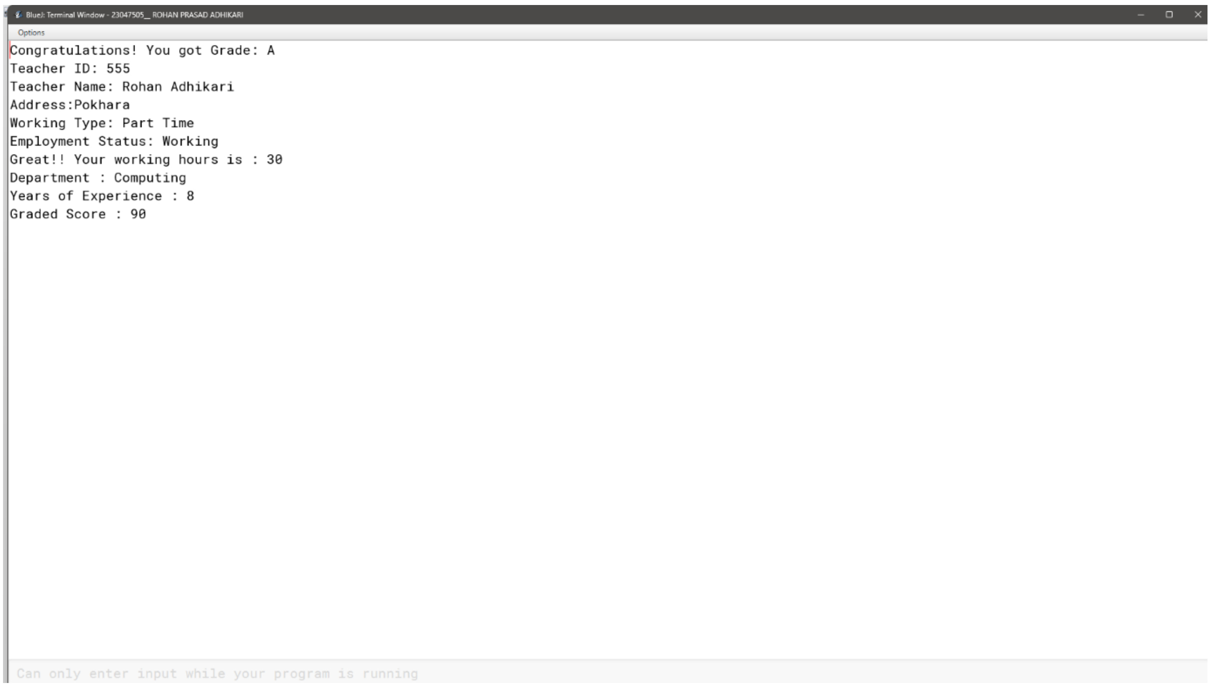
Teacher ID :	555
Teacher Name :	Rohan Adhikari
Address :	Pokhara
Working Type :	Part Time
Employment Status :	Working
Working Hours :	
Salary :	
Specialization :	
Academic Qualification :	
Performance Index :	
Department :	Computing
Years of Experience :	9
Working Hours :	30
Teacher ID :	
New Salary :	
New Performance Index :	
Teacher ID :	
Graded Score :	
Department :	Computing
Years Of Experience :	8
Teacher ID :	

Lecturer Information

```
Teacher ID: 555
Teacher Name: Rohan Adhikari
Teacher Address: Pokhara
Working Type: Part Time
Employment Status: Working
Department: Computing
Years of Experience: 8
Working Hours: 30
```

OK

Figure 20 : Grade Assigned


BlueJ: Terminal Window - 23047505__ ROHAN PRASAD ADHIKARI
Options
Can only enter input while your program is running

```
Congratulations! You got Grade: A
Teacher ID: 555
Teacher Name: Rohan Adhikari
Address:Pokhara
Working Type: Part Time
Employment Status: Working
Great!! Your working hours is : 30
Department : Computing
Years of Experience : 8
Graded Score : 90
```

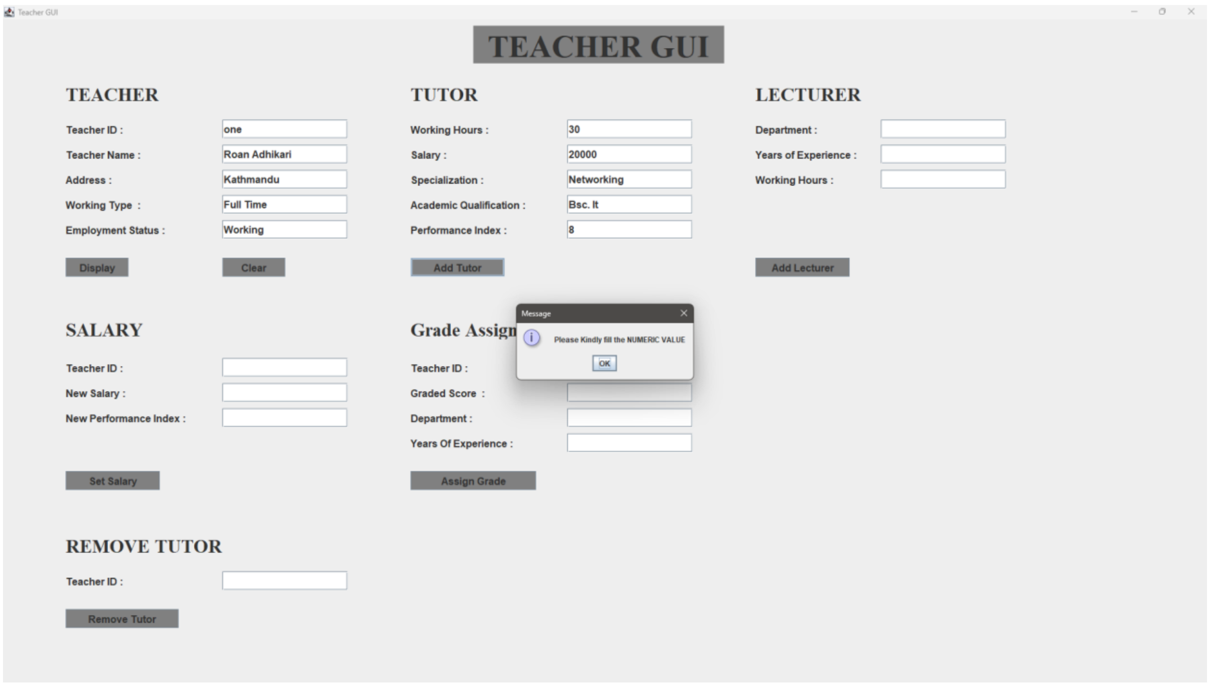
Figure 21 : Details of Lecturer Displayed

5.3. Test: 3

5.3.1. Testing with alphabet in Teacher Id in Tutor

Objective	**To test that Teacher ID of Tutor accepts only numeric values and dialog box appears on alphabetic entry**
Action	• The TeacherGUI class was compiled and Add Tutor button of main frame was clicked. • The following information for Tutor was entered: • Teacher – Teacher ID: one – Teacher Name: Roan Adhikari – Address: Kathmandu – Working Type: Full Time – Employment Status: Working • Tutor – Working Hours: 30 – Salary: 20000 – Specialization: Networking – Academic Qualification: BSc. It – Performance Index: 8 • And at last, Add Tutor button is clicked
Expected result	A warning message dialog box should be appeared on entering alphabetic values in Teacher ID text field.
Actual result	A warning message dialog box was displayed to the user after entering alphabetic values in Teacher ID text field.
Conclusion	Test Successful

Table 9 : Testing with alphabet in Teacher Id in Tutor



TEACHER

Teacher ID :	one
Teacher Name :	Roan Adhikari
Address :	Kathmandu
Working Type :	Full Time
Employment Status :	Working

TUTOR

Working Hours :	30
Salary :	20000
Specialization :	Networking
Academic Qualification :	Bsc. It
Performance Index :	8

LECTURER

Department :	
Years of Experience :	
Working Hours :	

SALARY

Teacher ID :	
New Salary :	
New Performance Index :	

Grade Assign

Teacher ID :	
Graded Score :	
Department :	
Years Of Experience :	

REMOVE TUTOR

Teacher ID :	

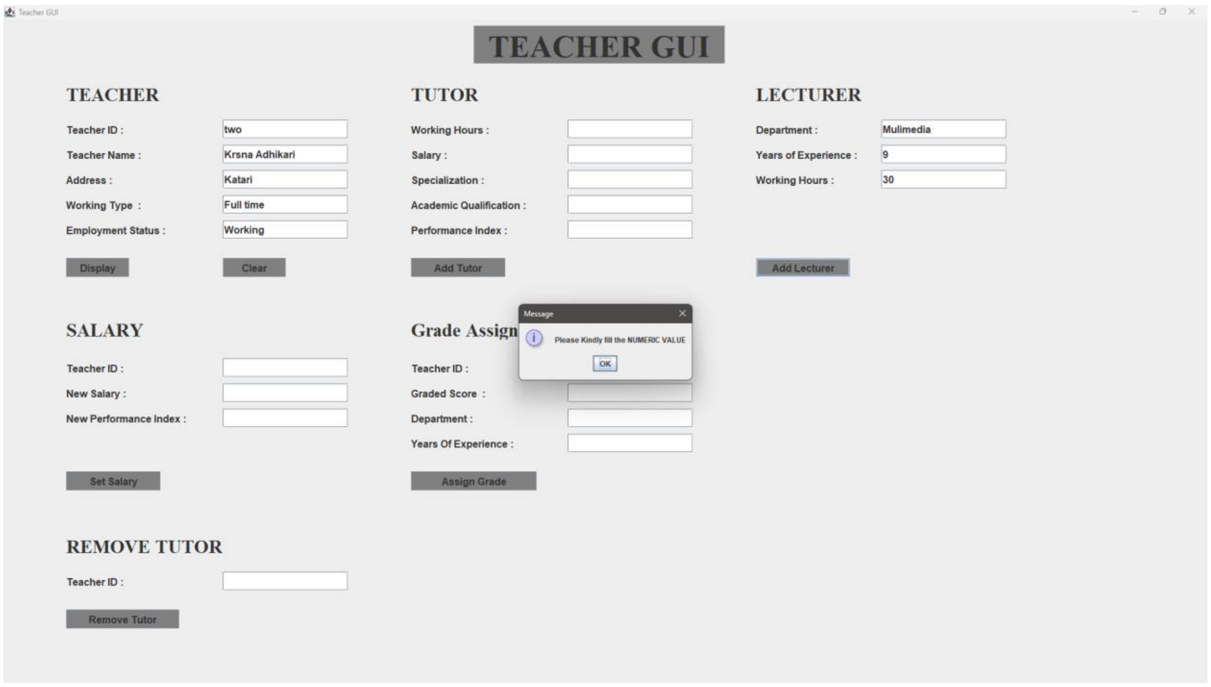
Please Kindly fill the NUMERIC VALUE

Figure 22 : Testing with alphabet in Teacher Id in Tutor

5.3.2. Testing with alphabet in Teacher Id in Lecturer

Objective	**To test that Teacher ID of Lecturer accepts only numeric values and dialog box appears on alphabetic entry**
Action	• The TeacherGUI class was compiled and Add Lecturer button of main frame was clicked. • The following information for Lecturer was entered: • Teacher – Teacher ID: Two – Teacher Name: Krsna Adhikari – Address: Katari – Working Type: Full Time – Employment Status: Working • Lecturer – Department: Multimedia – Years of Experience: 9 – Working Hours: 30 • And at last, Add Lecturer button is clicked
Expected result	A warning message dialog box should be appeared on entering alphabetic values in Teacher ID text field.
Actual result	A warning message dialog box was displayed to the user after entering alphabetic values in Teacher ID text field.
Conclusion	Test Successful

Table 10 : Testing with alphabet in Teacher Id in Lecturer



Teacher ID : two
Teacher Name : Krsna Adhikari
Address : Katari
Working Type : Full time
Employment Status : Working

Working Hours :
Salary :
Specialization :
Academic Qualification :
Performance Index :

Department : Mulimedia
Years of Experience : 9
Working Hours : 30

Please Kindly fill the NUMERIC VALUE

Teacher ID :
New Salary :
New Performance Index :

Teacher ID :
Graded Score :
Department :
Years Of Experience :

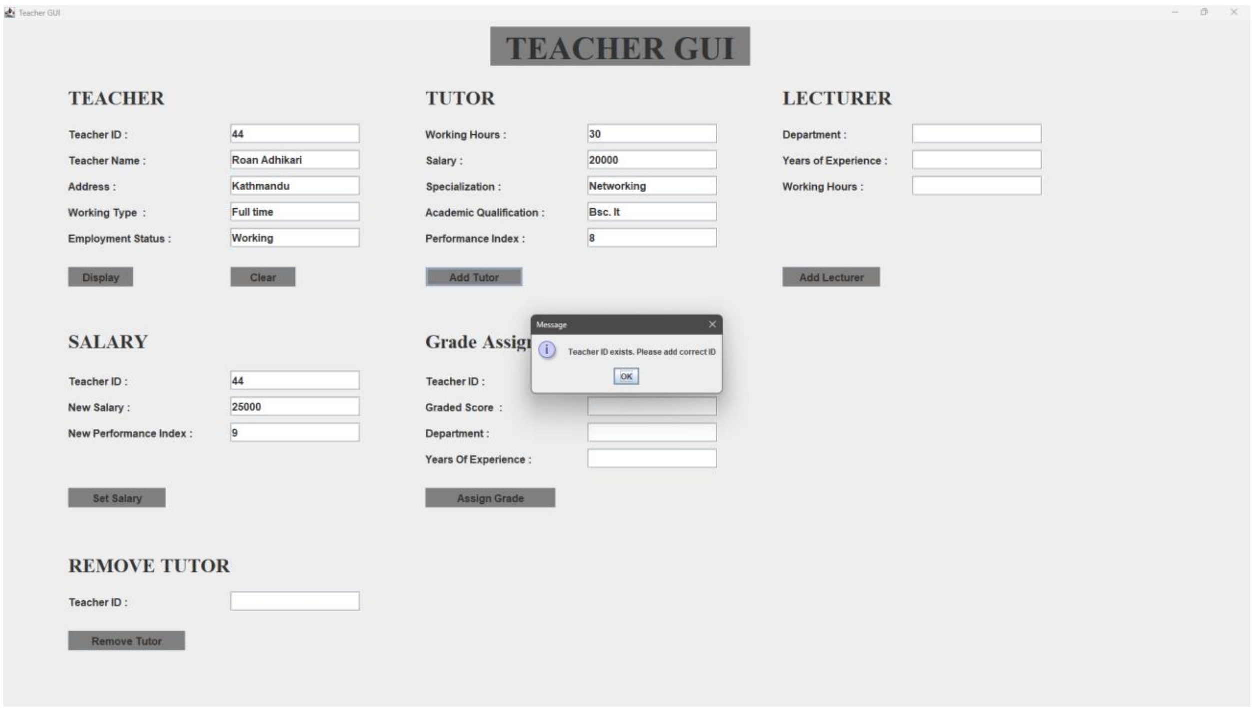
Teacher ID :

Figure 23 : Testing with alphabet in Teacher Id in Lecturer

5.3.3. Testing existing Teacher ID in Tutor

Objective	**To test if appropriate dialog box is displayed while trying to add Tutor with existing Teacher ID**
Action	• The TeacherGUI class was compiled and Add Tutor button of main frame was clicked. • The following information for Tutor was entered: • Teacher – Teacher ID: 44 – Teacher Name: Roan Adhikari – Address: Kathmandu – Working Type: Full Time – Employment Status: Working • Tutor – Working Hours: 30 – Salary: 20000 – Specialization: Networking – Academic Qualification: BSc. It – Performance Index: 8 • Salary – Teacher Id: – New Salary: 25000 – New Performance Index: 9 • And at last, Add Tutor button is clicked
Expected result	A warning message dialog box should be appeared on entering duplicate Teacher ID.
Actual result	A warning message dialog box was displayed to the user after entering duplicate Teacher ID.
Conclusion	Test Successful

Table 11 : Testing existing Teacher ID in Tutor



TEACHER

Teacher ID :	44
Teacher Name :	Roan Adhikari
Address :	Kathmandu
Working Type :	Full time
Employment Status :	Working

TUTOR

Working Hours :	30
Salary :	20000
Specialization :	Networking
Academic Qualification :	Bsc. It
Performance Index :	8

LECTURER

Department :	
Years of Experience :	
Working Hours :	

SALARY

Teacher ID :	44
New Salary :	25000
New Performance Index :	9

Grade Assign

Teacher ID :	
Graded Score :	
Department :	
Years Of Experience :	

REMOVE TUTOR

Teacher ID :	

Message

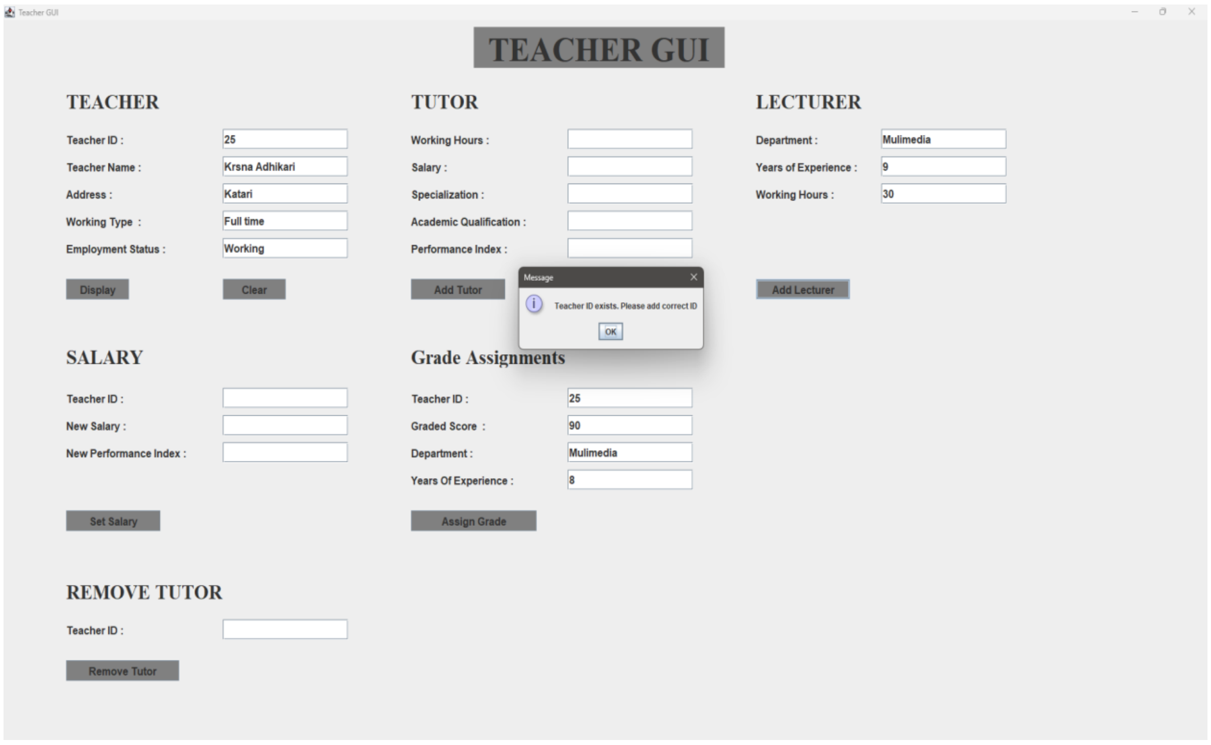
Teacher ID exists. Please add correct ID

Figure 24 : Testing existing Teacher ID in Tutor

5.3.4. Testing existing Teacher ID in Lecturer

Objective	**To test if appropriate dialog box is displayed while trying to add Lecturer with existing Teacher ID**
Action	• The TeacherGUI class was compiled and Add Lecturer button of main frame was clicked. • The following information for Lecturer was entered: • Teacher – Teacher ID: 25 – Teacher Name: Krsna Adhikari – Address: Katari – Working Type: Full Time – Employment Status: Working – Department: Multimedia – Years of Experience: 9 – Working Hours: 30 • And at last, Add Tutor button is clicked
Expected result	A warning message dialog box should be appeared on entering duplicate Teacher ID.
Actual result	A warning message dialog box was displayed to the user after entering duplicate Teacher ID.
Conclusion	Test Successful

Table 12 : Testing existing Teacher ID in Lecturer



Teacher GUI

TEACHER GUI

TEACHER

Teacher ID :	25
Teacher Name :	Krsna Adhikari
Address :	Katari
Working Type :	Full time
Employment Status :	Working

TUTOR

Working Hours :	
Salary :	
Specialization :	
Academic Qualification :	
Performance Index :	

LECTURER

Department :	Mulimedia
Years of Experience :	9
Working Hours :	30

Message

Teacher ID exists. Please add correct ID

SALARY

Teacher ID :	
New Salary :	
New Performance Index :	

Grade Assignments

Teacher ID :	25
Graded Score :	90
Department :	Mulimedia
Years Of Experience :	8

REMOVE TUTOR

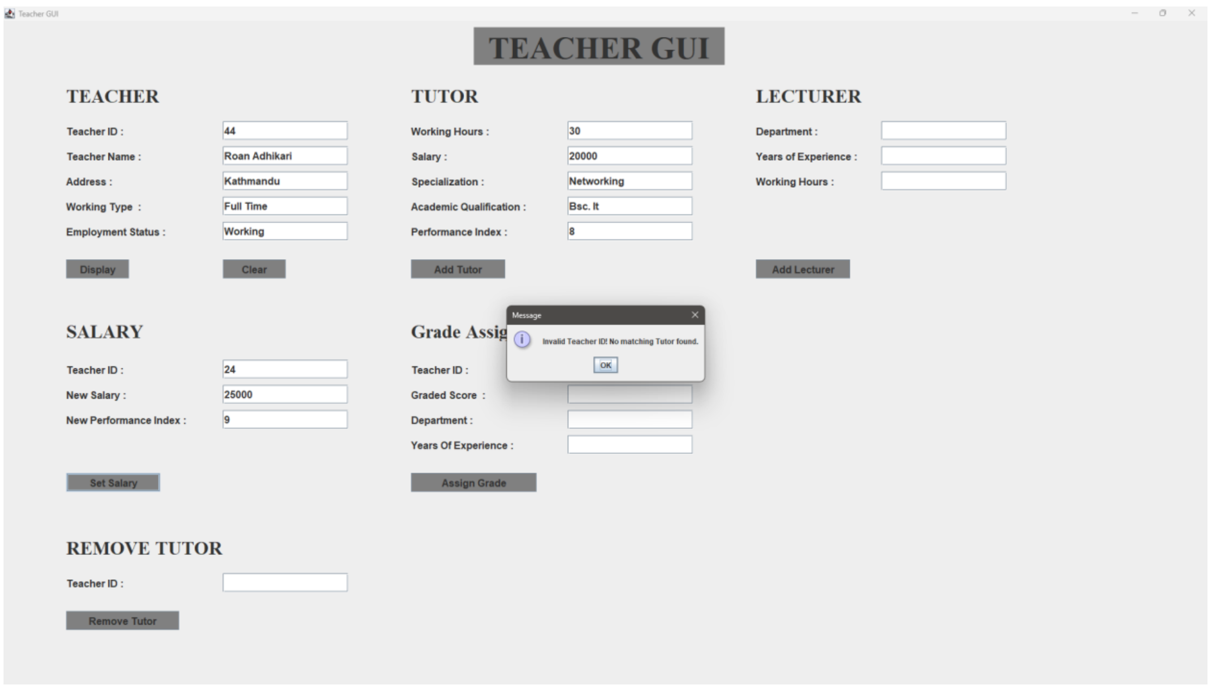
Teacher ID :	

Figure 25 : Testing existing Teacher ID in Lecturer

5.3.5. Wrong Teacher Id for Tutor in Set Salary

Objective	**To test if appropriate dialog box is displayed while trying to enter wrong Teacher ID in the text field in Tutor for set salary**
Action	Wrong Teacher ID was entered while setting the new salary for Tutor.
Expected result	A warning message dialog box should be appeared on entering wrong Teacher ID.
Actual result	A warning message dialog box was displayed to the user after entering wrong Teacher ID.
Conclusion	Test Successful

Table 13 : Wrong Teacher Id for Tutor in Set Salary



Teacher ID : 44
Teacher Name : Roan Adhikari
Address : Kathmandu
Working Type : Full Time
Employment Status : Working

Working Hours : 30
Salary : 20000
Specialization : Networking
Academic Qualification : Bsc. It
Performance Index : 8

Department :
Years of Experience :
Working Hours :

SALARY
Teacher ID : 24
New Salary : 25000
New Performance Index : 9

Teacher ID :
Graded Score :
Department :
Years Of Experience :

REMOVE TUTOR
Teacher ID :

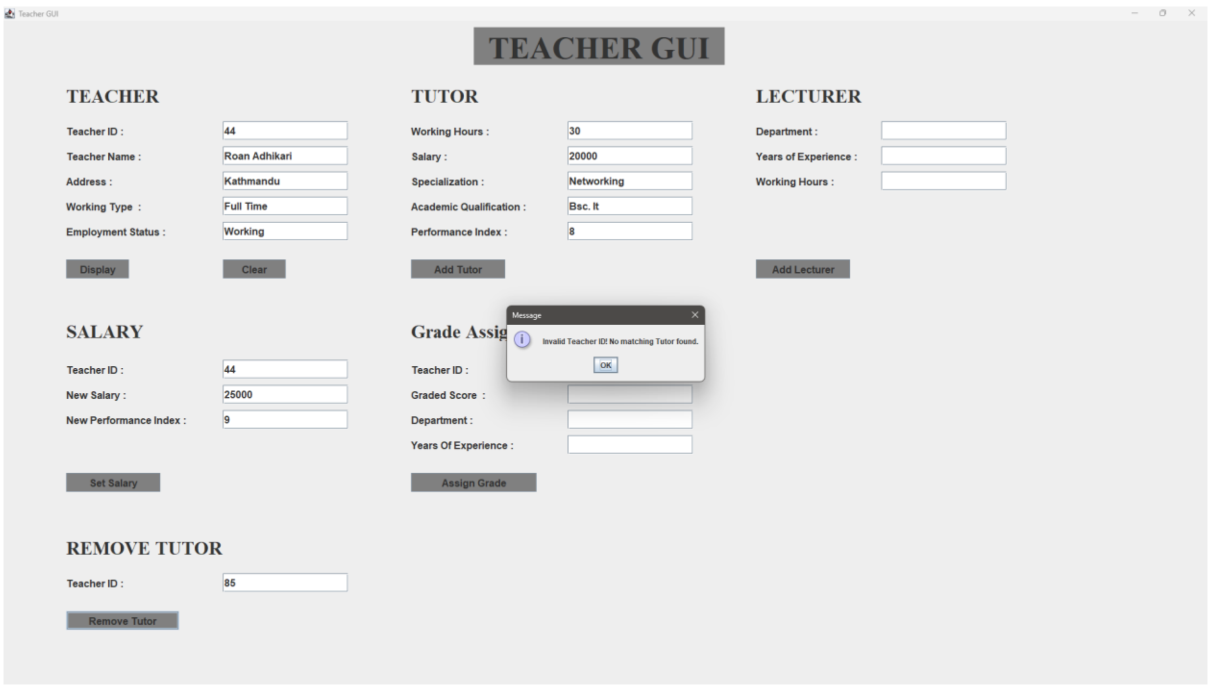
Message
Invalid Teacher ID! No matching Tutor found.
OK

Figure 26 : Wrong Teacher Id for Tutor in Set Salary

5.3.6. Wrong Teacher Id for Tutor in Remove Tutor

Objective	**To test if appropriate dialog box is displayed while trying to enter wrong Teacher ID in the text field in Tutor for Remove Tutor**
Action	Wrong Teacher ID was entered while removing Tutor.
Expected result	A warning message dialog box should be appeared on entering wrong Teacher ID.
Actual result	A warning message dialog box was displayed to the user after entering wrong Teacher ID.
Conclusion	Test Successful

Table 14 : Wrong Teacher Id for Tutor in Remove Tutor



TEACHER

Teacher ID : 44
Teacher Name : Roan Adhikari
Address : Kathmandu
Working Type : Full Time
Employment Status : Working

TUTOR

Working Hours : 30
Salary : 20000
Specialization : Networking
Academic Qualification : Bsc. It
Performance Index : 8

LECTURER

Department :
Years of Experience :
Working Hours :

SALARY

Teacher ID : 44
New Salary : 25000
New Performance Index : 9

Grade Assig

Teacher ID :
Graded Score :
Department :
Years Of Experience :

REMOVE TUTOR

Teacher ID : 85

Message

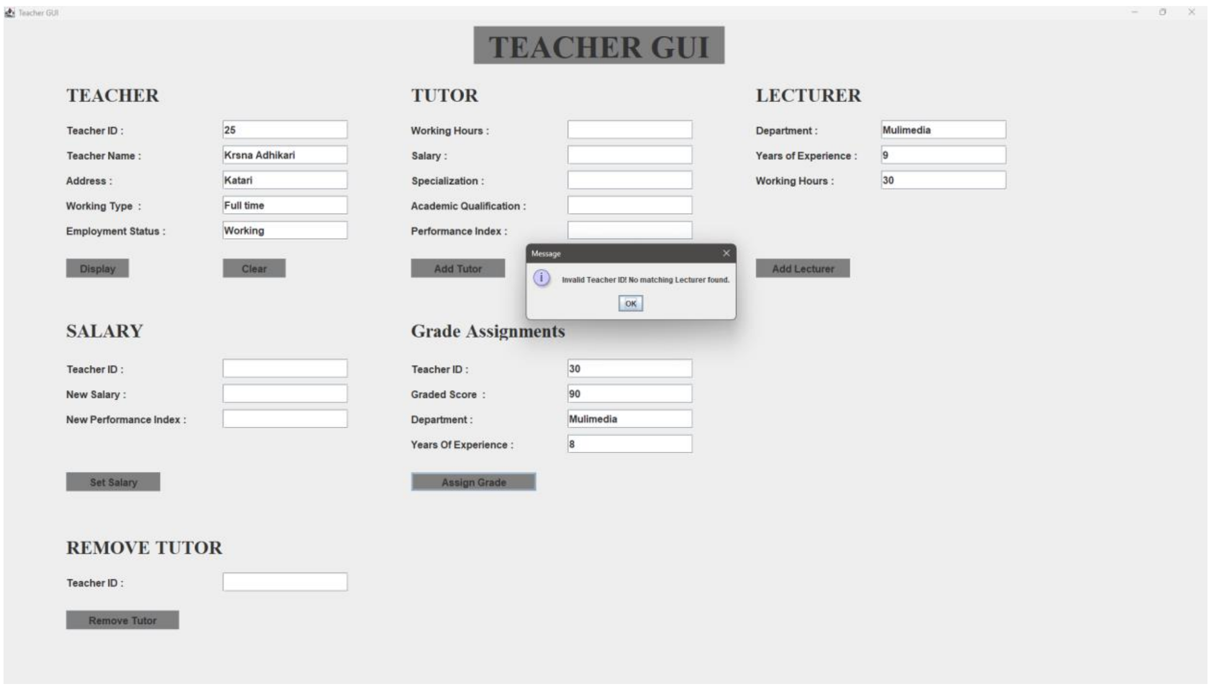
Invalid Teacher ID! No matching Tutor found.

Figure 27 : Wrong Teacher Id for Tutor in Remove Tutor

5.3.7. Wrong Teacher Id for Lecturer while assigning the grade

Objective	**To test if appropriate dialog box is displayed while trying to enter wrong Teacher ID in the text field in Lecturer while assigning the grade**
Action	Wrong Teacher ID was entered while assigning the grade.
Expected result	A warning message dialog box should be appeared on entering wrong Teacher ID.
Actual result	A warning message dialog box was displayed to the user after entering wrong Teacher ID.
Conclusion	Test Successful

Table 15 : Wrong Teacher Id in Graded Assignment


Teacher GUI

TEACHER GUI

TEACHER

Teacher ID : 25
Teacher Name : Krsna Adhikari
Address : Katari
Working Type : Full time
Employment Status : Working

TUTOR

Working Hours :
Salary :
Specialization :
Academic Qualification :
Performance Index :

LECTURER

Department : Mulimedia
Years of Experience : 9
Working Hours : 30

Message

Invalid Teacher ID! No matching Lecturer found.

SALARY

Teacher ID :
New Salary :
New Performance Index :

Grade Assignments

Teacher ID : 30
Graded Score : 90
Department : Mulimedia
Years Of Experience : 8

REMOVE TUTOR

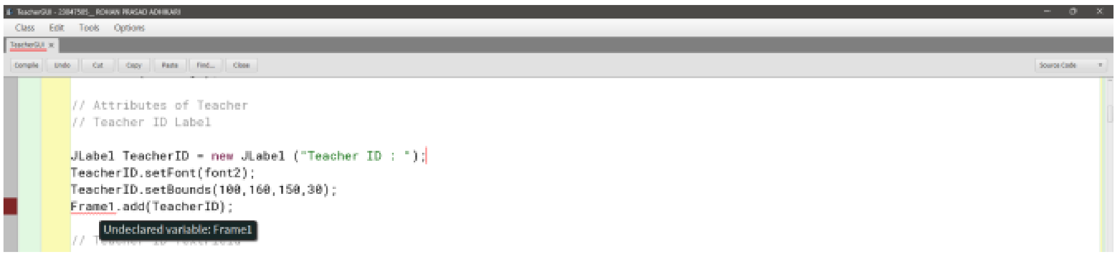
Teacher ID :

Figure 28 : Wrong Teacher Id in Graded Assignment

6. Error

6.1. Error 1: Syntax Error

6.1.1. Error Detection



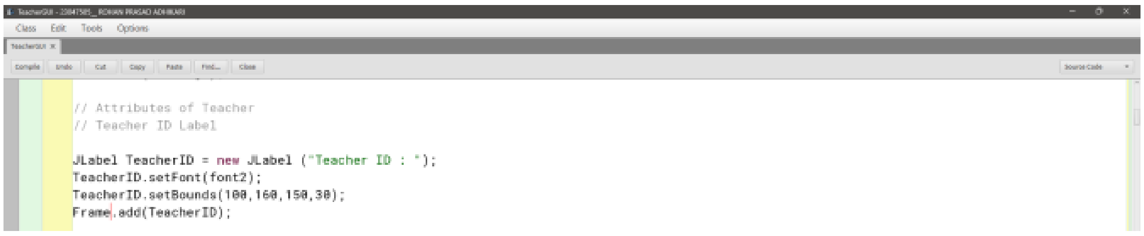
```
// Attributes of Teacher
// Teacher ID Label

JLabel TeacherID = new JLabel ("Teacher ID : ");
TeacherID.setFont(font2);
TeacherID.setBounds(100,160,150,30);
Frame1.add(TeacherID);
```

Undeclared variable: Frame1

Figure 29 : Error detection of syntax error

This error is the syntax error and the major reason behind this error is the undeclared variable After that, I try to compile the program without solving an error.

6.1.2. Error Correction



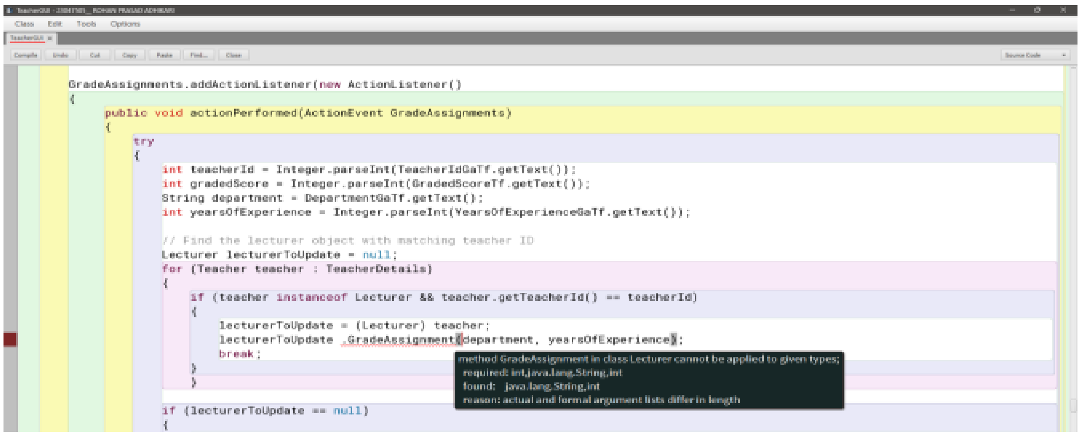
```
// Attributes of Teacher
// Teacher ID Label

JLabel TeacherID = new JLabel ("Teacher ID : ");
TeacherID.setFont(font2);
TeacherID.setBounds(100,160,150,30);
Frame.add(TeacherID);
```

Figure 30 : Error correction of syntax error

This error is the syntax error and the major reason behind this error is the undeclared variable. So, I checked the program and there is Frame instead of Frame1. After that, I compiled the program and it complied.

6.2. Error 2: Semantic Error

6.2.1. Error Detection



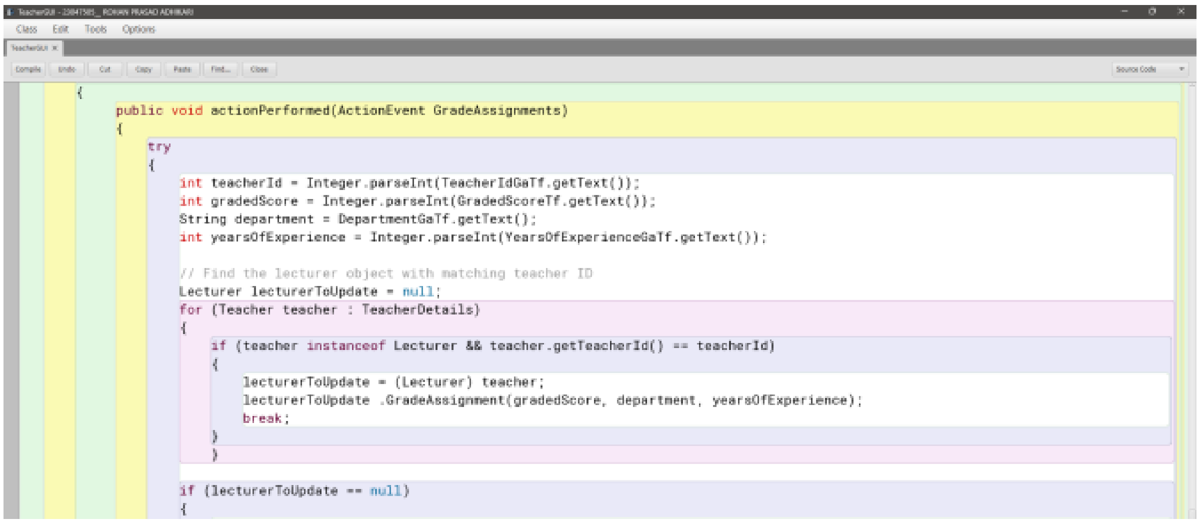
```
GradeAssignments.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent GradeAssignments)
    {
        try
        {
            int teacherId = Integer.parseInt(TeacherIdGaTf.getText());
            int gradedScore = Integer.parseInt(GradedScoreTf.getText());
            String department = DepartmentGaTf.getText();
            int yearsOfExperience = Integer.parseInt(YearsOfExperienceGaTf.getText());

            // Find the lecturer object with matching teacher ID
            Lecturer lecturerToUpdate = null;
            for (Teacher teacher : TeacherDetails)
            {
                if (teacher instanceof Lecturer && teacher.getTeacherId() == teacherId)
                {
                    lecturerToUpdate = (Lecturer) teacher;
                    lecturerToUpdate .GradeAssignment(department, yearsOfExperience);
                    break;
                }
            }

            if (lecturerToUpdate == null)
            {
```

```
method GradeAssignment in class Lecturer cannot be applied to given types;
 required: int,java.lang.String,int
 found:    java.lang.String,int
 reason: actual and formal argument lists differ in length
```

Figure 31 : Error detection of Semantic error

This error is the Semantic error and the major reason behind this error is the whole datatype are not mentioned. After that, I try to compile the program without solving an error

6.2.2. Error Correction



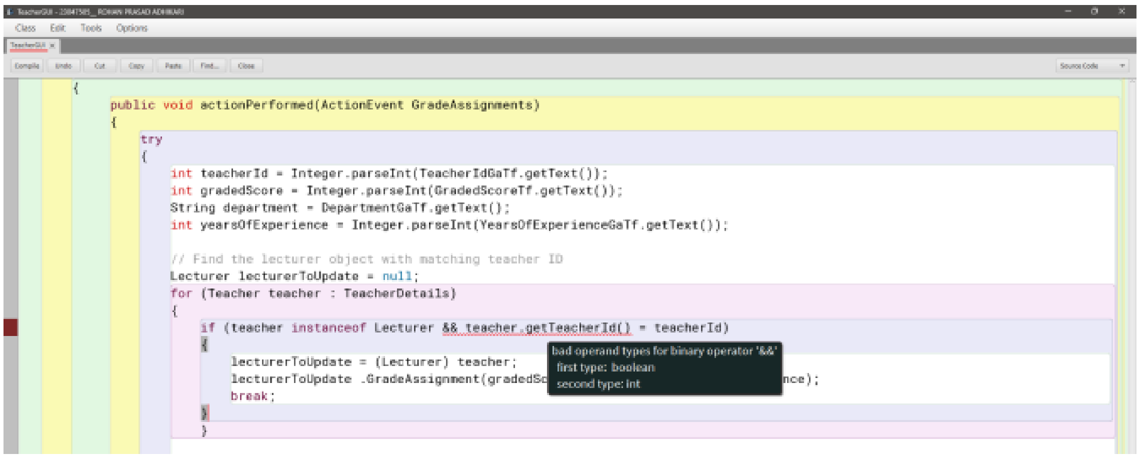
```
{
    public void actionPerformed(ActionEvent GradeAssignments)
    {
        try
        {
            int teacherId = Integer.parseInt(TeacherIdGaTf.getText());
            int gradedScore = Integer.parseInt(GradedScoreTf.getText());
            String department = DepartmentGaTf.getText();
            int yearsOfExperience = Integer.parseInt(YearsOfExperienceGaTf.getText());

            // Find the lecturer object with matching teacher ID
            Lecturer lecturerToUpdate = null;
            for (Teacher teacher : TeacherDetails)
            {
                if (teacher instanceof Lecturer && teacher.getTeacherId() == teacherId)
                {
                    lecturerToUpdate = (Lecturer) teacher;
                    lecturerToUpdate .GradeAssignment(gradedScore, department, yearsOfExperience);
                    break;
                }
            }

            if (lecturerToUpdate == null)
            {
```

Figure 32 : Error correction of Semantic error

This error is the Semantic error and the major reason behind this error is the whole datatype are not mentioned. As I added gradedScore which is the missing datatype in the above figure. After that, I compile the program without any error.

6.3. Error 3: Logical Error

6.3.1. Error Detection



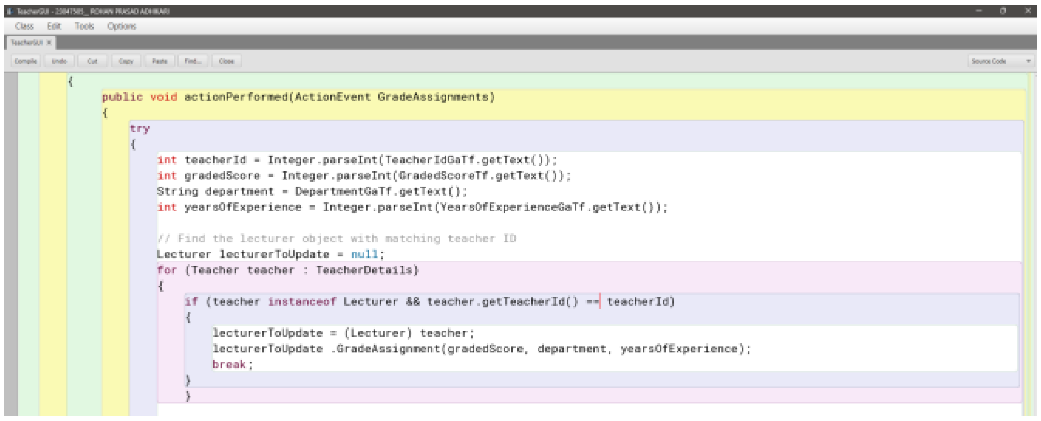
```
{
    public void actionPerformed(ActionEvent GradeAssignments)
    {
        try
        {
            int teacherId = Integer.parseInt(TeacherIdGaTf.getText());
            int gradedScore = Integer.parseInt(GradedScoreTf.getText());
            String department = DepartmentGaTf.getText();
            int yearsOfExperience = Integer.parseInt(YearsOfExperienceGaTf.getText());

            // Find the lecturer object with matching teacher ID
            Lecturer lecturerToUpdate = null;
            for (Teacher teacher : TeacherDetails)
            {
                if (teacher instanceof Lecturer && teacher.getTeacherId() = teacherId)
                {
                    lecturerToUpdate = (Lecturer) teacher;
                    lecturerToUpdate .GradeAssignment(gradedSc
                    break;
                }
            }
```

```
bad operand types for binary operator '&&'
first type: boolean
second type: int
```

Figure 33 : Error detection of logical error

This error is the logical error and the major reason behind this error is the I only entered one equals to symbol and I try to compile the program without solving an error.

6.3.2. Error Correction



```
{
    public void actionPerformed(ActionEvent GradeAssignments)
    {
        try
        {
            int teacherId = Integer.parseInt(TeacherIdGaTf.getText());
            int gradedScore = Integer.parseInt(GradedScoreTf.getText());
            String department = DepartmentGaTf.getText();
            int yearsOfExperience = Integer.parseInt(YearsOfExperienceGaTf.getText());

            // Find the lecturer object with matching teacher ID
            Lecturer lecturerToUpdate = null;
            for (Teacher teacher : TeacherDetails)
            {
                if (teacher instanceof Lecturer && teacher.getTeacherId() == teacherId)
                {
                    lecturerToUpdate = (Lecturer) teacher;
                    lecturerToUpdate .GradeAssignment(gradedScore, department, yearsOfExperience);
                    break;
                }
            }
```

Figure 34 : Error Correction of logical error

This error is the logical error and the major reason behind this error is the I only entered one equals to symbol and I try to compile the program without solving an error. So, I added one more equal to symbol and it didn’t show any error and I compiled the program without any error.

7. Conclusion

From this coursework, I learned the main concepts of making Window based applications with proper functionality. I gathered more ideas and information related to Swing, Abstract Window Toolkit (AWT), Exception and Event Handling, Object Casting, Object Oriented Programming (OOP). As an individual undertaking this coursework, I got an opportunity to test my skills, ideas, and understanding of programming skills. I got to know the information on how big tech-companies build desktop applications, database applications, business applications, educational software, and many others software.

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 doing this coursework, I knew just the basic concept of Object-oriented concept, but I did not know how to implement it in real- world problems. I couldn't make the proper GUI even though I had enough knowledge about Swing due to improper guidelines. I was so confused where and how to start. But after too much research, dedication, and hard work I created the basic design for this coursework. In other hands, my teachers and teammates helped me a lot at the journey of this coursework. After completing the GUI part, I got to know many ideas regarding how to create basic design before starting any projects or coursework.

In the case of Event Handling, I faced a bit of problems while calling methods of Tutor class and Lecturer class using object casting concepts. But after proper practice and solving the same problem, I solved the object casting problem in the coursework. Sometimes, I faced syntax error while implementing interface (ActionListener). But at the end, I completed my coursework after giving my full focus, time and doing some sacrifices. The concepts and knowledge I used in this course will help me in the future to get placements at big tech-companies like Google, Twitter, and many mores. This coursework also helped me to increase problem solving skill, creativity and thinking capacity of new ideas.

8. Bibliography

Anon., 2023. *Microsoft Word.* [Online]
Available at: https://www.microsoft.com/en/microsoft-365/word?market=af
[Accessed 28 April 2024].

Kölling , M. & Rosenberg, J., 20 June 2023. *BlueJ.* [Online]
Available at: https://www.bluej.org
[Accessed 24 Janurary 2024].

Williams, E., February 2020. *Draw IO.* [Online]
Available at: https://app.diagrams.net/
[Accessed 24 January 2024].

9. Appendix

9.1. Teacher GUI

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

 class TeacherGUI
{
   ArrayList<Teacher>TeacherDetails = new ArrayList<>();
   public void Gui()
   {
       // Adding Panel

       JPanel Panel = new JPanel();
       Panel. setBounds(750,10,400,60);
       Panel.setBackground(Color.GRAY);


       // Adding Frame

       JFrame Frame = new JFrame(" Teacher GUI");
       Frame.setSize(2000,2000);
       Frame.setLayout(null);
       Frame.setVisible(true);
       Frame.getContentPane().add(Panel);
```

```
// Font Customization
Font font = new Font("Times New Roman",Font.BOLD,50);
Font font1 = new Font("Times New Roman",Font.BOLD,30);
Font font2 = new Font("Bahnschrift SemiBold",Font.BOLD,16);

//Main Heading

JLabel Heading = new JLabel("TEACHER GUI");
Heading.setFont(font);
Heading. setBounds(750,10,360,40);
Panel.add(Heading);



// Adding label to Teacher

JLabel Heading1 = new JLabel("TEACHER");
Heading1.setFont(font1);
Heading1. setBounds(100,80,300,80);
Frame.add(Heading1);

// Attributes of Teacher
// Teacher ID Label

JLabel TeacherID = new JLabel ("Teacher ID : ");
TeacherID.setFont(font2);
TeacherID.setBounds(100,160,150,30);
Frame.add(TeacherID);

// Teacher ID TextField
```

```
JTextField TeacherId = new JTextField();
TeacherId.setFont(font2);
TeacherId.setBounds(350,160,200,30);
Frame.add(TeacherId);



//
// Teacher Name (Label)

JLabel TeacherName = new JLabel ("Teacher Name : ");
TeacherName.setFont(font2);
TeacherName.setBounds(100,200,150,30);
Frame.add(TeacherName);

// Teacher Name (TextField)

JTextField TeacherN = new JTextField();
TeacherN.setFont(font2);
TeacherN.setBounds(350,200,200,30);
Frame.add(TeacherN);

//
// Teacher Address (Label)

JLabel TeacherAddress = new JLabel ("Address : ");
TeacherAddress.setFont(font2);
TeacherAddress.setBounds(100,240,150,30);
Frame.add(TeacherAddress);
```

```
// Teacher Address (TextField)

JTextField TeacherA = new JTextField();
TeacherA.setFont(font2);
TeacherA.setBounds(350,240,200,30);
Frame.add(TeacherA);

//
// Teacher Working Type (Label)

JLabel TeacherWorkingType  = new JLabel ("Working Type  : ");
TeacherWorkingType .setFont(font2);
TeacherWorkingType .setBounds(100,280,150,30);
Frame.add(TeacherWorkingType);

// Teacher Working Type (TextField)

JTextField TeacherWt = new JTextField();
TeacherWt.setFont(font2);
TeacherWt.setBounds(350,280,200,30);
Frame.add(TeacherWt);

//
// Teacher Employment Status (Label)

JLabel TeacherEmploymentStatus = new JLabel ("Employment Status : ");
TeacherEmploymentStatus.setFont(font2);
TeacherEmploymentStatus.setBounds(100,320,180,30);
Frame.add(TeacherEmploymentStatus);
```

```
// Teacher Employment Status (TextField)

JTextField TeacherEs = new JTextField();
TeacherEs.setFont(font2);
TeacherEs.setBounds(350,320,200,30);
Frame.add(TeacherEs);

//
// Buttons
//Button for Display

JButton Display = new JButton("Display");
Display.setFont(font2);
Display.setBackground(Color.GRAY);
Display.setBounds(100,380,100,30);
Frame.add(Display);


// Button For Clear

JButton Clear = new JButton("Clear");
Clear.setFont(font2);
Clear.setBackground(Color.GRAY);
Clear.setBounds(350,380,100,30);
Frame.add(Clear);


//
// Button Action
```

```
    // Action Of Button Display


     Display.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent Display)
        {
            for (Teacher teacher : TeacherDetails) {
                if (teacher instanceof Tutor)
                {
                    Tutor tutor = (Tutor) teacher;
                    String tutorInfo = "Teacher ID: " + tutor.getTeacherId() +
                            "\nTeacher Name: " + tutor.getTeacherName() +
                            "\nTeacher Address: " + tutor.getAddress() +
                            "\nWorking Type: " + tutor.getWorkingType() +
                            "\nEmployment Status: " + tutor.getEmploymentStatus() +
                            "\nWorking Hours: " + tutor.getWorkingHours() +
                            "\nSalary: " + tutor.getSalary() +
                            "\nSpecialization: " + tutor.getSpecialization() +
                            "\nAcademic Qualification: " +
tutor.getAcademicQualifications() +
                            "\nPerformance Index: " + tutor.getPerformanceIndex();
                    tutor.displayDetails();
                    JOptionPane.showMessageDialog(null, tutorInfo, "Tutor Information",
JOptionPane.INFORMATION_MESSAGE);



                }
                else if (teacher instanceof Lecturer)
                {
                    Lecturer lecturer = (Lecturer) teacher;
                    String lecturerInfo = "Teacher ID: " + lecturer.getTeacherId() +
                    "\nTeacher Name: " + lecturer.getTeacherName() +
                    "\nTeacher Address: " + lecturer.getAddress() +
                    "\nWorking Type: " + lecturer.getWorkingType() +
                    "\nEmployment Status: " + lecturer.getEmploymentStatus() +
                    "\nDepartment: " + lecturer.getDepartment() +
                    "\nYears of Experience: " + lecturer.getYearsOfExperience()
+
                    "\nWorking Hours: " + lecturer.getWorkingHours();
            lecturer.display();
            JOptionPane.showMessageDialog(null, lecturerInfo, "Lecturer
Information", JOptionPane.INFORMATION_MESSAGE);
          }
        }



      }
    });






    //
    // Adding Tutor
    // Adding label  Tutor

    JLabel Heading2 = new JLabel("TUTOR");
    Heading2.setFont(font1);
    Heading2. setBounds(650,80,300,80);
    Frame.add(Heading2);
```

```
// Attributes of Tutor

//
// Tutor Working Hours Label

JLabel WorkingHours = new JLabel ("Working Hours : ");
WorkingHours.setFont(font2);
WorkingHours.setBounds(650,160,180,30);
Frame.add(WorkingHours);

// Tutor Working Hours TextField

JTextField WorkingHoursTfT = new JTextField();
WorkingHoursTfT.setFont(font2);
WorkingHoursTfT.setBounds(900,160,200,30);
Frame.add(WorkingHoursTfT);



 //
// Tutor Salary Label

JLabel Salary = new JLabel ("Salary : ");
Salary.setFont(font2);
Salary.setBounds(650,200,180,30);
Frame.add(Salary);

// Tutor Salary TextField

JTextField SalaryTf = new JTextField();
```

```
SalaryTf.setFont(font2);
SalaryTf.setBounds(900,200,200,30);
Frame.add(SalaryTf);

//
// Tutor Specialization Label

JLabel Specialization = new JLabel ("Specialization : ");
Specialization.setFont(font2);
Specialization.setBounds(650,240,180,30);
Frame.add(Specialization);

// Tutor Specialization Text Field

JTextField SpecializationTf = new JTextField();
SpecializationTf.setFont(font2);
SpecializationTf.setBounds(900,240,200,30);
Frame.add(SpecializationTf);



//
// Tutor Academic Qualification Label

JLabel AcademicQualification = new JLabel ("Academic Qualification : ");
AcademicQualification.setFont(font2);
AcademicQualification.setBounds(650,280,200,30);
Frame.add(AcademicQualification);

// Tutor Academic Qualification Text Field
```

```
JTextField AcademicQualificationTf = new JTextField();
AcademicQualificationTf.setFont(font2);
AcademicQualificationTf.setBounds(900,280,200,30);
Frame.add(AcademicQualificationTf);



//
// Tutor Performance Index Label

JLabel PerformanceIndex = new JLabel ("Performance Index : ");
PerformanceIndex.setFont(font2);
PerformanceIndex.setBounds(650,320,180,30);
Frame.add(PerformanceIndex);

// Tutor Performance Index TextField

JTextField PerformanceIndexTf = new JTextField();
PerformanceIndexTf.setFont(font2);
PerformanceIndexTf.setBounds(900,320,200,30);
Frame.add(PerformanceIndexTf);

//
// Buttons for Tutor.
// Button for Add Tutor.



JButton AddTutor = new JButton("Add Tutor");
AddTutor.setFont(font2);
AddTutor.setBackground(Color.GRAY);
AddTutor.setBounds(650,380,150,30);
```

```
    Frame.add(AddTutor);




    //
    // Action Button Of Add Tutor

    AddTutor.addActionListener(new ActionListener()
    {
      public void actionPerformed(ActionEvent AddTutor)
      {
        try
        {
          int TeacherID = Integer.parseInt(TeacherId.getText());
          String TeacherName = TeacherN.getText();
          String TeacherAddress = TeacherA.getText();
          String TeacherWorkingType  = TeacherWt.getText();
          String TeacherEmploymentStatus = TeacherEs.getText();
          int WorkingHours = Integer.parseInt(WorkingHoursTfT.getText());
          double Salary = Double.parseDouble(SalaryTf.getText());
          String Specialization = SpecializationTf.getText();
          String AcademicQualification = AcademicQualificationTf.getText();
          int PerformanceIndex = Integer.parseInt(PerformanceIndexTf.getText());



          for(Teacher teacher : TeacherDetails)
          {
            if(teacher.getTeacherId() == TeacherID)
            {
              JOptionPane.showMessageDialog(null, "Teacher ID exists. Please
add correct ID");
```

```
                    return;
                }
            }

            Tutor TutorValue = new Tutor(TeacherID, TeacherName, TeacherAddress,
TeacherWorkingType, TeacherEmploymentStatus, WorkingHours,
            Salary, Specialization, AcademicQualification,PerformanceIndex );

            TeacherDetails.add(TutorValue);

            JOptionPane.showMessageDialog(null, "Successfully! Added Tutor");
        }
        catch (NumberFormatException e)
          {
            JOptionPane.showMessageDialog(null," Please Kindly fill the NUMERIC
VALUE ");
          }
        if (WorkingHoursTfT.getText().isEmpty() || SalaryTf.getText().isEmpty() ||
PerformanceIndexTf.getText().isEmpty()
                || TeacherId.getText().isEmpty() || TeacherN.getText().isEmpty() ||
TeacherA.getText().isEmpty()
                || TeacherWt.getText().isEmpty() || TeacherEs.getText().isEmpty() )
                {
                    JOptionPane.showMessageDialog(null," Please Kindly fill the  VALUE
");
                }

        }
    }
```

```
);



//
//  Lecturer

// Adding label to Lecturer

JLabel Heading3 = new JLabel("LECTURER");
Heading3.setFont(font1);
Heading3. setBounds(1200,80,300,80);
Frame.add(Heading3);

// Attributes of Lecturer

// Department Label

JLabel Department = new JLabel ("Department : ");
Department.setFont(font2);
Department.setBounds(1200,160,120,30);
Frame.add(Department);

// Department TextField

JTextField DepartmentTf = new JTextField();
DepartmentTf.setFont(font2);
DepartmentTf.setBounds(1400,160,200,30);
Frame.add(DepartmentTf);

 // Years of Experience Label
```

```
JLabel YearsOfExperience = new JLabel ("Years of Experience : ");
YearsOfExperience.setFont(font2);
YearsOfExperience.setBounds(1200,200,180,30);
Frame.add(YearsOfExperience);

// Years of Experience TextField

JTextField YearsOfExperienceTf = new JTextField();
YearsOfExperienceTf.setFont(font2);
YearsOfExperienceTf.setBounds(1400,200,200,30);
Frame.add(YearsOfExperienceTf);


 // Working Hours Label

JLabel WorkingHour = new JLabel ("Working Hours : ");
WorkingHour.setFont(font2);
WorkingHour.setBounds(1200,240,150,30);
Frame.add(WorkingHour);

// Working Hours TextField

JTextField WorkingHourTf = new JTextField();
WorkingHourTf.setFont(font2);
WorkingHourTf.setBounds(1400,240,200,30);
Frame.add(WorkingHourTf);
```

```
 //
// Buttons
//Buttons for Add Lecturer

JButton AddLecturer = new JButton("Add Lecturer");
AddLecturer.setFont(font2);
AddLecturer.setBackground(Color.GRAY);
AddLecturer.setBounds(1200,380,150,30);
Frame.add(AddLecturer);




//
// Action OF Button of Add Lecturer

AddLecturer.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent AddLecturer)
    {
      try
       {
         String Department = DepartmentTf.getText();
         int YearsOfExperience = Integer.parseInt(YearsOfExperienceTf.getText());
         int TeacherID = Integer.parseInt(TeacherId.getText());
         String TeacherName = TeacherN.getText();
         String TeacherAddress = TeacherA.getText();
         String TeacherWorkingType  = TeacherWt.getText();
         String TeacherEmploymentStatus = TeacherEs.getText();
         int WorkingHours = Integer.parseInt(WorkingHourTf.getText());
```

```
            for(Teacher teacher : TeacherDetails)
            {
               if(teacher.getTeacherId() == TeacherID)
               {
                  JOptionPane.showMessageDialog(null, "Teacher ID exists. Please
add correct ID");
                  return;
               }
            }

            Lecturer Lect = new
Lecturer(Department,YearsOfExperience,TeacherID,TeacherName,TeacherAddress
,TeacherWorkingType,TeacherEmploymentStatus, WorkingHours);

            TeacherDetails.add(Lect);

            JOptionPane.showMessageDialog(null, "Successfully! Added Lecturer");
          }

         catch (NumberFormatException e)
          {
             JOptionPane.showMessageDialog(null,  "Please Kindly fill the NUMERIC
VALUE");
          }
         if (DepartmentTf.getText().isEmpty() ||
YearsOfExperienceTf.getText().isEmpty() || WorkingHourTf.getText().isEmpty()
               || TeacherId.getText().isEmpty() || TeacherN.getText().isEmpty() ||
TeacherA.getText().isEmpty()
               || TeacherWt.getText().isEmpty() || TeacherEs.getText().isEmpty() )
               {
                  JOptionPane.showMessageDialog(null," Please Kindly fill the  VALUE
");
```

```
            }

    }
  }
  );



  //
  // Adding Salary
  // Adding label Salary

  JLabel Heading4 = new JLabel("SALARY");
  Heading4.setFont(font1);
  Heading4. setBounds(100,480,150,30);
  Frame.add(Heading4);



  // Attributes of Salary

  //
  // Salary Teacher ID Label

  JLabel TeacherIdd = new JLabel ("Teacher ID : ");
  TeacherIdd.setFont(font2);
  TeacherIdd.setBounds(100,540,100,30);
  Frame.add(TeacherIdd);

  // Salary Teacher ID Label

  JTextField TeacherIddTf = new JTextField();
```

```
TeacherIddTf.setFont(font2);
TeacherIddTf.setBounds(350,540,200,30);
Frame.add(TeacherIddTf);

    //
// Salary New Salary Label

JLabel NewSalary = new JLabel ("New Salary : ");
NewSalary.setFont(font2);
NewSalary.setBounds(100,580,150,30);
Frame.add(NewSalary);

// Salary New Salary TextField

JTextField NewSalaryTf = new JTextField();
NewSalaryTf.setFont(font2);
NewSalaryTf.setBounds(350,580,200,30);
Frame.add(NewSalaryTf);



//
// Salary New Performance Index Label

JLabel NewPerformanceIndex = new JLabel ("New Performance Index : ");
NewPerformanceIndex.setFont(font2);
NewPerformanceIndex.setBounds(100,620,200,30);
Frame.add(NewPerformanceIndex);

// Salary New Performance Index TextField
```

```
JTextField NewPerformanceIndexTf = new JTextField();
NewPerformanceIndexTf.setFont(font2);
NewPerformanceIndexTf.setBounds(350,620,200,30);
Frame.add(NewPerformanceIndexTf);


//
// Buttons for Salary.
// Button for Set Salary.


JButton SetSalary = new JButton("Set Salary");
SetSalary.setFont(font2);
SetSalary.setBackground(Color.GRAY);
SetSalary.setBounds(100,720,150,30);
Frame.add(SetSalary);



//
// Action OF Button Set Salary

SetSalary.addActionListener(new ActionListener()
{
   public void actionPerformed(ActionEvent SetSalary)
   {
      try
      {
         int teacherId = Integer.parseInt(TeacherIddTf.getText());
         double newSalary = Double.parseDouble(NewSalaryTf.getText());
```

```
          int newPerformanceIndex =
Integer.parseInt(NewPerformanceIndexTf.getText());


              // Find the tutor object with matching teacher ID
          Tutor tutorToUpdate = null;
            for (Teacher teacher : TeacherDetails)
              {
                if (teacher instanceof Tutor && teacher.getTeacherId() ==
teacherId)
                {
                  tutorToUpdate = (Tutor) teacher;
                  break;
                }
              }


            if (tutorToUpdate == null)
            {
              JOptionPane.showMessageDialog(null, "Invalid Teacher ID! No
matching Tutor found.");
              return;
            }


            // Update salary and performance index
             tutorToUpdate.setSalary(newSalary,newPerformanceIndex);



          JOptionPane.showMessageDialog(null, "Salary and Performance Index
updated successfully!");


        }
        catch (NumberFormatException e)
        {
                JOptionPane.showMessageDialog(null, "Empty Fields!! Please Kindly
FILL all the FIELDS ");

            }

            catch (Exception e)

            {

                JOptionPane.showMessageDialog(null,e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);

            }

        }

    }

    );





    //

    // Adding Grade Assignments

    // Adding label Grade Assignments


    JLabel Heading5 = new JLabel("Grade Assignments ");

    Heading5.setFont(font1);

    Heading5. setBounds(650,480,300,30);

    Frame.add(Heading5);



    // Attributes of Grade Assignments


    //

    // Grade Assignments Teacher ID Label


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

    TeacherIdGa.setFont(font2);
    TeacherIdGa.setBounds(650,540,100,30);
    Frame.add(TeacherIdGa);

    // Grade Assignments Teacher ID Label

    JTextField TeacherIdGaTf = new JTextField();
    TeacherIdGaTf.setFont(font2);
    TeacherIdGaTf.setBounds(900,540,200,30);
    Frame.add(TeacherIdGaTf);


        //
    // Grade Assignments Graded Score Label

    JLabel GradedScore  = new JLabel ("Graded Score  : ");
    GradedScore.setFont(font2);
    GradedScore.setBounds(650,580,150,30);
    Frame.add(GradedScore);

    // Grade Assignments Teacher ID Label

    JTextField GradedScoreTf = new JTextField();
    GradedScoreTf.setFont(font2);
    GradedScoreTf.setBounds(900,580,200,30);
    Frame.add(GradedScoreTf);


        //
    // Grade Assignments Teacher ID Label
```

```
JLabel DepartmentGa = new JLabel ("Department : ");
DepartmentGa.setFont(font2);
DepartmentGa.setBounds(650,620,120,30);
Frame.add(DepartmentGa);

// Grade Assignments Teacher ID TextField

JTextField DepartmentGaTf = new JTextField();
DepartmentGaTf.setFont(font2);
DepartmentGaTf.setBounds(900,620,200,30);
Frame.add(DepartmentGaTf);



    //
// Grade Assignments Years Of Experience Label

JLabel YearsOfExperienceGa = new JLabel ("Years Of Experience : ");
YearsOfExperienceGa.setFont(font2);
YearsOfExperienceGa.setBounds(650,660,200,30);
Frame.add(YearsOfExperienceGa);

// Grade Assignments Years Of Experience TextField

JTextField YearsOfExperienceGaTf = new JTextField();
YearsOfExperienceGaTf.setFont(font2);
YearsOfExperienceGaTf.setBounds(900,660,200,30);
Frame.add(YearsOfExperienceGaTf);



//
```

```
    // Button for Grade Assignments.

    JButton GradeAssignments = new JButton("Assign Grade");
    GradeAssignments.setFont(font2);
    GradeAssignments.setBackground(Color.GRAY);
    GradeAssignments.setBounds(650,720,200,30);
    Frame.add(GradeAssignments);




    //
    // Action OF Button Grade Assignments

    GradeAssignments.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent GradeAssignments)
        {
            try
            {
                int teacherId = Integer.parseInt(TeacherIdGaTf.getText());
                int gradedScore = Integer.parseInt(GradedScoreTf.getText());
                String department = DepartmentGaTf.getText();
                int yearsOfExperience =
Integer.parseInt(YearsOfExperienceGaTf.getText());

                // Find the lecturer object with matching teacher ID
                Lecturer lecturerToUpdate = null;
                for (Teacher teacher : TeacherDetails)
                {
                    if (teacher instanceof Lecturer && teacher.getTeacherId() ==
teacherId)
                    {
                        lecturerToUpdate = (Lecturer) teacher;
                        lecturerToUpdate .GradeAssignment(gradedScore, department,
yearsOfExperience);
                        break;
                    }
                    }

                if (lecturerToUpdate == null)
                {
                    JOptionPane.showMessageDialog(null, "Invalid Teacher ID! No
matching Lecturer found.");
                    return;
                }

                // Check if the department and years of experience match
                if (!lecturerToUpdate.getDepartment().equals(department))
                {
                    JOptionPane.showMessageDialog(null, "Department or Years of
Experience do not match the Lecturer's details.");
                    return;
                }




                JOptionPane.showMessageDialog(null, "Graded Score updated
successfully!");
                }
            catch (NumberFormatException e)
            {
                JOptionPane.showMessageDialog(null,"EMPTY FIELDS!! Please Kindly
fill all the fields");

            }

            catch (Exception e)

            {

                JOptionPane.showMessageDialog(null, e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);

            }

        }

    });



    //

    // Adding Remove Tutor


   // Adding Label Remove Tutor



   JLabel Heading6 = new JLabel("REMOVE TUTOR");

   Heading6.setFont(font1);

   Heading6. setBounds(100,800,300,80);

   Frame.add(Heading6);




   // Remove Tutor Teacher ID Label


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

   TeacherIdRt.setFont(font2);

   TeacherIdRt.setBounds(100,880,180,30);

   Frame.add(TeacherIdRt);
```

```
// Remove Tutor Teacher ID TextField


JTextField TeacherIdRtTf = new JTextField();
TeacherIdRtTf.setFont(font2);
TeacherIdRtTf.setBounds(350,880,200,30);
Frame.add(TeacherIdRtTf);



 //
// Buttons for Remove Tutor.
// Button for Remove Tutor.



JButton RemoveTutor = new JButton("Remove Tutor");
RemoveTutor.setFont(font2);
RemoveTutor.setBackground(Color.GRAY);
RemoveTutor.setBounds(100,940,180,30);
Frame.add(RemoveTutor);






//
// Action OF Button RemoveTutor


RemoveTutor.addActionListener(new ActionListener()
 {
    public void actionPerformed(ActionEvent RemoveTutor)
```

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

            // Find the tutor object with matching teacher ID
                Tutor tutorToRemove = null;
            for (Teacher teacher : TeacherDetails)
            {
            if (teacher instanceof Tutor && teacher.getTeacherId() == teacherId)
            {
                tutorToRemove = (Tutor) teacher;
                break;
            }
            }

            if (tutorToRemove == null)
            {
            JOptionPane.showMessageDialog(null, "Invalid Teacher ID! No matching
Tutor found.");
            return;
            }

            // Remove the tutor from the TeacherDetails list
            TeacherDetails.remove(tutorToRemove);



            TeacherId.setText("");
            TeacherN.setText("");
            TeacherA.setText("");
            TeacherWt.setText("");
            TeacherEs.setText("");
            WorkingHoursTfT.setText("");
            SalaryTf.setText("");
            SpecializationTf.setText("");
            AcademicQualificationTf.setText("");
            PerformanceIndexTf.setText("");
            TeacherIddTf.setText("");
            NewSalaryTf.setText("");
            NewPerformanceIndexTf.setText("");
            TeacherIdRtTf.setText("");
            JOptionPane.showMessageDialog(null, "Tutor removed successfully!");
            }
            catch (NumberFormatException e)
            {
               JOptionPane.showMessageDialog(null, "EMPTY FIELDS!! Please
Kindly fill all the fields");
            }
            catch (Exception e)
            {
               JOptionPane.showMessageDialog(null, e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
            }
         }


         });



      //
     // Action OF Button Clear
      Clear.addActionListener(new ActionListener()
     {
        public void actionPerformed(ActionEvent clear)
        {
         TeacherId.setText("");
         TeacherN.setText("");
         TeacherA.setText("");
         TeacherWt.setText("");
         TeacherEs.setText("");
         WorkingHoursTfT.setText("");
         SalaryTf.setText("");
         SpecializationTf.setText("");
         AcademicQualificationTf.setText("");
         PerformanceIndexTf.setText("");
         DepartmentTf.setText("");
         YearsOfExperienceTf.setText("");
         WorkingHourTf.setText("");
         TeacherIddTf.setText("");
         NewSalaryTf.setText("");
         NewPerformanceIndexTf.setText("");
         TeacherIdGaTf.setText("");
         GradedScoreTf.setText("");
         DepartmentGaTf.setText("");
         YearsOfExperienceGaTf.setText("");
         TeacherIdRtTf.setText("");
         JOptionPane.showMessageDialog(null, "All Fields are CLEARED");
        }
     }
     );
 }
 public static void main(String[] args)
 {
     // Create an instance of TeacherGUI
     TeacherGUI teachersGui = new TeacherGUI();

     // Calling the gui() method
     teachersGui.Gui();
 }
}
```