CS4001NI Programming

30% Individual Coursework

2023-24 Autumn

Student Name: Manish Shrestha

London Met ID: 23050365

College ID: np01cp4a230415

Group: c10

Assignment Due Date: Friday, May 10, 2024

Assignment Submission Date: Friday, May 10, 2024

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

Table of Contents

Contents

Table of Figures

Table of Tables

1. Introduction

1.1 Java

Java, both a programming language and a software platform, stands as a cornerstone technology renowned for its extensive utilization. Rooted in object-oriented principles, it draws its syntax and regulations from C and C++, making it a favoured choice in software development. At the heart of digital enterprises, web applications find their stronghold, with Java serving as a robust foundation for their creation. Its hallmark portability facilitates seamless migration of Java application code from laptops to mobile devices, adding to its allure. When deliberating on a programming language and environment for corporate applications, Java emerges as a compelling choice due to its interoperability, scalability, and adaptability. Despite facing competition from an array of languages like Python, Ruby, PHP, Swift, and C++, Java's enduring presence of over two decades speaks volumes about its enduring relevance and widespread adoption in application software development (IBM, 2024).

Figure 1:Java Logo

A Graphical User Interface (GUI) serves as a visual means for users to interact with machines, facilitating straightforward communication. It's a prevalent interface featuring graphical elements like buttons and icons, enabling users to engage with the system through visual cues rather than traditional text or command inputs. Basic actions such as

clicking enable users to convey their intentions to the GUI, which promptly translates them into machine-understandable instructions, often in assembly code.

1.2 Aims and Objective of the Coursework

- This coursework effectively demonstrates comprehension of Graphical User Interface (GUI) and its practical application.
- Its purpose is to aid students in acquiring diverse skills related to the topic.
- It aims to educate about the functioning of the banking system.

2. Tools Used

2.1 BlueJ

BlueJ is a Java integrated development environment aimed for college and university students. It was created to teach object orientation in a Java development environment by the University of Kent and Deakin University. First-year students can learn the Java programming language with the aid of BlueJ's user-friendly teaching environment before moving on to a popular IDE (NetBeans). The highly dynamic BlueJ integrated development environment promotes experimentation and discovery (Knowledge Boat, 2023).

Figure 2: BlueJ Logo

2.2 Ms Word

Microsoft Word, commonly known as Winword, MS Word, or simply Word, is a word processing software developed by Microsoft and included in the suite of Microsoft Office

productivity tools. Originally developed by Charles Simonyi and Richard Brodie, it was released in 1983. Microsoft Word enables users to create professional-quality documents, reports, letters, and resumes. It offers various features like spell check, grammar check, text and font formatting, HTML support, image integration, advanced page layout options, and more, setting it apart from basic text editors (Hope, Computer Hope, 2021).

Figure 3: Ms-Word Logo

2.3 Draw.io

Draw.io is proprietary software for creating diagrams and charts. You can use the software's automatic layout option or create your own layout. They provide a vast selection of shapes and hundreds of graphic features to make your diagram or chart unique. The drag-and-drop capability makes it simple to create a visually appealing diagram or chart. Depending on your needs, Draw.io can save stored charts in the cloud, on a server, or in network storage at a data centre (Hope, Computer Hope, 2023).

Figure 4: Draw.io Logo

2..4 Moqups

Moqups is a web-based design tool which can help you generate wireframes, mockups and prototypes all inside one environment. You can navigate between projects and take your design from low fidelity (initial stages and rough sketches) to high fidelity (more complete and detailed). Mockups, wireframes, and prototyping are essential components of digital design. Before bringing anything public, you may simulate your ideas, test multiple user journeys and flows, and try out various user experiences by visualizing your thoughts. The beauty of Moqups is that you can switch back and forth as often as you like between your wireframes and mockups; you are not required to develop one before the other (Themes, 2022).

Figure 5: Moqups Logo

3. Class Diagram

A class diagram is a static structure diagram used in object-oriented programming to illustrate class relationships. It is also an effective approach to display a system's class hierarchy. It is a visual representation of class objects in a model system, arranged according to class types (Venngage, 2023).

3.1 Class Diagram of Teacher

Teacher
-teacher_id: int -teacher_name: String -address: String -working_type: String -employment_status: String -working_hour: int
+ <<constructor>>Teacher(teacher_id:int, teacher_name:String, address: String, working_type:String, employment_status: String) + getTeacher_id(): int + getTeacher_name(): String + getAddress(): String + getWorking_type(): String + getEmployment_status(): String + getWorking_hour(): int +setWorking_hour(working_hour: int): void +display(): void

Figure 6: Teacher Class Diagram

3.2Class Diagram of Lecturer

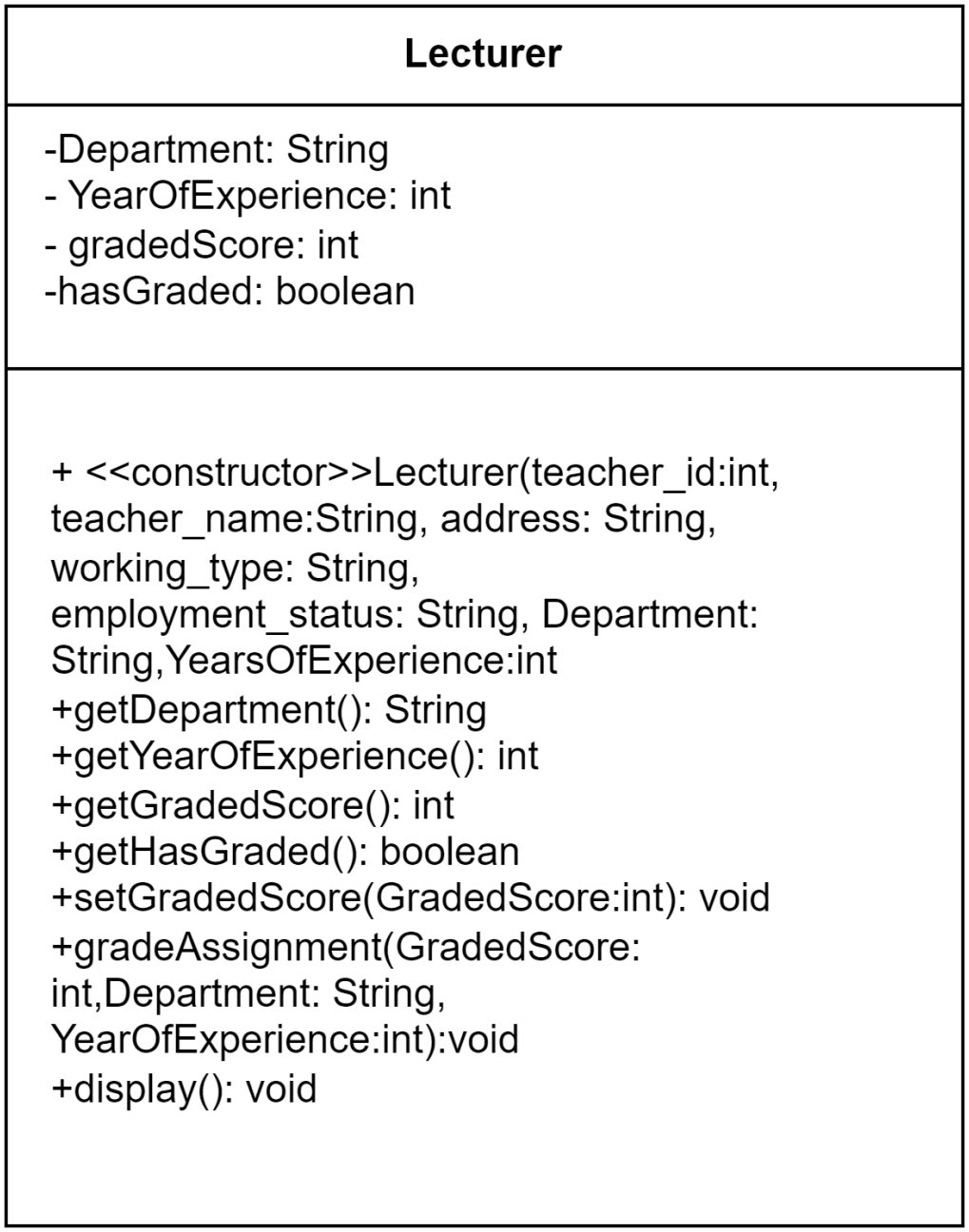

Figure 7: Lecturer Class Diagram

3.3 Class Diagram of Tutor

Tutor
-salary: double -specialization: String -academic_qualifications: String -performance_index: int -isCertifed: boolean
+<<constructor>>Tutor(teacher_id:int, teacher_name:String, address: String, working_type: String, employment_status: String, working_hour: int, salary: double, specialization:String, academic_qualifications: String, performance_index:int) +getSalary(): double +getSpecialization(): String +getAcademic_qualification(): String +getPerformance_index(): int +getisCertified():boolean +setSalary(salary: double): void +setPerformance_index(performance_index: int): void +set(salary: double, performance_index: int,working_hour):void +removeTutor(): void +display(): void

Figure 8: Tutor Class Diagram

3.4 Combined Class Diagram

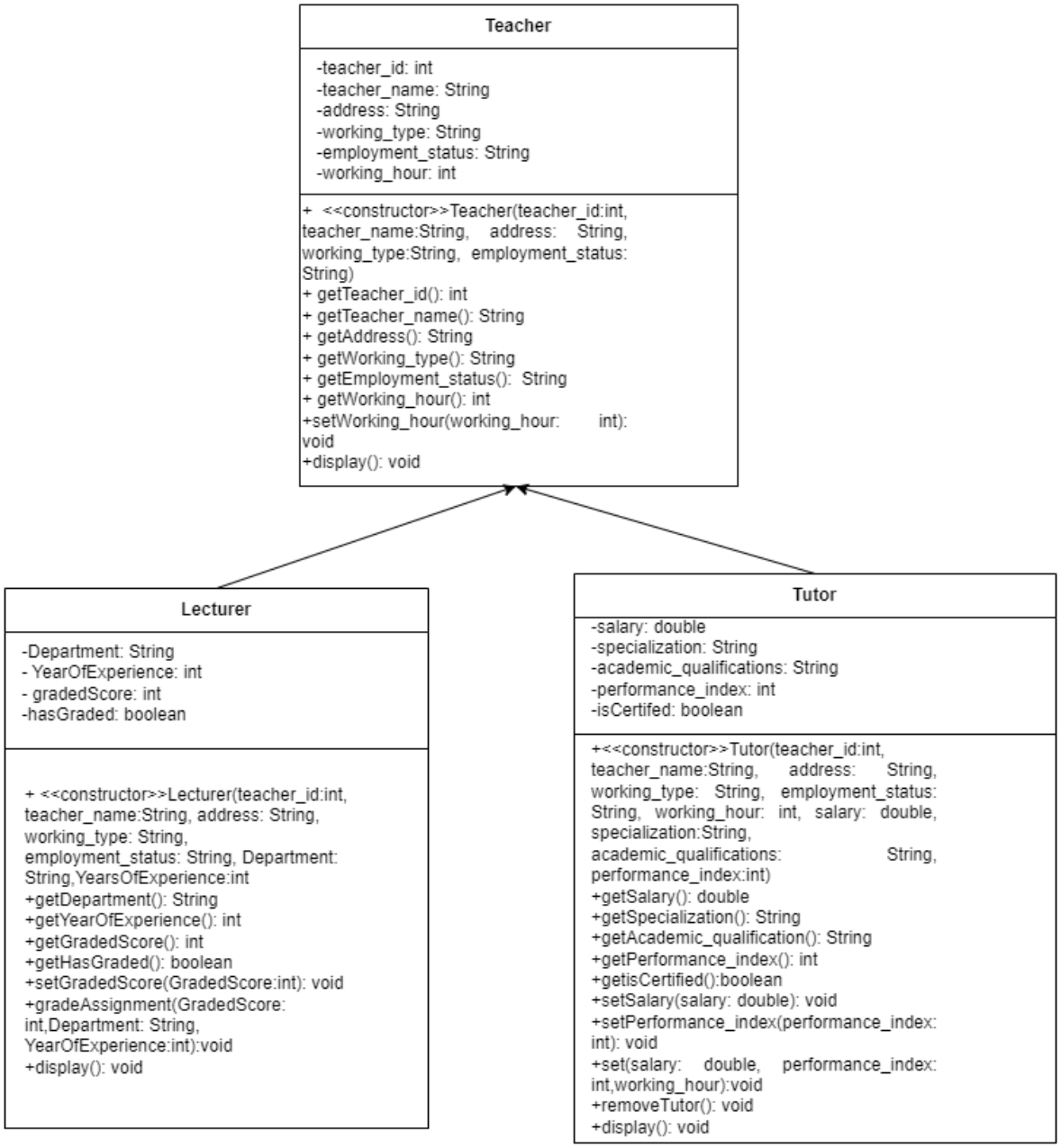

Figure 9: Combined Class Diagram

3.5 Class Diagram of TeacherGUI

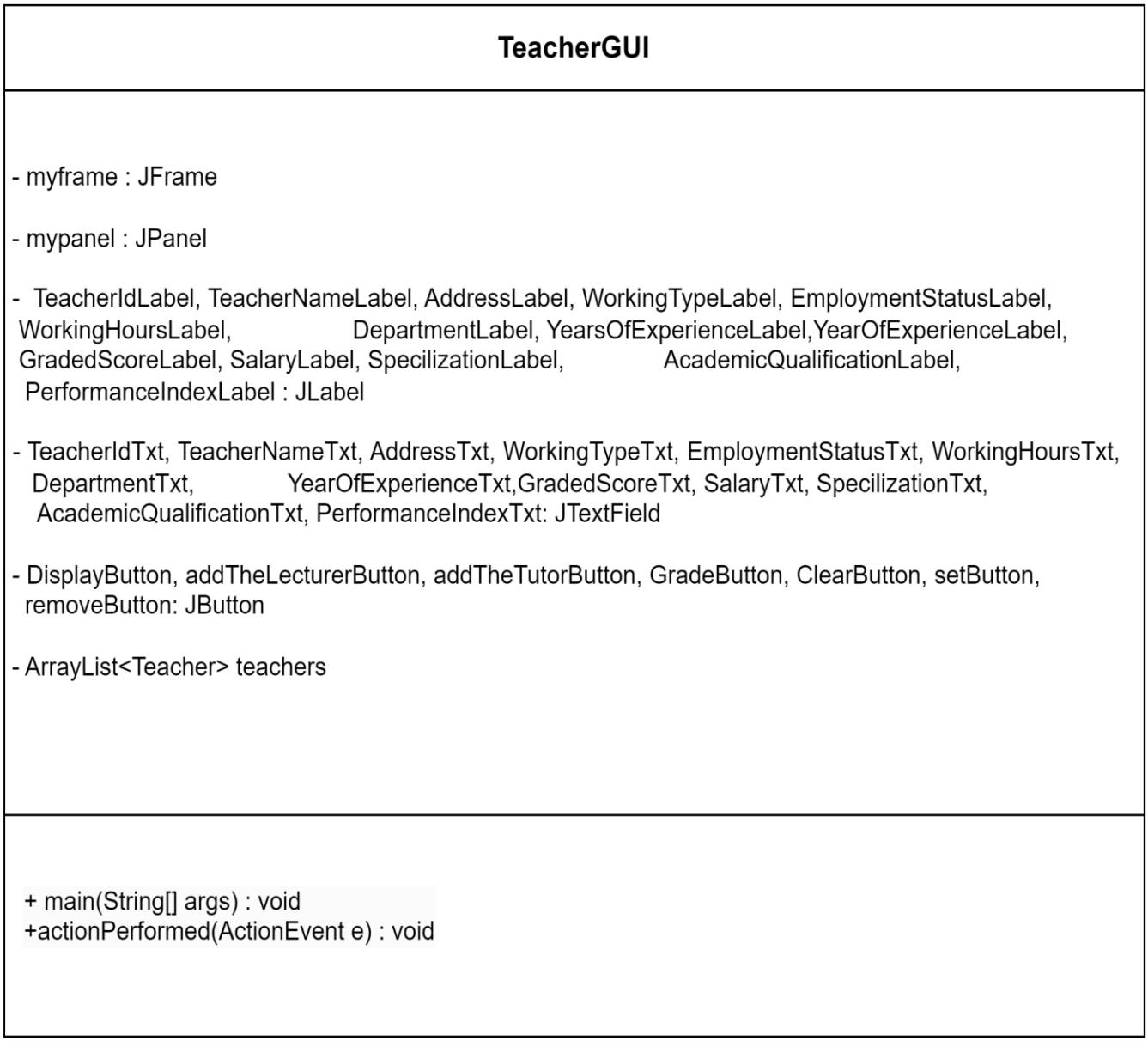

Figure 10: TeacherGUI Class Diagram

4. PseudoCode

4.1 Pseudocode for TeacherGUI

CREATE a class TeacherGUI

DO

DECLARE myframe as JFrame

DECLARE mypanel as JPanel

DECLARE TMLabel as JLabel

DECLARE TeacherIdLabel, TeacherNameLabel, AddressLabel, WorkingTypeLabel, EmploymentStatusLabel, WorkingHoursLabel, DepartmentLabel, YearsOfExperienceLabel, YearOfExperienceLabel, GradedScoreLabel, SalaryLabel, SpecilizationLabel, AcademicQualificationLabel, PerformanceIndexLabel as JLabel

DECLARE TeacherIdTxt, TeacherNameTxt, AddressTxt, WorkingTypeTxt, EmploymentStatusTxt, WorkingHoursTxt, DepartmentTxt, YearOfExperienceTxt, GradedScoreTxt, SalaryTxt, SpecilizationTxt, AcademicQualificationTxt, PerformanceIndexTxt as JTextField

DECLARE DisplayButton, addTheLecturerButton, addTheTutorButton, GradeButton, ClearButton, setButton, removeButton as JButton

DECLARE isLecturerAdded, isTutorAdded as boolean

DECLARE teachers as ArrayList of Teacher objects

CREATE a constructor TeacherGUI

DO

CREATE myframe as JFrame

SET myframe title

SET myframe size

SET myframe default close operation on exit

CREATE mypanel as JPanel with GridBagLayout

DECLARE constraints as GridBagConstraints

SET constraints fill to GridBagConstraints.VERTICAL

SET constraints insets

CREATE TMLabel as JLabel

SET TMLabel font

CREATE TeacherIdLabel, TeacherNameLabel, AddressLabel, WorkingTypeLabel, EmploymentStatusLabel, WorkingHoursLabel, DepartmentLabel, YearsOfExperienceLabel, YearOfExperienceLabel, GradedScoreLabel, SalaryLabel, SpecilizationLabel, AcademicQualificationLabel, PerformanceIndexLabel as JLabel

SET labels font

CREATE TeacherIdTxt, TeacherNameTxt, AddressTxt, WorkingTypeTxt, EmploymentStatusTxt, WorkingHoursTxt, DepartmentTxt, YearOfExperienceTxt, GradedScoreTxt, SalaryTxt, SpecilizationTxt, AcademicQualificationTxt, PerformanceIndexTxt as JTextField

CREATE DisplayButton, addTheLecturerButton, addTheTutorButton, GradeButton, ClearButton, setButton, removeButton as JButton

INITIALIZE teachers as new ArrayList of Teacher objects

CREATE a JButton addTheLecturerButton with label "Add The Lecturer"

CREATE a method actionPerformed with parameter e of type ActionEvent for addTheLecturerButton

DO

IF any of the required fields are empty

THEN

DISPLAY an error message prompting to fill in all required fields

ELSE

TRY

```
        PARSE the input values
        CREATE a new Lecturer object
        ADD the new Lecturer to the ArrayList
        SET isLecturerAdded to true
        DISPLAY a success message
    CATCH NumberFormatException
        DISPLAY an error message prompting to input numerical values for Teacher ID and Years of Experience
  END IF
END DO
CREATE a JButton addTheTutorButton with label "Add The Tutor"
CREATE a method actionPerformed with parameter e of type ActionEvent for addTheTutorButton
DO
  IF any of the required fields are empty
  THEN
    DISPLAY an error message prompting to fill in all required fields
  ELSE
    TRY
        PARSE the input values
        CREATE a new Tutor object
        ADD the new Tutor to the ArrayList
        SET isTutorAdded to true
```

```
            DISPLAY a success message
        CATCH NumberFormatException
            DISPLAY an error message prompting to input numerical values for Teacher ID, Working Hours, Salary, and Performance Index
    END IF
END DO
CREATE a JButton GradeButton with label "Grade the Assignment"
CREATE a method actionPerformed with parameter e of type ActionEvent for GradeButton
DO
    GET input values from text fields
    IF any of the required fields are empty
    THEN
        DISPLAY an error message prompting to fill in all required fields
        RETURN
    END IF
    TRY
        PARSE input values
        FIND the teacher with the given ID in the ArrayList
        IF the teacher with the given ID exists
        THEN
            IF the selected teacher is an instance of Lecturer
            THEN
```

```
            CAST the selectedTeacher to Lecturer
            GRADE the assignment using the Lecturer method
            DISPLAY an information dialog with entered data
        ELSE
            DISPLAY an error message that the selected teacher is not a Lecturer
        END IF
    ELSE
        DISPLAY an error message that no teacher found with the entered ID
    END IF
  CATCH NumberFormatException
    DISPLAY an error message prompting to input numerical values for Teacher ID,
Graded Score, and Years of Experience
END DO
CREATE a JButton DisplayButton with label "Display"
CREATE a method actionPerformed with parameter e of type ActionEvent for
DisplayButton
DO
  GET the teacher ID input
  IF any of the required fields are empty
  THEN
    DISPLAY an error message prompting to fill in all required fields
  ELSE
    CONSTRUCT the display message
```

```
        IF "Add The Lecturer" button was clicked
        THEN
            ADD an additional field for lecturers
        END IF
        IF "Add The Tutor" button was clicked
        THEN
            ADD additional fields for tutors
        END IF
        SHOW the message dialog
        DISPLAY a success message for displaying
    END IF
END DO
CREATE a JButton ClearButton with label "Clear"
CREATE a method actionPerformed with parameter e of type ActionEvent for
ClearButton
DO
    CLEAR all the text fields
    DISPLAY a success message for clearing
END DO
CREATE a JButton setButton with label "Set"
CREATE a method actionPerformed with parameter e of type ActionEvent for setButton
DO
    GET the input value of teacher ID
```

```
IF teacher ID is not empty
THEN
    PARSE the teacher ID to integer
    SEARCH for the teacher with the given ID in the teachers ArrayList
    FOR each teacher in teachers
        IF the teacher ID matches
        THEN
            IF the teacher is an instance of Tutor
            THEN
                CAST the teacher object as Tutor
                DISPLAY the current salary and performance index
                GET the new salary and performance index
                IF new salary and performance index are not empty
                THEN
                    SET the new salary and performance index
                    DISPLAY the updated details
                ELSE
                    DISPLAY an error message to enter both new salary and new
performance index
                END IF
            ELSE
                DISPLAY an error message that the teacher with this ID is not a Tutor
            END IF
```

```
            EXIT loop since teacher ID match found
        END IF
    END FOR
    IF no teacher with the entered ID is found
    THEN
        DISPLAY an error message that teacher with this ID does not exist
    END IF
  ELSE
    DISPLAY an error message to enter a valid teacher ID
  END IF
  DISPLAY a success message for setting
END DO
CREATE a JButton removeButton with label "Remove"
CREATE a method actionPerformed with parameter e of type ActionEvent for
removeButton
DO
  GET the input value of teacher ID
  IF teacher ID is not empty
  THEN
    PARSE the teacher ID to integer
    SEARCH for the teacher with the given ID in the teachers ArrayList
    FOR each teacher in teachers
        IF the teacher ID matches
```

```
            THEN
                IF the teacher is an instance of Tutor
                THEN
                    CAST the teacher object as Tutor
                    REMOVE the tutor
                    DISPLAY a success message for tutor removal
                    EXIT loop since teacher ID match found
                ELSE
                    DISPLAY an error message that teacher with this ID is not a Tutor
                END IF
            END IF
        END FOR
        IF no teacher with the entered ID is found
        THEN
            DISPLAY an error message that teacher with this ID does not exist
        END IF
    ELSE
        DISPLAY an error message to enter a valid teacher ID
    END IF
END DO
ADD TeacherIdTxt, TeacherIdLabel, TeacherNameTxt, TeacherNameLabel,
AddressTxt, AddressLabel,
```

WorkingTypeTxt, WorkingTypeLabel, EmploymentStatusTxt, EmploymentStatusLabel, WorkingHoursTxt,

WorkingHoursLabel, DepartmentTxt, DepartmentLabel, GradedScoreTxt, GradedScoreLabel,

YearOfExperienceTxt, YearOfExperienceLabel, SalaryTxt, SalaryLabel, SpecilizationTxt,

SpecilizationLabel, AcademicQualificationTxt, AcademicQualificationLabel, PerformanceIndexTxt,

PerformanceIndexLabel, DisplayButton, addTheLecturerButton, addTheTutorButton, GradeButton,

setButton, ClearButton, and removeButton to mypanel.

SET the background color of mypanel to the RGB color (173, 216, 230).

ADD mypanel to myframe

SET myframe visible true

END DO

CREATE main method

DO

INSTANTIATE a new TeacherGUI object

END DO

END DO

4.2 Pseudocode for class Teacher

```
CREATE a class Teacher

DO

 DECLARE private integer teacher_id

 DECLARE private string teacher_name

 DECLARE private string address

 DECLARE private string working_type

 DECLARE private string employment_status

 DECLARE private integer working_hours

 CREATE a constructor Teacher (teacher_id: int, teacher_name: string, address: string,

working_type: string, employment_status: string)

 DO

 ASSIGN this.teacher_id = teacher_id

 ASSIGN this.teacher_name = teacher_name

 ASSIGN this.address = address

 ASSIGN this.working_type = working_type

 ASSIGN this.employment_status = employment_status

 END DO


 CREATE an accessor method getTeacher_id() with return type int

 DO

 RETURN teacher_id

 END DO
```

CREATE an accessor method getTeacher_name() with return type string

DO

RETURN teacher_name

END DO

CREATE an accessor method getAddress() with return type string

DO

RETURN address

END DO

CREATE an accessor method getWorking_type() with return type string

DO

RETURN working_type

END DO

CREATE an accessor method getEmployment_status() with return type string

DO

RETURN employment_status

END DO

CREATE an accessor method getWorking_hour() with return type int

DO

RETURN working_hour

END DO

CREATE a mutator method setWorking_hour(newWorking_hour: int) with no return type

DO

```
ASSIGN this.working_hour = newWorking_hour

END DO

CREATE a method displayDetails() with no return type

DO

OUTPUT "Teacher ID: " + teacher_id

OUTPUT "Teacher Name: " + teacher_name

OUTPUT "Address: " + address

OUTPUT "Working Type: " + working_type

OUTPUT "Employment Status: " + employment_status

IF working_hour == 0 THEN

OUTPUT "Working Hours: Not assigned"

ELSE

OUTPUT "Working Hours: " + working_hour

END IF

END DO

END DO
```

4.3 Pseudocode for Tutor(Sub-class)

```
CREATE a child class Tutor EXTENDING Teacher

DO

DECLARE private variable salary as double

DECLARE private variable specialization as string

DECLARE private variable academic_qualifications as string
```

```
DECLARE private variable performance_index as int
DECLARE private variable isCertified as boolean
CREATE constructor Tutor WITH PARAMETERS
teacher_id, teacher_name, address, working_type, employment_status,
working_hour,
salary, specialization, academic_qualifications, performance_index
DO
CALL super(teacher_id, teacher_name, address, working_type,
employment_status)
CALL setWorking_hour(working_hour)
SET this.salary = salary
SET this.specialization = specialization
SET this.academic_qualifications = academic_qualifications
SET this.performance_index = performance_index
SET this.isCertified = false
END CONSTRUCTOR
CREATE method getSalary WITH NO PARAMETERS
DO
RETURN salary
END METHOD
CREATE method getSpecialization WITH NO PARAMETERS
DO
RETURN specialization
```

```
END METHOD
CREATE method getAcademic _Qualifications WITH NO PARAMETERS
DO
RETURN academic_qualifications
END METHOD
CREATE method getPerformance_index WITH NO PARAMETERS
DO
RETURN performance_index
END METHOD
CREATE method isCertified WITH NO PARAMETERS
DO
RETURN isCertified
END METHOD
CREATE method setSalaryAndCertification WITH PARAMETERS newSalary,
newPerformance_index
DO
IF newPerformance_index > 5 AND getWorking_hour() > 20
DECLARE appraisalPercentage as double
IF newPerformance_index >= 5 AND newPerformance_index <= 7
SET appraisalPercentage = 0.05
ELSE IF newPerformance_index >= 8 AND newPerformance_index <= 9
SET appraisalPercentage = 0.1
ELSE
```

```
SET appraisalPercentage = 0.2

END IF

SET salary = newSalary + (appraisalPercentage * newSalary)

SET isCertified = true

ELSE

DISPLAY "Salary cannot be approved. Tutor is not certified yet."

END IF

END METHOD

CREATE method removeTutor WITH NO PARAMETERS

DO

IF NOT isCertified

SET salary = 0

SET specialization = ""

SET academic_qualifications = ""

SET performance_index = 0

SET isCertified = false

END IF

END METHOD

CREATE method displayDetails WITH NO PARAMETERS

DO

CALL super.displayDetails()

IF isCertified

DISPLAY "Salary: " + salary
```

```
 DISPLAY "Specialization: " + specialization

 DISPLAY "Academic Qualifications: " + academic_qualifications

 DISPLAY "Performance Index: " + performance_index

 END IF

 END METHOD

 CREATE Main class

 DO

 CREATE method main WITH PARAMETERS args

 DO

 CREATE instance tutor of Tutor WITH PARAMETERS

CALL tutor.setSalaryAndCertification(35000, 9)

 CALL tutor.displayDetails()

 CALL tutor.removeTutor()

 CALL tutor.displayDetails()

 END METHOD

 END Main class

END CLASS
```

4.4 Pseudocode for Lecturer(Sub-class)

```
CREATE a child class Lecturer that extends the parent class Teacher

DO

 DECLARE private instance variables department as String

 DECLARE private instance variables yearsOfExperience as int
```

DECLARE private instance variables gradedScore as int

DECLARE private instance variables hasGraded as boolean

CREATE a constructor Lecturer with parameters:

teacher_id as int

teacher_name as String

address as String

working_type as String

employment_status as String

department as String

yearsOfExperience as int

DO

CALL the constructor of the parent class Teacher using super keyword

SET the value of department to the provided department

SET the value of yearsOfExperience to the provided yearsOfExperience

SET gradedScore to 0

SET hasGraded to false

END DO

END CREATE

CREATE an accessor method getDepartment() with return type String

DO

RETURN department

END DO

CREATE an accessor method getYearsOfExperience() with return type int

DO

RETURN yearsOfExperience

END DO

CREATE an accessor method getGradedScore() with return type int

DO

RETURN gradedScore

END DO

CREATE an accessor method hasGraded() with return type boolean

DO

RETURN hasGraded

END DO

CREATE a method setGradedScore() with parameter newGradedScore as int

DO

SET gradedScore to the provided newGradedScore

END DO

CREATE a method gradeAssignment() with parameters:

score as int

studentDepartment as String

studentYearsOfExperience as int

DO

IF NOT hasGraded AND yearsOfExperience >= 5 AND department equals

studentDepartment THEN

```
IF score >= 70 THEN

SET gradedScore to score

ELSE IF score >= 60 THEN

SET gradedScore to 60

ELSE IF score >= 50 THEN

SET gradedScore to 50

ELSE IF score >= 40 THEN

SET gradedScore to 40

ELSE

SET gradedScore to 0

END IF

SET hasGraded to true

ELSE

PRINT "Assignment is not been graded yet. Try again after some time"

END IF

END DO

END CREATE

CREATE a method displayDetails()

DO

CALL the displayDetails method of the parent class Teacher using super keyword

PRINT "Working Department : " + department

PRINT "Years of Experience is : " + yearsOfExperience

IF hasGraded THEN
```

```
PRINT "Total Graded Score is : " + gradedScore

ELSE

PRINT "The total Graded Score is : Not graded"

END IF

END DO

CREATE an inner class named Main

DO

CREATE a main method

DO

CREATE an instance of Lecturer with specific attributes

CALL setGradedScore method with a provided value

CALL displayDetails method of the lecturer instance

END DO

END DO

END DO
```

5. Method Description

5.1 . Public TeacherGUI():

This method constructs the GUI for the Teacher Management System. It initializes the graphical user interface by creating necessary components, such as panels and buttons, within the frame. Specifically, it utilizes a tabbed pane structure to organize different functionalities related to teacher management. Components like JLabels and JTextFields are added to the panels to display and input information. The buttons are positioned within the interface to trigger specific actions associated with teacher management tasks.

5.2. Public Static void main(String[] args):

This method serves as the entry point for the program. It lacks a return type and is solely responsible for instantiating the TeacherGUI object, thereby allowing the program to be executed.

5.3. ActionListener Interface and actionPerformed() method:

The ActionListener Interface, found in the java.awt.event package, is employed in this method. When a user interacts with a button, the ActionListener is notified, triggering the actionPerformed() method. This method is automatically invoked upon button clicks, allowing for the execution of predefined actions associated with the user's interaction. Implementing ActionListener involves implementing the interface, registering the relevant components with the listener, and overriding the actionPerformed() method to define the desired event-handling logic.

Following are some of the actionPerformed() method used in this java program:

i. Add The Lecturer Button (addTheLecturerButton):

This method is invoked when the "Add The Lecturer" button is clicked. It validates whether all required fields are filled. If so, it parses the input values and creates a new Lecturer object with the entered details. The new Lecturer is then added to the ArrayList of teachers, and a success message is displayed. If any numerical input fields contain non-numeric values, an error message is shown.

ii. Add The Tutor Button (addTheTutorButton):

When the "Add The Tutor" button is clicked, this method is triggered. It validates whether all required fields are filled with information. If so, it parses the input values and creates a new Tutor object with the entered details. The new Tutor is added to the ArrayList of teachers, and a success message is displayed. If any numerical input fields contain non-numeric values, an error message is shown.

iii. Grade The Assignment Button (GradeButton):

This method is called when the "Grade the Assignment" button is clicked. It retrieves the input values from text fields, including teacher ID, graded score, department, and years of experience. It then validates whether all required fields are filled. If so, it parses the input values, searches for the teacher with the given ID in the ArrayList, and grades the assignment for the corresponding teacher if found. It displays an information dialog with the entered data upon successful grading.

iv. Display Button (DisplayButton):

Upon clicking the "Display" button, this method is invoked. It retrieves input values from text fields and constructs a display message containing the teacher's information. Additional information is included based on whether the "Add The Lecturer" or "Add The Tutor" button was clicked previously. The constructed message is displayed in a dialog box, along with a success message.

v. Clear Button (ClearButton):

When the "Clear" button is clicked, this method is triggered. It clears all text fields in the GUI interface, resetting them to their default state. After clearing, a success message is displayed.

vi. Set Button (setButton):

This method is invoked upon clicking the "Set" button. It retrieves the teacher ID input and searches for the corresponding teacher in the ArrayList. If found and identified as a Tutor, it displays the current salary and performance index, prompts the user for new values, and updates the tutor's salary and performance index accordingly. Success or error messages are shown based on user actions and input validity.

vii. Remove Button (removeButton):

Upon clicking the "Remove" button, this method is called. It retrieves the teacher ID input and searches for the corresponding teacher in the ArrayList. If found and identified as a

Tutor, it removes the tutor from the system. Success or error messages are displayed accordingly.

6. Testing

6.1 Test 1: To Test that the program can be compiled and run using the command prompt.

Objective	To compile and run the program through command prompt
Action	Navigate to the folder where Java source code file is located using the cd command
Expected Result	The program would be compiled and then executed.
Actual Result	The program was compiled and executed successfully.
Conclusion	The test was successful.

Table 1: Compiling and running program via command prompt.

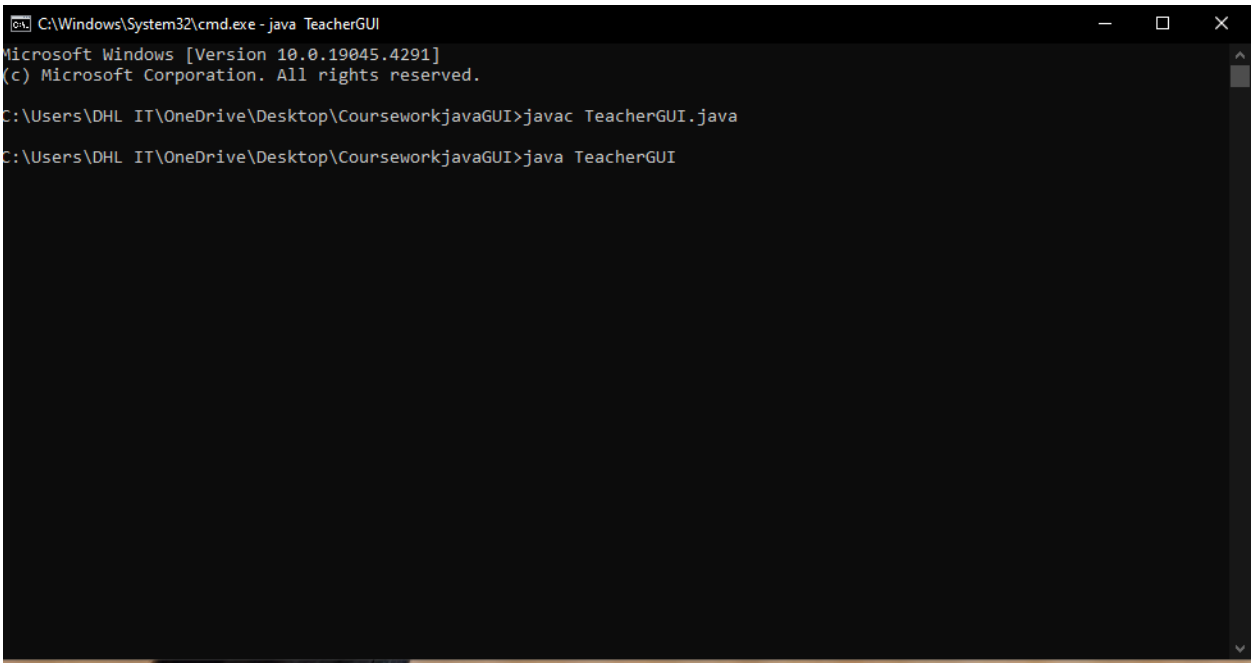

Figure 11: Compiling and running program through command prompt

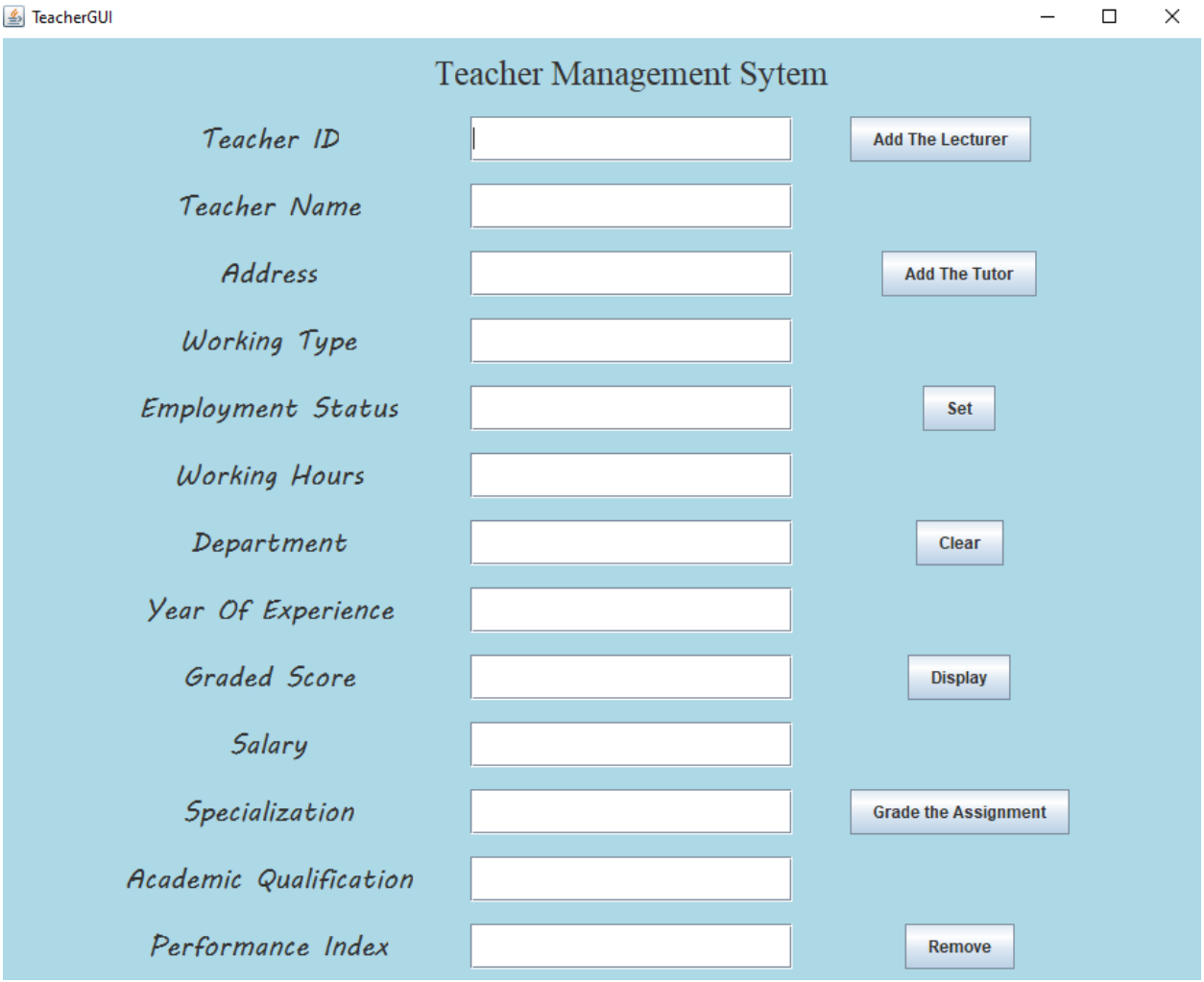

Figure 12: After compiling

6.2 Test 2: Adding the Lecturer, Adding the Tutor, Grading Assignments from Lecturer, Setting the salary, Removing the tutor.

a. Add the Lecturer

Objective	To add the Lecturer
Action	All the empty text fields were filled with below information: Teacher ID=12 Teacher Name = Manish Address = Syangja Working Type = supportive Employment Status = full-time Graded Score =25 Years of Experience =3.
Expected Result	The new lecturer will be added.
Actual Result	The new lecturer was added.
Conclusion	The test was successful.

Table 2: Adding the Lecturer.

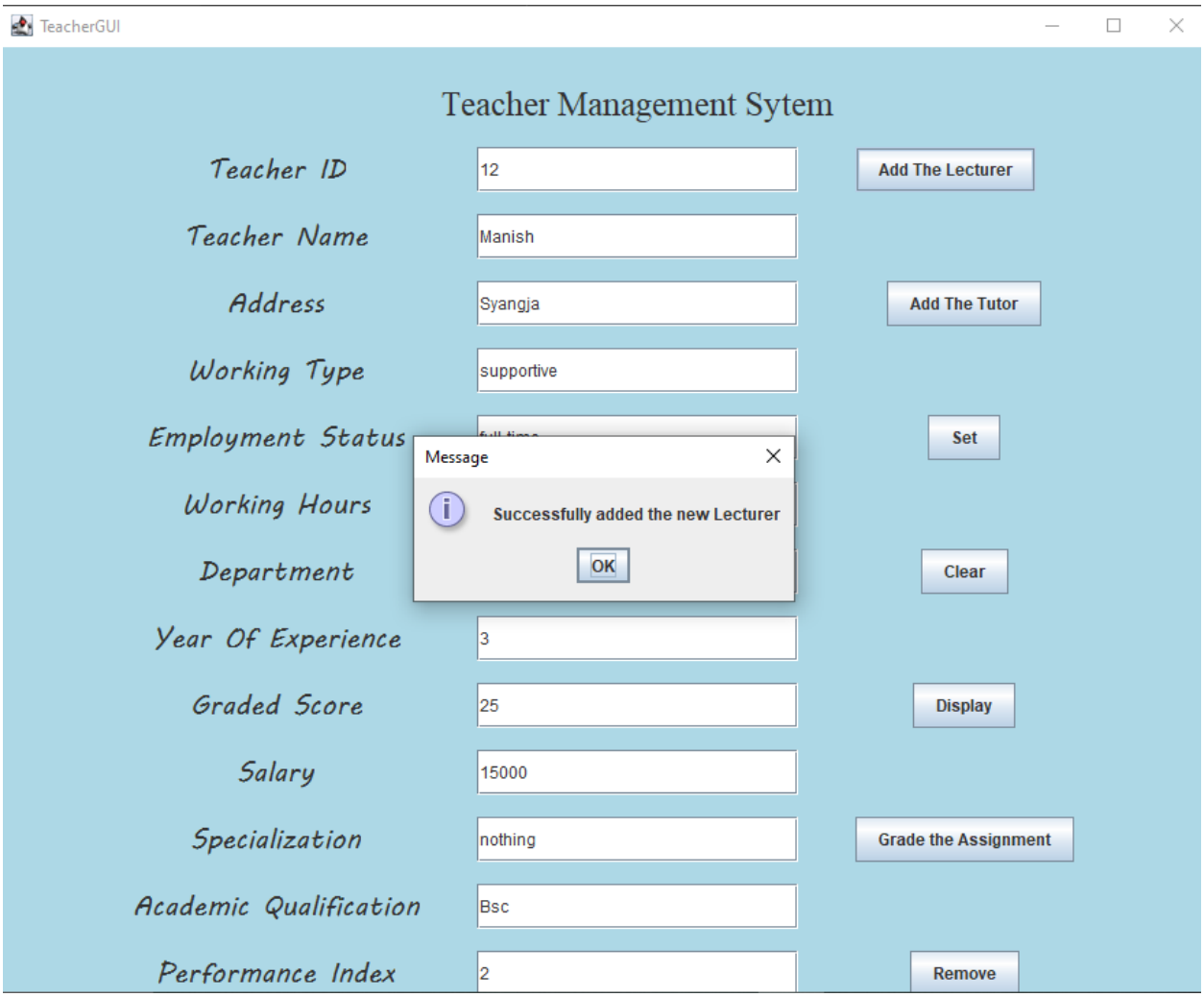

Figure 13: Adding the new Lecturer

b. Add the Tutor

Objective	To add the Tutor
Action	All the empty text fields were filled with below information: Teacher ID=12 Teacher Name = Manish Address = Syangja Working Type = supportive Employment Status = full-time Working Hours = 5 Salary =15000 Specialization =nothing Acedemic Qualification=Bsc Performance Index =2
Expected Result	The new Tutor will be added.
Actual Result	The new Tutor was added.
Conclusion	The test was successful.

Table 3: Adding the Tutor.

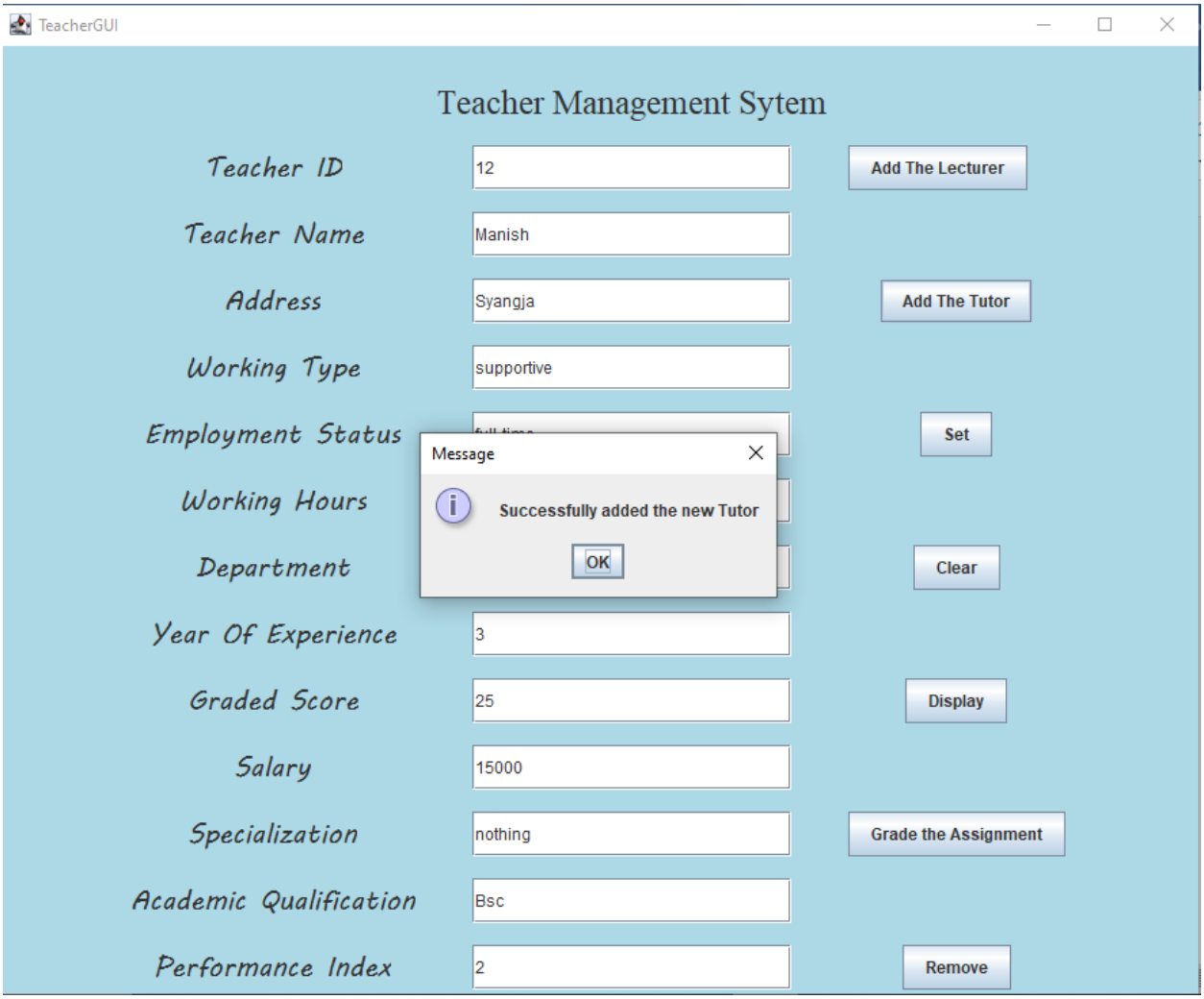

Figure 14: Adding the new Tutor

c. Grade the assignment from Lecturer

Objective	To Grade Assignments from Lecturer
Action	All the empty text fields were filled with below information: Teacher ID=12 Grade Score=25 Department =personal. Year Of Experience=3
Expected Result	The assignment graded information will be displayed.
Actual Result	The assignment graded information was displayed.
Conclusion	The test was successful.

Table 4: Grading Assignments from Lecturer.

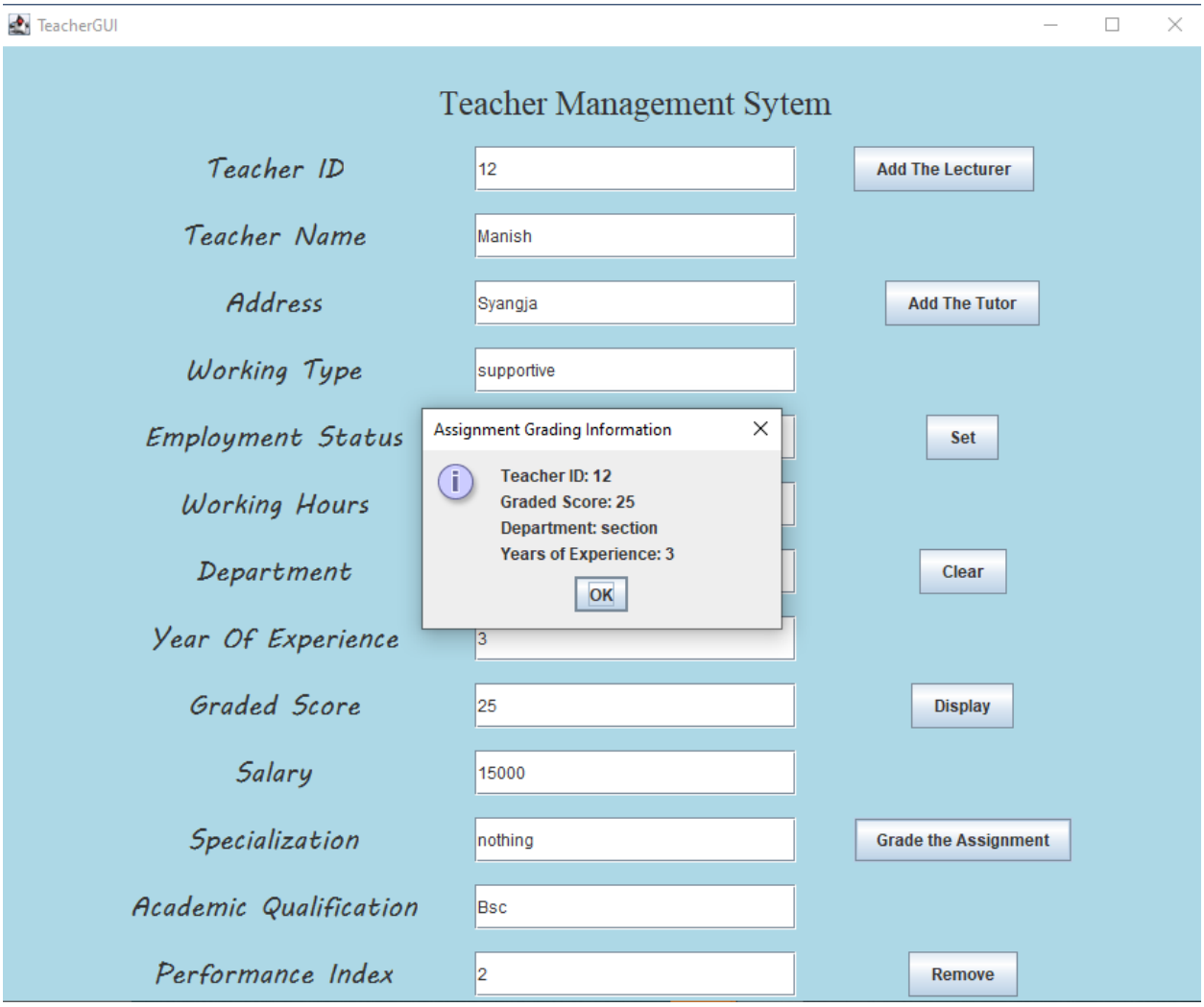

Figure 15: Grading the Assignments from Lecturer

d. Set the Salary

Objective	To set the salary.
Action	All the empty text fields were filled with below information: Teacher ID=12 Performance Index =2 Salary =15000
Expected Result	The current Salary will be displayed.
Actual Result	The current Salary was displayed.
Conclusion	The test was successful.

Table 5: Setting the salary.

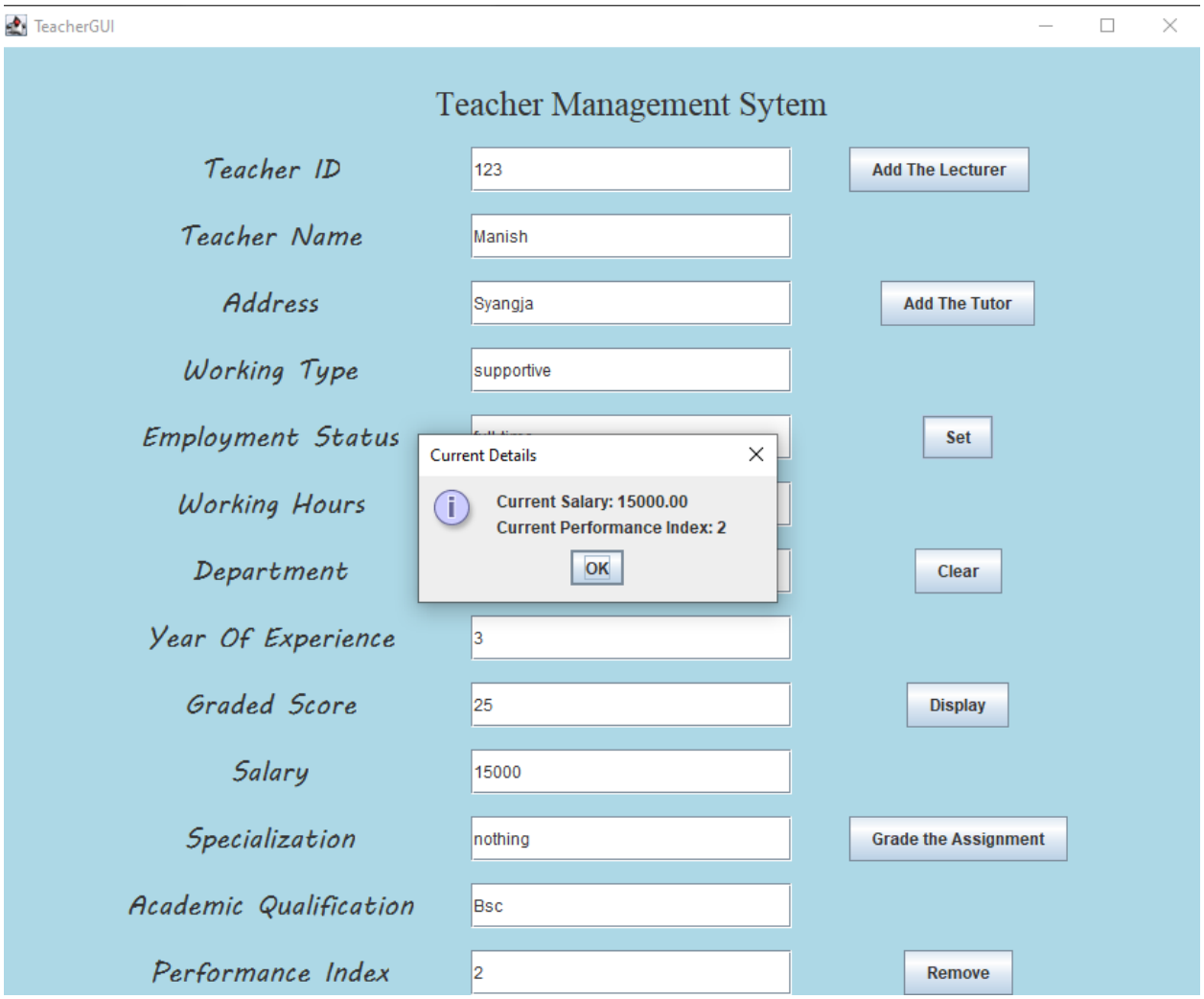

Figure 16: setting the salary.

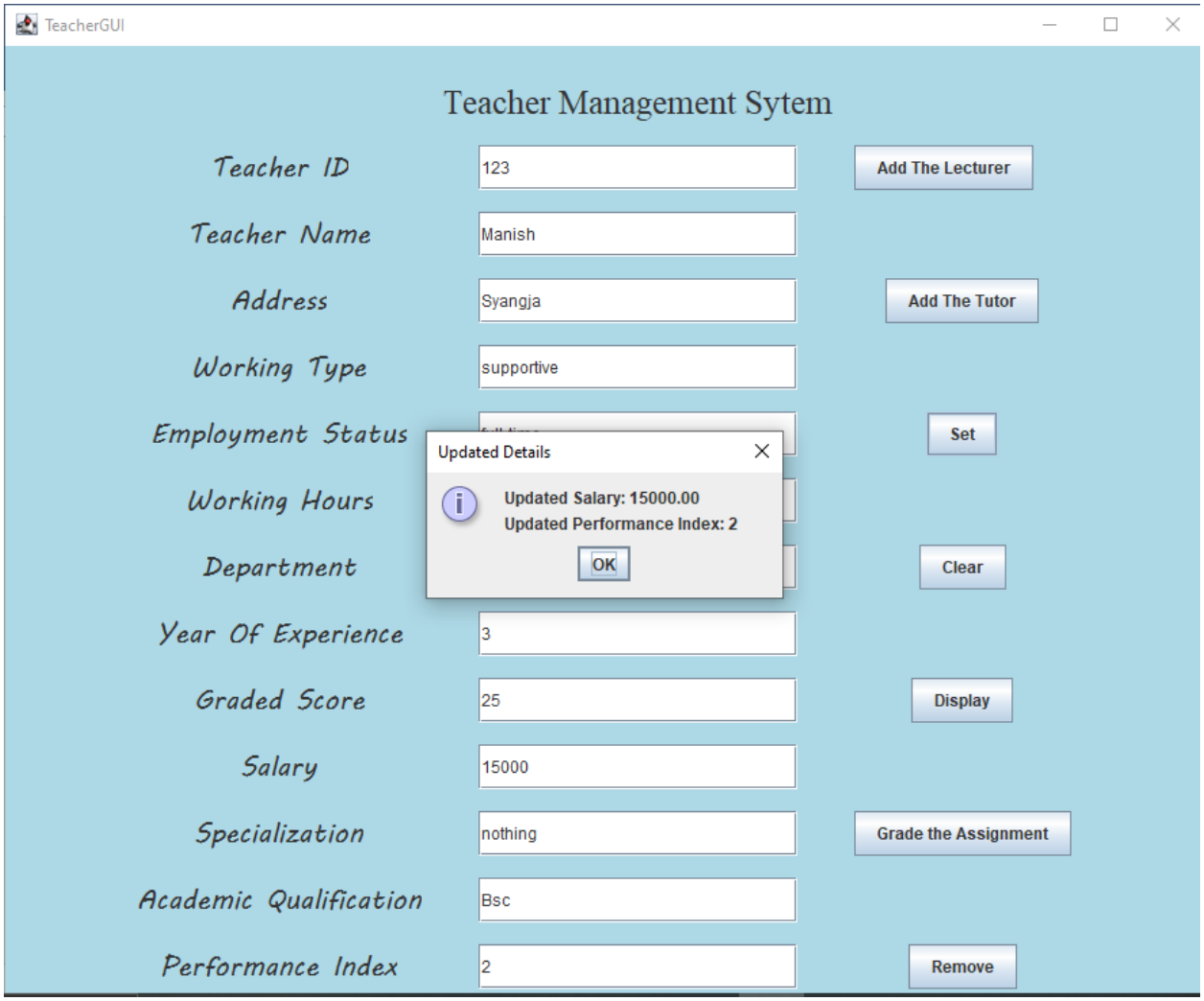

Figure 17:Updated Salary

e. Remove the Tutor

Objective	To remove the Tutor
Action	This action can only be performed after adding Tutor.
Expected Result	The Tutor will be removed.
Actual Result	The Tutor was removed.
Conclusion	The test was successful.

Table 6: Removing the Tutor.

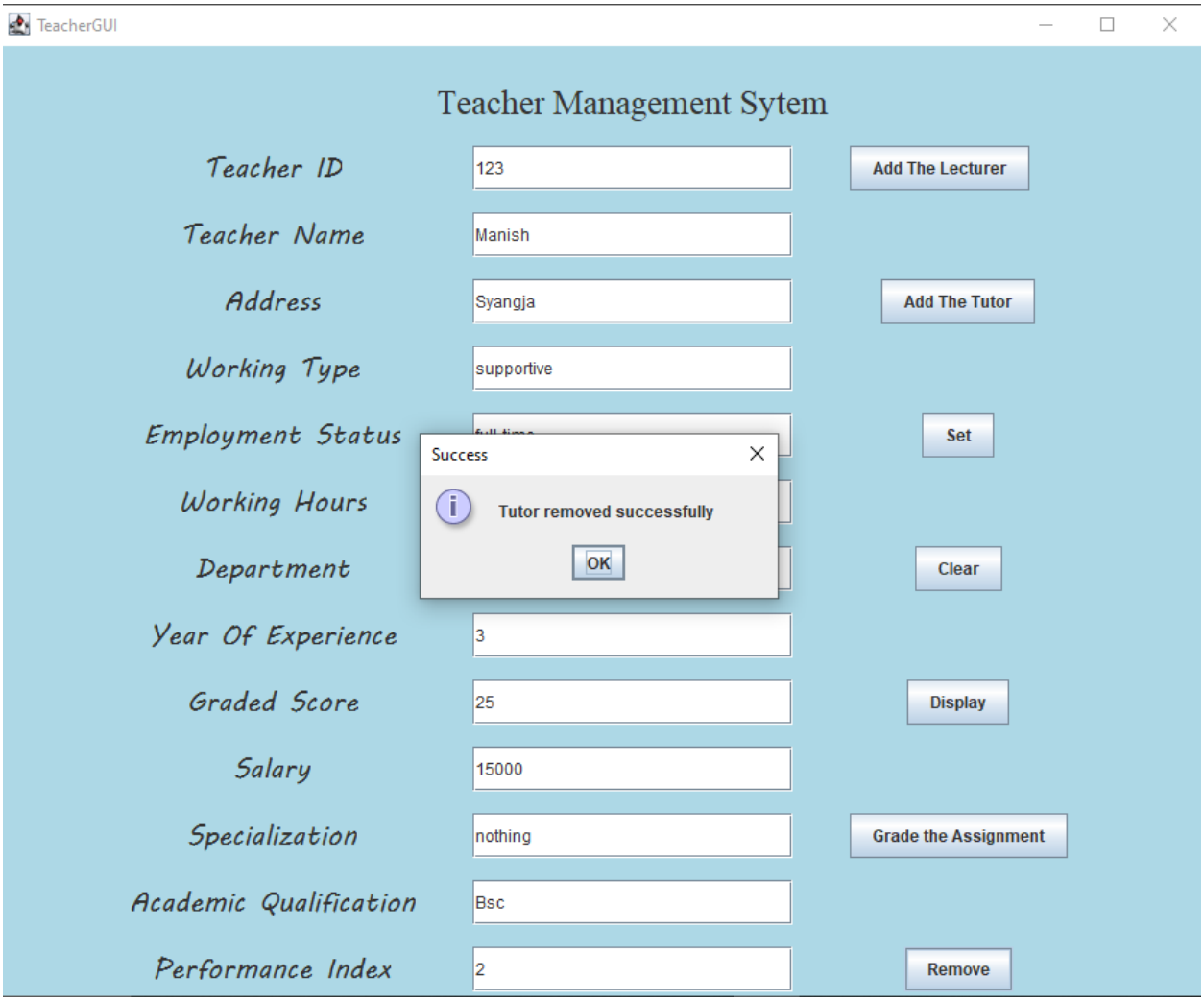

Figure 18: Removing the Tutor

6.3 Test 3: Testing appropriate dialog boxes appear when unsuitable values are entered.

a. Adding without filling the text field

Objective	To add the Lecturer button without filling the text field.
Action	Leave all the text fields empty and click to the Add The Lecturer.
Expected Result	After clicking Add The Lecturer button the message will be shown.
Actual Result	After clicking Add The Lecturer button the message was shown.
Conclusion	The test was successful.

Table 7: adding the Lecturer button without filling the text field.

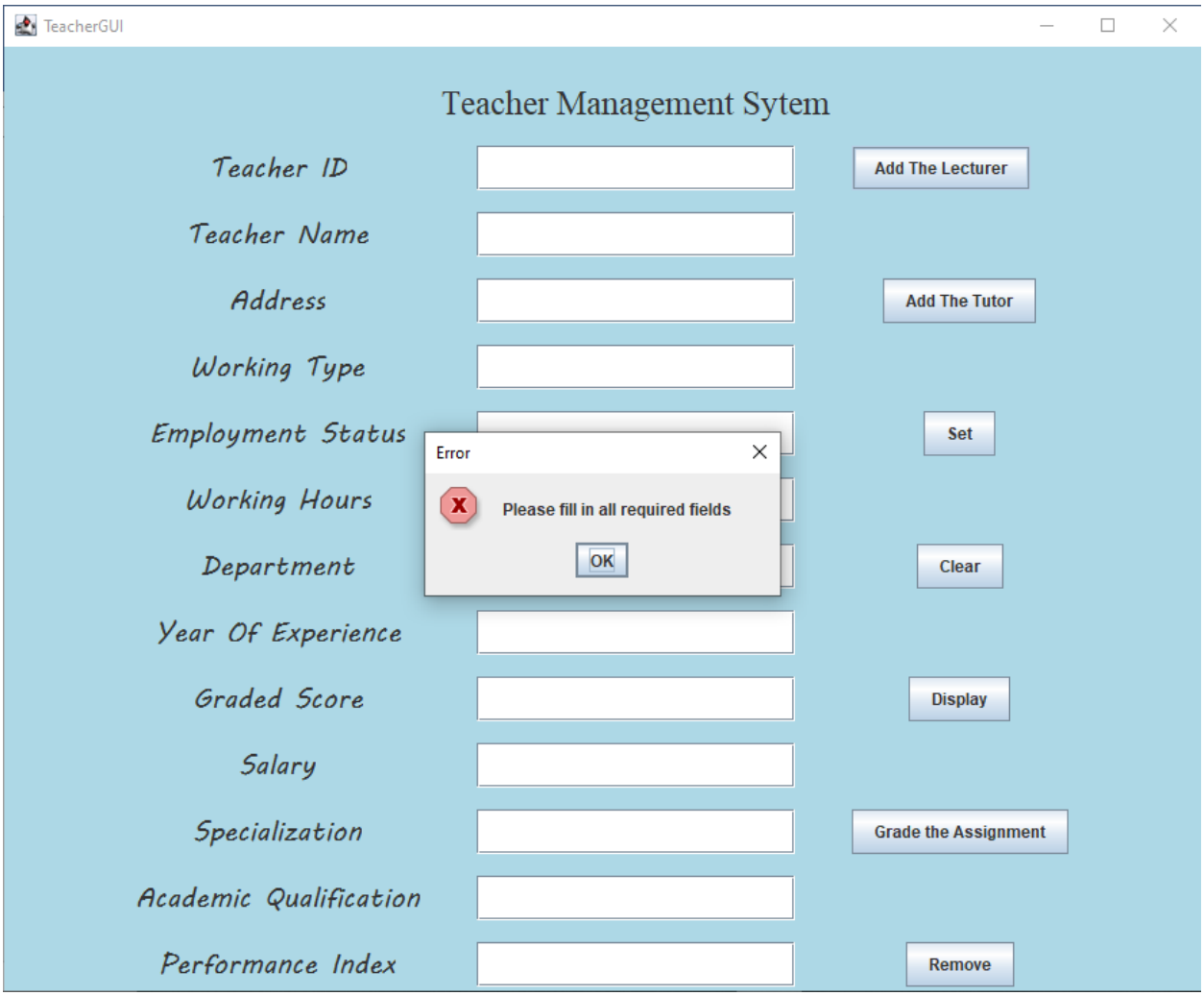

Figure 19: Adding without filling the text field.

b. Error Message for Out-of-Range IDs

Objective	To add the Teacher Id with Out-of-range id.
Action	Here, at first all the empty text fields required to add The Lecturer were filled with proper data. After that add The Lecturer button should be clicked.
Expected Result	After clicking Add The Lecturer button twice the message will be pop up.
Actual Result	After clicking Add The Lecturer button twice the message was pop up.
Conclusion	The test was successful.

Table 8: Adding the Teacher Id with Out-of-range id.

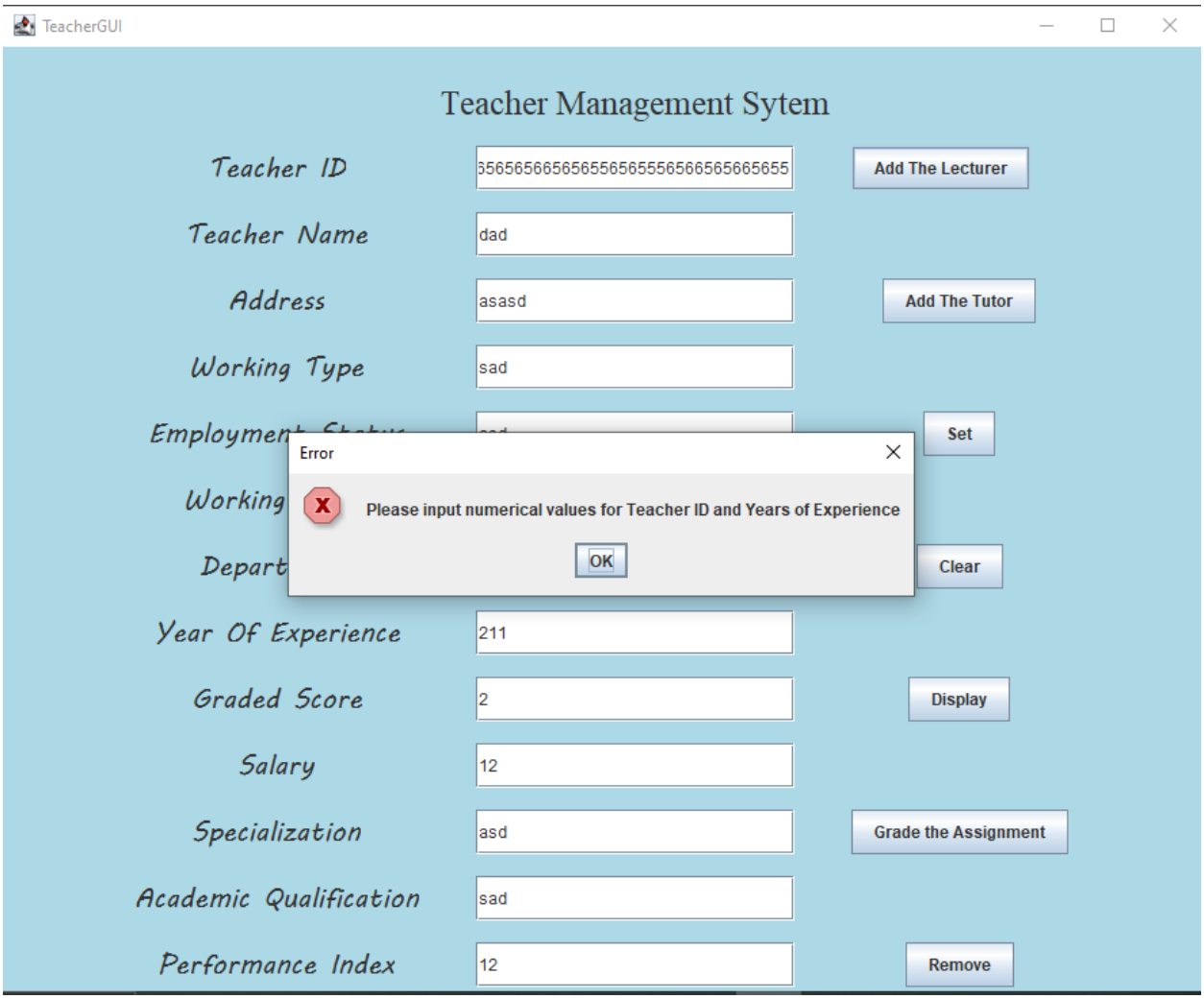

Figure 20: Adding the Teacher Id with Out-of-range id.

c. Adding different Teacher ID while setting the Tutor.

Objective	To add different Teacher ID while setting the Tutor.
Action	Here, at first all the empty text fields required to add The Tutor were filled with proper data. After that add The Tutor button should be clicked. Again the removing the teacher id another teacher ID should be filled.
Expected Result	After clicking Set button the message will be pop up.
Actual Result	After clicking Set button the message was pop up.
Conclusion	The test was successful.

Table 9: Adding different Teacher ID while setting the Tutor.

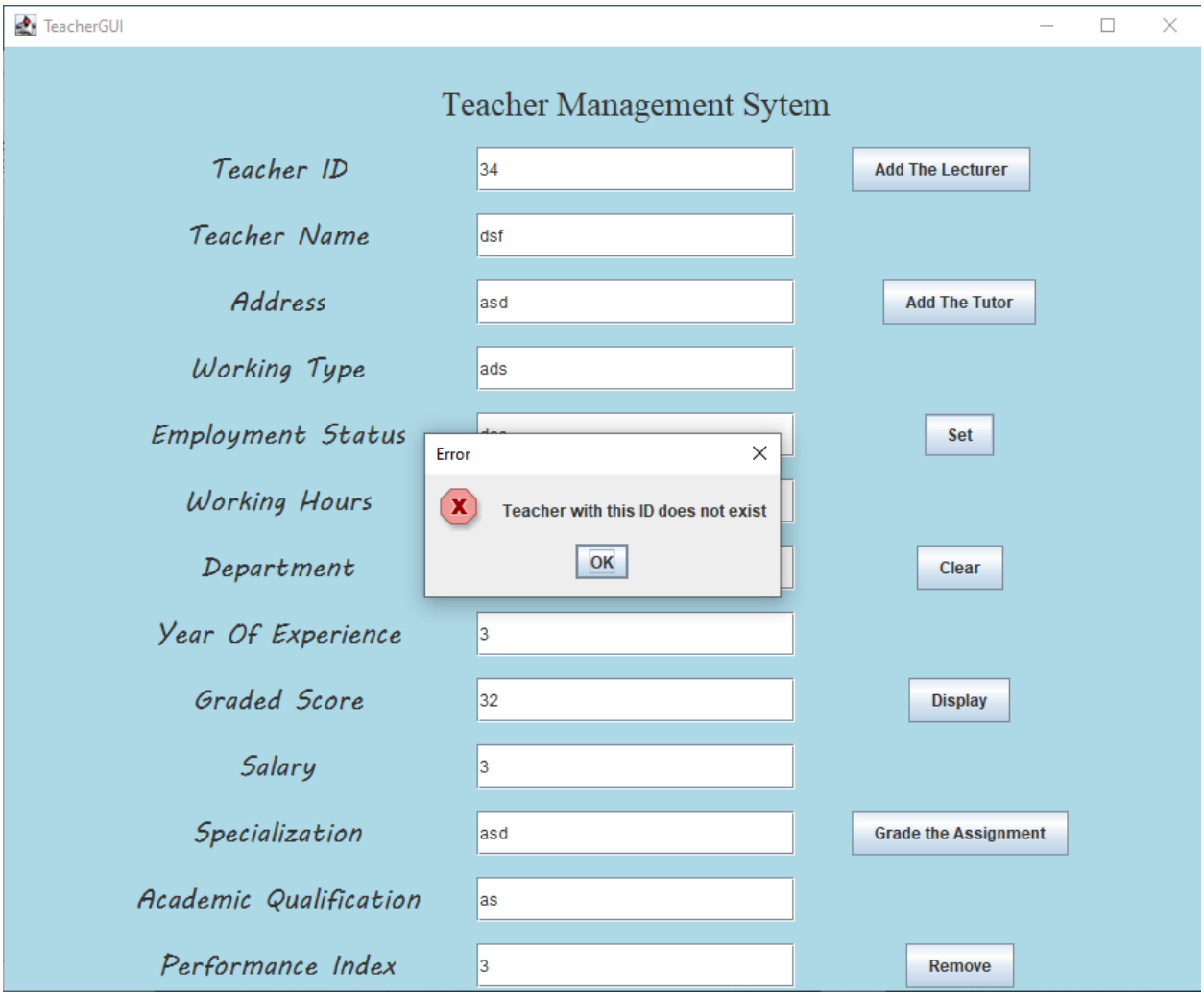

Figure 21: Adding different Teacher ID while setting the Tutor.

7. Error Detection and Error Correction

7.1.Syntax Error

Syntax errors generally appears during analysis phase. This error is found during the compilation of the program. Some syntax error can occur due to missing operators, imbalance in parenthesis, or error in structure (Javatpoint, 2021)

Error:

I forget to include curly brackets.

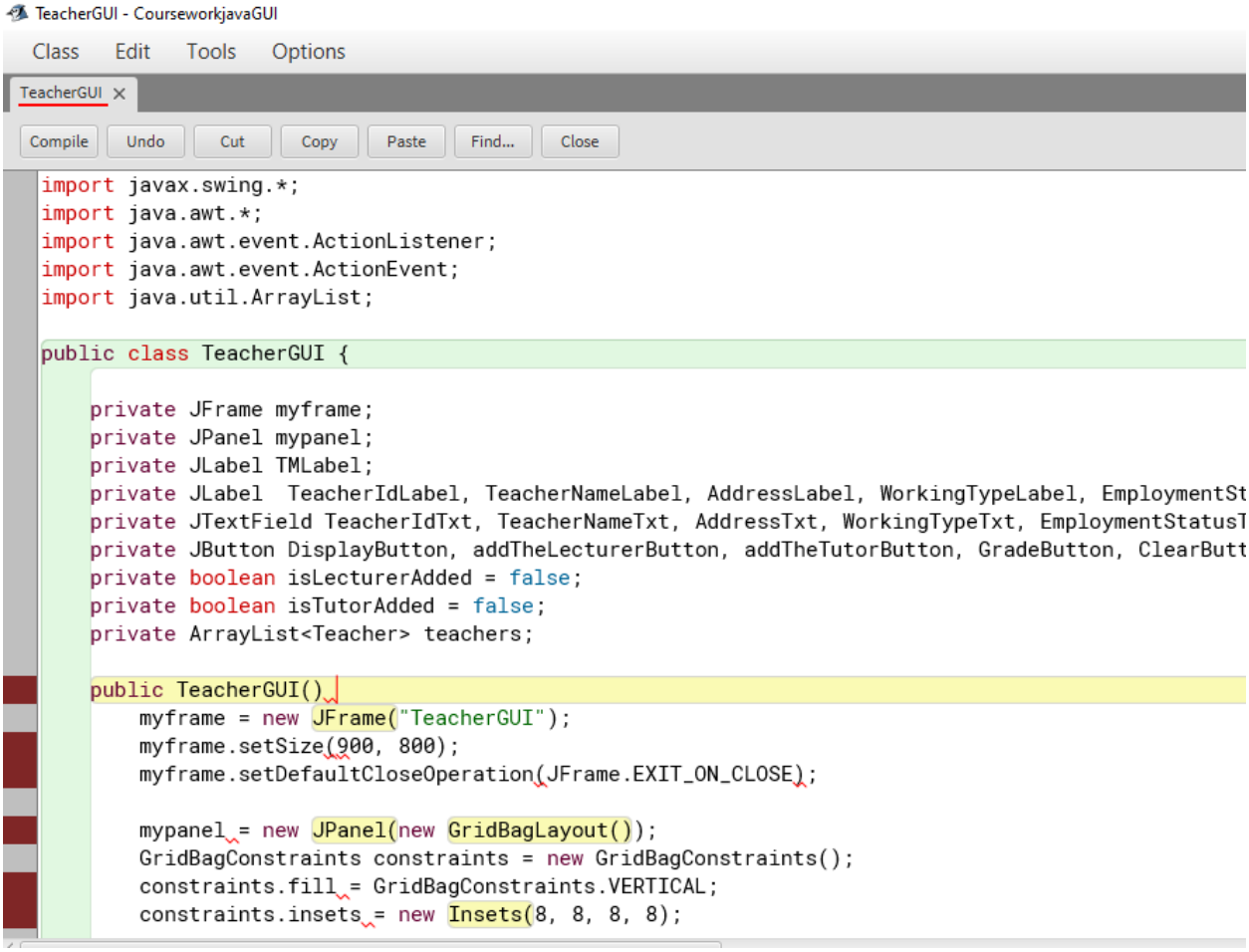

Figure 22: Syntax Error

Correction:

After including the curly bracket.

TeacherGUI - CourseworkjavaGUI

Class Edit Tools Options

TeacherGUI

Compile Undo Cut Copy Paste Find... Close

```
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.util.ArrayList;

public class TeacherGUI {

    private JFrame myframe;
    private JPanel mypanel;
    private JLabel TMLabel;
    private JLabel  TeacherIdLabel, TeacherNameLabel, AddressLabel, WorkingTypeLabel, Emplc
    private JTextField TeacherIdTxt, TeacherNameTxt, AddressTxt, WorkingTypeTxt, Employment
    private JButton DisplayButton, addTheLecturerButton, addTheTutorButton, GradeButton, Cl
    private boolean isLecturerAdded = false;
    private boolean isTutorAdded = false;
    private ArrayList<Teacher> teachers;

    public TeacherGUI() {
        myframe = new JFrame("TeacherGUI");
        myframe.setSize(900, 800);
        myframe.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        mypanel = new JPanel(new GridBagLayout());
        GridBagConstraints constraints = new GridBagConstraints();
        constraints.fill = GridBagConstraints.VERTICAL;
        constraints.insets = new Insets(8, 8, 8, 8);

        TMLabel = new JLabel("Teacher Management Sytem");
```

Figure 23: Correction of Syntax Error

2. Semantics error:

Semantic errors encompass those detected during compile time. These errors arise when an incorrect variable is utilized or when operations are executed in an improper sequence (Javatpoint, 2021)

Examples of semantic errors include:

- Operand types that are incompatible.

- Variables that are undeclared.
- Actual arguments that do not match formal arguments.

Error:

I forget to add the variable mypanel.

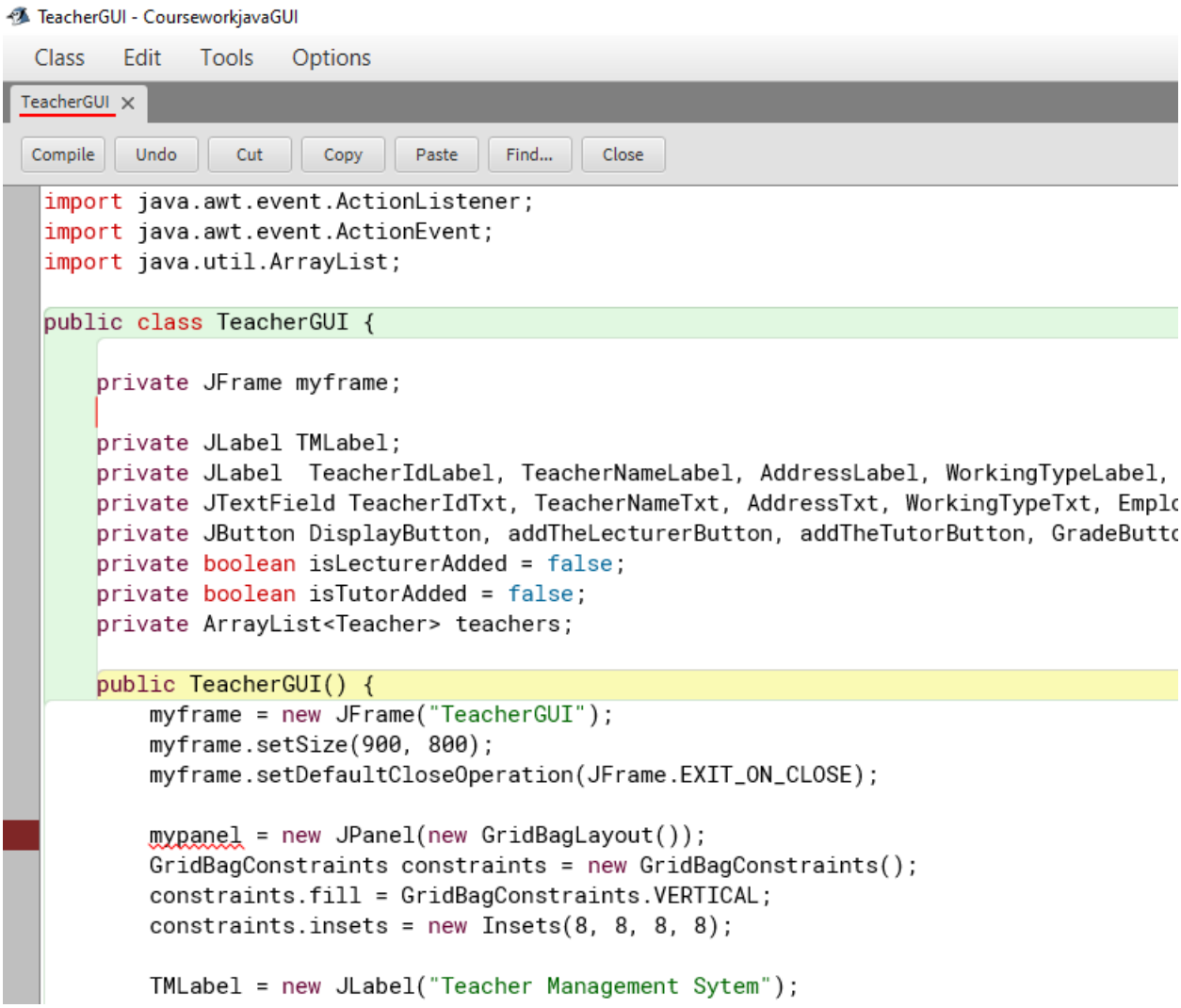

Figure 24: Semantics Error.

Correction:

After declaring the variable mypanel.

TeacherGUI - CourseworkjavaGUI

Class Edit Tools Options

TeacherGUI

Compile Undo Cut Copy Paste Find... Close

```
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.util.ArrayList;

public class TeacherGUI {

    private JFrame myframe;
    private JPanel mypanel;
    private JLabel TMLabel;
    private JLabel  TeacherIdLabel, TeacherNameLabel, AddressLabel, WorkingTypeLabel, Emplc
    private JTextField TeacherIdTxt, TeacherNameTxt, AddressTxt, WorkingTypeTxt, Employment
    private JButton DisplayButton, addTheLecturerButton, addTheTutorButton, GradeButton, Cl
    private boolean isLecturerAdded = false;
    private boolean isTutorAdded = false;
    private ArrayList<Teacher> teachers;

    public TeacherGUI() {
        myframe = new JFrame("TeacherGUI");
        myframe.setSize(900, 800);
        myframe.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        mypanel = new JPanel(new GridBagLayout());
        GridBagConstraints constraints = new GridBagConstraints();
        constraints.fill = GridBagConstraints.VERTICAL;
        constraints.insets = new Insets(8, 8, 8, 8);

        TMLabel = new JLabel("Teacher Management Sytem");
```

Figure 25: Correction of Semantics Error

3. Logical Error:

Logic errors stand apart from other types of errors in that they do not trigger any Java errors. Programs containing logic flaws will compile, execute, and terminate without issue, yet they will fail to produce the anticipated results ((Burnham, 2023).

Error:

The logical error was that code currently checks if the selected teacher is an instance of Tutor, but it should check if the teacher is an instance of Lecturer.

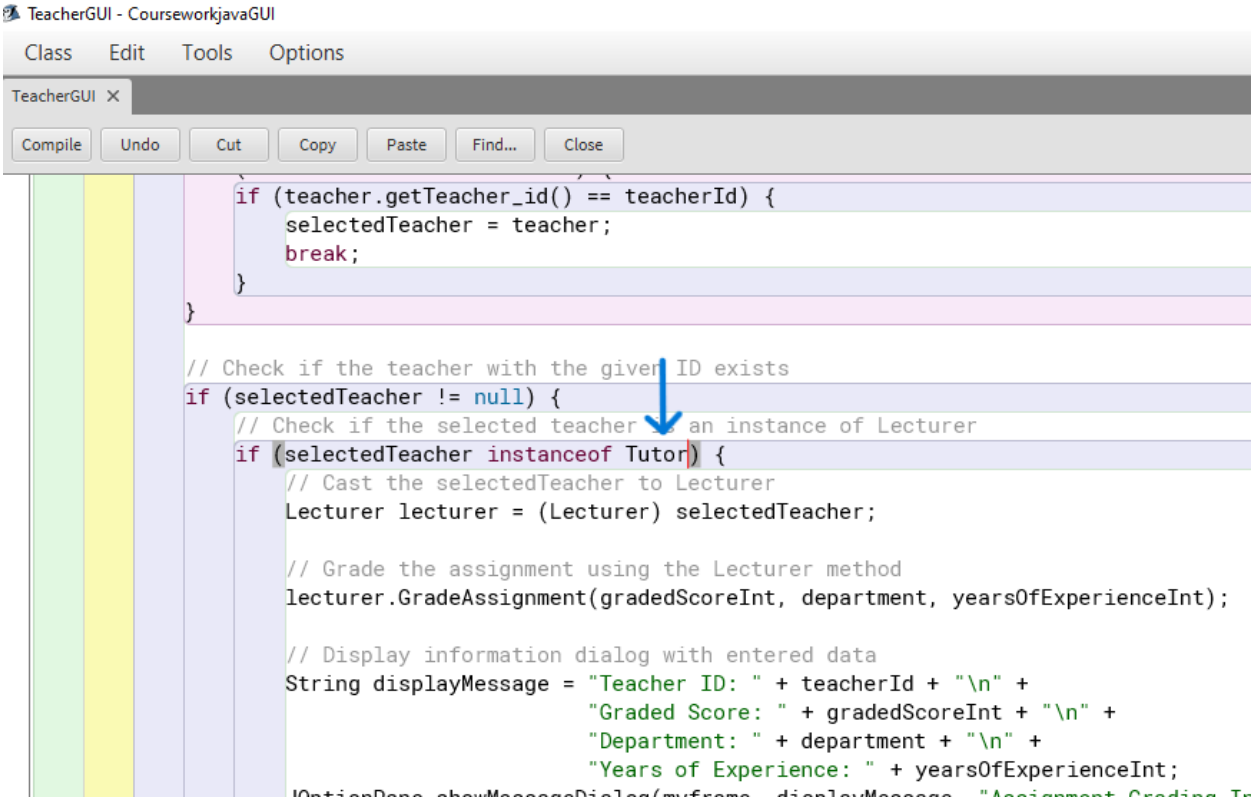

Figure 26: Logical Error

Correction:

I corrected the error by modifying to check if the selected teacher is an instance of Lecturer.

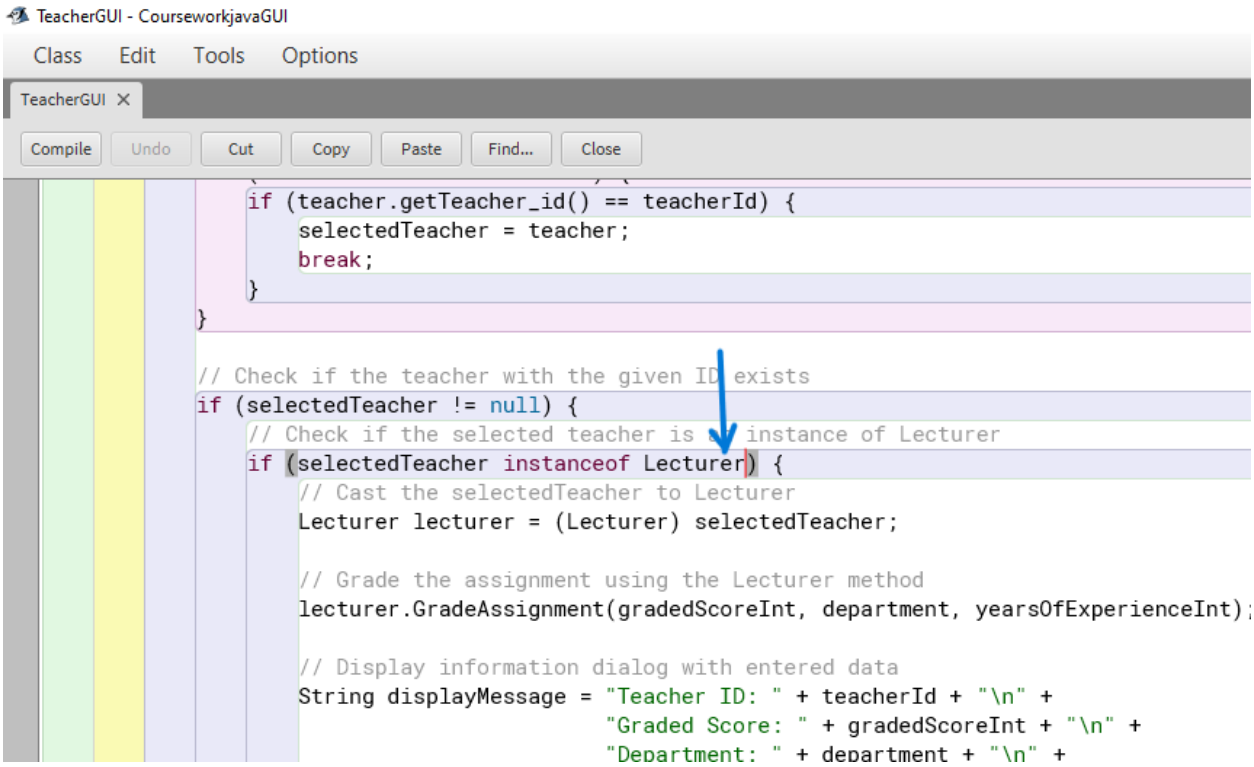

Figure 27: Correction of Logical Error

8. Conclusion

In conclusion, this report sheds light on essential concepts in Graphical User Interface (GUI), exception handling, and event handling in Java Programming. Despite having a solid foundation in Java Programming, completing this coursework presented its challenges. Exploring GUI, exception handling, and event handling for the first time introduced various hurdles, particularly in assigning functionalities to GUI buttons. Utilizing tools like moqups facilitated the creation of GUI wireframes, streamlining the

design process. The program incorporates multiple GUI components like buttons, labels, text fields, and panels, enhancing its interactivity.

During the coding phase, I gained a deeper understanding of how the program operates. Each button is equipped with specific functions, contributing to event handling, while managing unforeseen events falls under exception handling. This coursework paves the way for the creation of innovative applications in the future, grounded in related concepts. It also deepened my knowledge of Java, presenting both challenges and excitement in application development. This report provides a comprehensive overview of three critical Java programming concepts: GUI, exception handling, and event handling. The code implementation successfully achieves its intended objectives and holds potential for reliable application across various contexts.

In conclusion, this Java program report has effectively met its anticipated objectives and requirements, validated through thorough execution and testing phases.

9. Bibliography

Bibliography

Boat, K. (2023). *Knowledge Boat*. Retrieved from Knowledge Boat: https://www.knowledgeboat.com/question/what-is-bluej-what-are-the-features-of-bluej--19075147497221724

Burnham, D. (2023). *Burnham, D*. Retrieved from Burnham, D: https://codehs.com/tutorial/david/errors-in-java

Educba. (n.d.). *Educba*. Retrieved from Educba: https://www.educba.com/what-is-gui/

Hope, C. (2021). *Computer Hope*. Retrieved from Computer Hope: https://www.computerhope.com/jargon/m/microsoft-word.htm

Hope, C. (2023). *Computer Hope*. Retrieved from Computer Hope: https://www.computerhope.com/jargon/d/drawio.htm

IBM. (2024). *IBM*. Retrieved from IBM: https://www.ibm.com/topics/java

Javatpoint. (2021). Retrieved from Javatpoint: https://www.javatpoint.com/semantic-error

Javatpoint. (2021). Retrieved from https://www.javatpoint.com/syntax-error

Javatpoint. (2021). Retrieved from https://www.javatpoint.com/java-actionlistener

Themes, E. (2022). *Elegant Themes*. Retrieved from Elegant Themes: https://www.elegantthemes.com/blog/design/moqups-an-overview-and-review

Venngage. (2023). *Venngage*. Retrieved from Venngage: https://venngage.com/blog/class-diagram/

10. Appendix

10.1 Appendix of Teacher.java

```
/**
 * Write a description of class Teacher here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Teacher
{
   private int teacher_id;
    private String teacher_name;
    private String address;
    private String working_type;
    private String employment_status;
    private int working_hour;

    public Teacher(int teacher_id, String teacher_name, String address, String
working_type,String employment_status)
    {
        this.teacher_id=teacher_id;
```

```
    this.teacher_name=teacher_name;
    this.address=address;
    this.working_type=working_type;
    this.employment_status=employment_status;

}

public int getTeacher_id()
{
    return this.teacher_id;
}

public String getTeacher_name()
{
    return this.teacher_name;
}

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

public String getWorking_type()
{
    return this.working_type;
}

public String getEmployment_status()
```

```
    {
        return this.employment_status;
    }
   public void setWorking_hour( int working_hour)
    {
        this.working_hour=working_hour;
    }
   public int getWorking_hour()
    {
        return this.working_hour;
    }
       public void displayDetails() {
      System.out.println("Teacher ID: " + teacher_id);
      System.out.println("Teacher Name: " + teacher_name);
      System.out.println("Address: " + address);
      System.out.println("Working Type: " + working_type);
      System.out.println("Employment Status: " + employment_status);

      if  (working_hour == 0) {
          System.out.println("Working Hours: Not assigned");
      } else {
          System.out.println("Working Hours: " + working_hour);
      }
  }


}
```

10.2 Appendix of Lecturer.java

```
/**
 * Write a description of class Lecturer here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Lecturer extends Teacher
{
    private String Department;
    private int YearOfExperience;
    private int gradedScore;
    private boolean hasGraded;

    public Lecturer( int teacher_id, String teacher_name, String address, String
working_type,String employment_status, String Department,int YearOfExperience)
    {
        super(teacher_id, teacher_name, address, working_type, employment_status);
        this.Department=Department;
        this.YearOfExperience=YearOfExperience;
        this.gradedScore=0;
```

```
        this.hasGraded=false;
    }

    public String getDepartment()
    {
        return this.Department;
    }

    public int getYearOfExperience()
    {
       return this.YearOfExperience;
    }

    public int getGradedScore()
    {
        return this.gradedScore;
    }
     public void setGradedScore(int newGradedScore) {
        this.gradedScore = newGradedScore;
    }

    public boolean getHasGraded()
    {
        return this.hasGraded;
    }
    public void GradeAssignment( int gradeScore, String Department, int
YearofExperience)
    {
            if (!hasGraded && YearOfExperience >= 5 && Department.equals(Department)) {
```

```
            if(gradeScore>=70){
                 gradedScore = gradeScore;
            } else if (gradeScore >= 60) {
                gradedScore = 60;
            } else if (gradeScore>= 50) {
                gradedScore = 50;
            } else if (gradeScore >= 40) {
                gradedScore = 40;
            } else {
                gradedScore = 0;
            }

            hasGraded = true;
        }
        else {
            System.out.println("Assignment not graded yet or conditions not met.");
        }
    }

public void displayDetails() {
        super.displayDetails();
        System.out.println("Department: " + Department);
        System.out.println("Years of Experience: " + YearOfExperience);
        if (hasGraded) {
            System.out.println("Graded Score: " + gradedScore);
        } else {
```

```
            System.out.println("Graded Score: Not graded yet");
        }
    }
}
```

10.3 Appendix of Tutor.java

```
/**
 * Write a description of class Tutor here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Tutor extends Teacher
{
    public double salary;
    public String specialization;
    public String academic_qualifications;
    public int performance_index;
    public boolean isCertified;



    public Tutor(int teacher_id, String teacher_name, String address, String
working_type, String employment_status,
    int working_hour, double salary, String specialization, String academic_qualifications,
    int performance_index) {
        super(teacher_id, teacher_name, address, working_type, employment_status);
        this.setWorking_hour(working_hour);
        this.salary = salary;
```

```
    this.specialization = specialization;
    this.academic_qualifications = academic_qualifications;
    this.performance_index = performance_index;
    this.isCertified = false;
}


public double getSalary() {
    return this.salary;
}

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

public String getAcademic_qualifications() {
    return this.academic_qualifications;
}

public int getPerformance_index() {
    return this.performance_index;
}

public boolean getisCertified() {
    return this.isCertified;
}
```

```
public void setSalaryAndCertification(double newSalary, int newPerformance_index) {
    if (newPerformance_index > 5 && getWorking_hour() > 20) {
        double appraisalPercentage;
        if (newPerformance_index >= 5 && newPerformance_index <= 7) {
            appraisalPercentage = 0.05;
        } else if (newPerformance_index >= 8 && newPerformance_index <= 9) {
            appraisalPercentage = 0.1;
        } else {
            appraisalPercentage = 0.2;
        }

        salary = newSalary + (appraisalPercentage * newSalary);
        isCertified = true;
    } else {
        System.out.println("Salary cannot be approved. Tutor is not certified yet.");
    }
}


public void removeTutor() {
    if (!isCertified) {
        salary = 0;
        specialization = "";
        academic_qualifications = "";
        performance_index = 0;
        isCertified = false;
    }
}
```

```
    public void displayDetails() {
        super.displayDetails();

        if (isCertified) {
            System.out.println("Salary: " + salary);
            System.out.println("Specialization: " + specialization);
            System.out.println("Academic Qualifications: " + academic_qualifications);
            System.out.println("Performance Index: " + performance_index);
        }
    }

}
```

10.4 Appendix of TeacherGUI

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import java.util.ArrayList;


public class TeacherGUI {


    private JFrame myframe;

    private JPanel mypanel;

    private JLabel TMLabel;

    private JLabel  TeacherIdLabel, TeacherNameLabel, AddressLabel,
WorkingTypeLabel, EmploymentStatusLabel, WorkingHoursLabel, DepartmentLabel,
YearsOfExperienceLabel,YearOfExperienceLabel, GradedScoreLabel, SalaryLabel,
SpecilizationLabel, AcademicQualificationLabel, PerformanceIndexLabel;

    private JTextField TeacherIdTxt, TeacherNameTxt, AddressTxt, WorkingTypeTxt,
EmploymentStatusTxt, WorkingHoursTxt, DepartmentTxt,
YearOfExperienceTxt,GradedScoreTxt, SalaryTxt, SpecilizationTxt,
AcademicQualificationTxt, PerformanceIndexTxt;

    private JButton DisplayButton, addTheLecturerButton, addTheTutorButton,
GradeButton, ClearButton, setButton, removeButton;

    private boolean isLecturerAdded = false;

    private boolean isTutorAdded = false;

    private ArrayList<Teacher> teachers;


    public TeacherGUI() {

        myframe = new JFrame("TeacherGUI");

        myframe.setSize(900, 800);

        myframe.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


        mypanel = new JPanel(new GridBagLayout());
```

```
GridBagConstraints constraints = new GridBagConstraints();
constraints.fill = GridBagConstraints.VERTICAL;
constraints.insets = new Insets(8, 8, 8, 8);

TMLabel = new JLabel("Teacher Management Sytem");
TMLabel.setFont(new Font("Times New Roman", Font.PLAIN, 25));

TeacherIdLabel = new JLabel("Teacher ID");
TeacherIdTxt = new JTextField(21);
TeacherIdLabel.setFont(new Font("MV Boli", Font.PLAIN, 20));

TeacherNameLabel = new JLabel("Teacher Name");
TeacherNameTxt = new JTextField(21);
TeacherNameLabel.setFont(new Font("MV Boli", Font.PLAIN, 20));

AddressLabel = new JLabel("Address");
AddressTxt = new JTextField(21);
AddressLabel.setFont(new Font("MV Boli", Font.PLAIN, 20));

WorkingTypeLabel = new JLabel("Working Type");
WorkingTypeTxt = new JTextField(21);
WorkingTypeLabel.setFont(new Font("MV Boli", Font.PLAIN, 20));

EmploymentStatusLabel = new JLabel("Employment Status");
EmploymentStatusTxt = new JTextField(21);
EmploymentStatusLabel.setFont(new Font("MV Boli", Font.PLAIN, 20));

WorkingHoursLabel = new JLabel("Working Hours");
```

```
WorkingHoursTxt = new JTextField(21);
WorkingHoursLabel.setFont(new Font("MV Boli", Font.PLAIN, 20));

DepartmentLabel = new JLabel("Department");
DepartmentTxt = new JTextField (21);
DepartmentLabel.setFont(new Font("MV Boli", Font.PLAIN, 20));

YearOfExperienceLabel = new JLabel("Year Of Experience");
YearOfExperienceTxt = new JTextField (21);
YearOfExperienceLabel.setFont(new Font("MV Boli", Font.PLAIN, 20));

GradedScoreLabel = new JLabel("Graded Score");
GradedScoreTxt = new JTextField(21);
GradedScoreLabel.setFont(new Font("MV Boli", Font.PLAIN, 20));

SalaryLabel = new JLabel("Salary");
SalaryTxt = new JTextField(21);
SalaryLabel.setFont(new Font("MV Boli", Font.PLAIN, 20));

SpecilizationLabel = new JLabel("Specialization");
SpecilizationTxt = new JTextField(21);
SpecilizationLabel.setFont(new Font("MV Boli", Font.PLAIN, 20));

AcademicQualificationLabel = new JLabel("Academic Qualification");
AcademicQualificationTxt = new JTextField(21);
AcademicQualificationLabel.setFont(new Font("MV Boli", Font.PLAIN, 20));

PerformanceIndexLabel = new JLabel("Performance Index");
```

```
PerformanceIndexTxt = new JTextField(21);
PerformanceIndexLabel.setFont(new Font("MV Boli", Font.PLAIN, 20));



teachers = new ArrayList<>();




addTheLecturerButton = new JButton("Add The Lecturer");
addTheLecturerButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {


        if (TeacherIdTxt.getText().isEmpty() || TeacherNameTxt.getText().isEmpty() ||
AddressTxt.getText().isEmpty() || WorkingTypeTxt.getText().isEmpty() ||
EmploymentStatusTxt.getText().isEmpty() || GradedScoreTxt.getText().isEmpty() ||
YearOfExperienceTxt.getText().isEmpty()) {
        JOptionPane.showMessageDialog(myframe, "Please fill in all required fields",
"Error", JOptionPane.ERROR_MESSAGE);
    } else {
        try {
            int teacherId = Integer.parseInt(TeacherIdTxt.getText());
            String teacherName = TeacherNameTxt.getText();
            String address = AddressTxt.getText();
            String workingType = WorkingTypeTxt.getText();
            String employmentStatus = EmploymentStatusTxt.getText();
            String gradedScore = GradedScoreTxt.getText();
            int yearsOfExperience = Integer.parseInt(YearOfExperienceTxt.getText());


            Lecturer newLecturer = new Lecturer(teacherId, teacherName, address,
workingType, employmentStatus, gradedScore, yearsOfExperience);
```

```
            teachers.add(newLecturer);
             isLecturerAdded = true;
            JOptionPane.showMessageDialog(myframe, "Successfully added the new
Lecturer");
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(myframe, "Please input numerical values
for Teacher ID and Years of Experience", "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
  }
});



    addTheTutorButton = new JButton("Add The Tutor");
    addTheTutorButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {



    if (TeacherIdTxt.getText().isEmpty() || TeacherNameTxt.getText().isEmpty() ||
AddressTxt.getText().isEmpty() || WorkingTypeTxt.getText().isEmpty() ||
EmploymentStatusTxt.getText().isEmpty() || WorkingHoursTxt.getText().isEmpty() ||
SalaryTxt.getText().isEmpty() || SpecilizationTxt.getText().isEmpty() ||
AcademicQualificationTxt.getText().isEmpty() ||
PerformanceIndexTxt.getText().isEmpty()) {
        JOptionPane.showMessageDialog(myframe, "Please fill in all required fields",
"Error", JOptionPane.ERROR_MESSAGE);
    } else {
        try {
```

```
            int teacherId = Integer.parseInt(TeacherIdTxt.getText());
            String teacherName = TeacherNameTxt.getText();
            String address = AddressTxt.getText();
            String workingType = WorkingTypeTxt.getText();
            String employmentStatus = EmploymentStatusTxt.getText();
            int workingHours = Integer.parseInt(WorkingHoursTxt.getText());
            double salary = Double.parseDouble(SalaryTxt.getText());
            String specialization = SpecilizationTxt.getText();
            String academicQualifications = AcademicQualificationTxt.getText();
            int performanceIndex = Integer.parseInt(PerformanceIndexTxt.getText());



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



            teachers.add(newTutor);
            isTutorAdded = true;



            JOptionPane.showMessageDialog(myframe, "Successfully added the new
Tutor");
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(myframe, "Please input numerical values
for Teacher ID, Working Hours, Salary, and Performance Index", "Error",
JOptionPane.ERROR_MESSAGE);
        }
    }
```

```
        }
    });

    GradeButton = new JButton("Grade the Assignment");
    GradeButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
    String teacherIdInput = TeacherIdTxt.getText();
    String gradedScore = GradedScoreTxt.getText();
    String department = DepartmentTxt.getText();
    String yearsOfExperience = YearOfExperienceTxt.getText();

    if (teacherIdInput.isEmpty() || gradedScore.isEmpty() || department.isEmpty() ||
yearsOfExperience.isEmpty()) {
        JOptionPane.showMessageDialog(myframe, "Please fill in all required fields",
"Error", JOptionPane.ERROR_MESSAGE);
        return;
    }

    try {

        int teacherId = Integer.parseInt(teacherIdInput);
        int gradedScoreInt = Integer.parseInt(gradedScore);
        int yearsOfExperienceInt = Integer.parseInt(yearsOfExperience);

        Teacher selectedTeacher = null;
        for (Teacher teacher : teachers) {
            if (teacher.getTeacher_id() == teacherId) {
                selectedTeacher = teacher;
```

```
                break;
            }
        }

        if (selectedTeacher != null) {
            if (selectedTeacher instanceof Lecturer) {

                Lecturer lecturer = (Lecturer) selectedTeacher;

                lecturer.GradeAssignment(gradedScoreInt, department,
yearsOfExperienceInt);
                String displayMessage = "Teacher ID: " + teacherId + "\n" +
                                "Graded Score: " + gradedScoreInt + "\n" +
                                "Department: " + department + "\n" +
                                "Years of Experience: " + yearsOfExperienceInt;
                JOptionPane.showMessageDialog(myframe, displayMessage, "Assignment
Grading Information", JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(myframe, "The selected teacher is not a
Lecturer", "Error", JOptionPane.ERROR_MESSAGE);
            }
        } else {
            JOptionPane.showMessageDialog(myframe, "No teacher found with the
entered ID", "Error", JOptionPane.ERROR_MESSAGE);
        }
    } catch (NumberFormatException ex) {
        JOptionPane.showMessageDialog(myframe, "Please input numerical values for
Teacher ID, Graded Score, and Years of Experience", "Error",
JOptionPane.ERROR_MESSAGE);
    }
```

```
    }
});


            DisplayButton = new JButton("Display");
        DisplayButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {

         String teacherIdInput = TeacherIdTxt.getText();
        String teacherName = TeacherNameTxt.getText();
        String address = AddressTxt.getText();
        String workingType = WorkingTypeTxt.getText();
        String employmentStatus = EmploymentStatusTxt.getText();

        String gradedScore = GradedScoreTxt.getText();
        String academicQualification = AcademicQualificationTxt.getText();

        String displayMessage = "Teacher ID: " + teacherIdInput + "\n";
        displayMessage += "Teacher Name: " + teacherName + "\n";
        displayMessage += "Address: " + address + "\n";
        displayMessage += "Working Type: " + workingType + "\n";
        displayMessage += "Employment Status: " + employmentStatus + "\n";

        displayMessage += "Graded Score: " + gradedScore + "\n";
        displayMessage += "Academic Qualification: " + academicQualification + "\n";

           if (isLecturerAdded) {
            // Additional field for lecturers
```

```
            String yearsOfExperience = YearOfExperienceTxt.getText();
            displayMessage += "Years of Experience: " + yearsOfExperience + "\n";
        }

        if (isTutorAdded) {
            String workingHours = WorkingHoursTxt.getText();
            String salary = SalaryTxt.getText();
            String specialization = SpecilizationTxt.getText();
            String performanceIndex = PerformanceIndexTxt.getText();

            displayMessage += "Working Hours: " + workingHours + "\n";
            displayMessage += "Salary: " + salary + "\n";
            displayMessage += "Specialization: " + specialization + "\n";
            displayMessage += "Performance Index: " + performanceIndex + "\n";
        }

        JOptionPane.showMessageDialog(myframe, displayMessage, "Teacher
Information", JOptionPane.INFORMATION_MESSAGE);

        JOptionPane.showMessageDialog(myframe, "Successfully Displayed");
    }
        });

        ClearButton = new JButton("Clear");
        ClearButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {

                TeacherIdTxt.setText("");
                TeacherNameTxt.setText("");
```

```
        AddressTxt.setText("");
        WorkingTypeTxt.setText("");
        EmploymentStatusTxt.setText("");
        WorkingHoursTxt.setText("");
        DepartmentTxt.setText("");
        YearOfExperienceTxt.setText("");
        GradedScoreTxt.setText("");
        SalaryTxt.setText("");
        SpecilizationTxt.setText("");
        AcademicQualificationTxt.setText("");
        PerformanceIndexTxt.setText("");

        JOptionPane.showMessageDialog(myframe, "Successfully Cleared");
    }
});

setButton = new JButton("Set");
setButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
          String teacherIdInput = TeacherIdTxt.getText();

if (!teacherIdInput.isEmpty()) {

    int teacherId = Integer.parseInt(teacherIdInput);



    for (Teacher teacher : teachers) {
```

```
            if (teacher.getTeacher_id() == teacherId) {

                if (teacher instanceof Tutor) {

                    Tutor tutor = (Tutor) teacher;



                    double currentSalary = tutor.getSalary();
                    int currentPerformanceIndex = tutor.getPerformance_index();
                    String message = String.format("Current Salary: %.2f\nCurrent
Performance Index: %d", currentSalary, currentPerformanceIndex);
                    JOptionPane.showMessageDialog(myframe, message, "Current
Details", JOptionPane.INFORMATION_MESSAGE);


                    String newSalaryInput = SalaryTxt.getText();
                    String newPerformanceIndexInput = PerformanceIndexTxt.getText();



                    if (!newSalaryInput.isEmpty() && !newPerformanceIndexInput.isEmpty())
{


                        double newSalary = Double.parseDouble(newSalaryInput);
                        int newPerformanceIndex =
Integer.parseInt(newPerformanceIndexInput);



                        tutor.setSalaryAndCertification(newSalary, newPerformanceIndex);


                        double updatedSalary = tutor.getSalary();
                        int updatedPerformanceIndex = tutor.getPerformance_index();
```

```
                    String updatedMessage = String.format("Updated Salary:
%.2f\nUpdated Performance Index: %d", updatedSalary, updatedPerformanceIndex);

                    JOptionPane.showMessageDialog(myframe, updatedMessage,
"Updated Details", JOptionPane.INFORMATION_MESSAGE);

                } else {


                    JOptionPane.showMessageDialog(myframe, "Please enter both new
salary and new performance index", "Error", JOptionPane.ERROR_MESSAGE);

                }

            } else {


                JOptionPane.showMessageDialog(myframe, "Teacher with this ID is not
a Tutor", "Error", JOptionPane.ERROR_MESSAGE);

            }




            return;

        }

    }


    JOptionPane.showMessageDialog(myframe, "Teacher with this ID does not
exist", "Error", JOptionPane.ERROR_MESSAGE);

  } else {

    JOptionPane.showMessageDialog(myframe, "Please enter a valid teacher ID",
"Error", JOptionPane.ERROR_MESSAGE);

  }


      JOptionPane.showMessageDialog(myframe, "Successfully Set");

    }

  });
```

```
removeButton = new JButton("Remove");
removeButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {



String teacherIdInput = TeacherIdTxt.getText();

if (!teacherIdInput.isEmpty()) {
    int teacherId = Integer.parseInt(teacherIdInput);



    for (Teacher teacher : teachers) {
        if (teacher.getTeacher_id() == teacherId) {
            if (teacher instanceof Tutor) {

                Tutor tutor = (Tutor) teacher;

                tutor.removeTutor();



                JOptionPane.showMessageDialog(myframe, "Tutor removed
successfully", "Success", JOptionPane.INFORMATION_MESSAGE);



                return;
            } else {
```

```
                    JOptionPane.showMessageDialog(myframe, "Teacher with this ID is not
a Tutor", "Error", JOptionPane.ERROR_MESSAGE);
                }
            }
        }



        JOptionPane.showMessageDialog(myframe, "Teacher with this ID does not
exist", "Error", JOptionPane.ERROR_MESSAGE);
    }else {


        JOptionPane.showMessageDialog(myframe, "Please enter a valid teacher ID",
"Error", JOptionPane.ERROR_MESSAGE);
    }
  }





    });




    mypanel.add(TeacherIdTxt, constraints); mypanel.add(TeacherIdLabel);
     mypanel.add(TeacherIdTxt);
     mypanel.add(TMLabel);
     mypanel.add(TeacherNameLabel);
    mypanel.add(TeacherNameTxt);
    mypanel.add(AddressLabel);
```

```
mypanel.add(AddressTxt);
mypanel.add(WorkingTypeLabel);
mypanel.add(WorkingTypeTxt);
mypanel.add(EmploymentStatusLabel);
mypanel.add(EmploymentStatusTxt);
 mypanel.add(WorkingHoursLabel);
mypanel.add(WorkingHoursTxt);
 mypanel.add( DepartmentLabel);
mypanel.add( DepartmentTxt);
 mypanel.add(GradedScoreLabel);
mypanel.add(GradedScoreTxt);
  mypanel.add(YearOfExperienceLabel);
mypanel.add(YearOfExperienceTxt);
 mypanel.add( SalaryLabel);
mypanel.add( SalaryTxt);
 mypanel.add(SpecilizationLabel);
mypanel.add(SpecilizationTxt);
mypanel.add(AcademicQualificationLabel);
mypanel.add(AcademicQualificationTxt);
mypanel.add(PerformanceIndexLabel);
mypanel.add(PerformanceIndexTxt);
mypanel.add(DisplayButton);
mypanel.add(addTheLecturerButton);
mypanel.add(addTheTutorButton);
mypanel.add(GradeButton);
mypanel.add(setButton);
mypanel.add(ClearButton);
mypanel.add(removeButton);
```

```
mypanel.setBackground(new java.awt.Color(173, 216, 230));

constraints.gridx = 1;
 constraints.gridy = 0;
mypanel.add(TMLabel, constraints);

constraints.gridx = 0;
constraints.gridy = 1;
mypanel.add(TeacherIdLabel, constraints);

constraints.gridx = 1;
mypanel.add(TeacherIdTxt, constraints);

constraints.gridx = 0;
constraints.gridy = 2;
mypanel.add(TeacherNameLabel, constraints);

constraints.gridx = 1;
mypanel.add(TeacherNameTxt, constraints);

constraints.gridx = 0;
constraints.gridy = 3;
mypanel.add(AddressLabel,constraints);

constraints.gridx = 1;
mypanel.add(AddressTxt, constraints);
```

```
constraints.gridx = 0;
constraints.gridy = 4;
mypanel.add(WorkingTypeLabel,constraints);

constraints.gridx = 1;
mypanel.add(WorkingTypeTxt, constraints);

constraints.gridx = 0;
constraints.gridy = 5;
mypanel.add(EmploymentStatusLabel,constraints);

constraints.gridx = 1;
mypanel.add(EmploymentStatusTxt, constraints);

constraints.gridx = 0;
constraints.gridy = 6;
mypanel.add(WorkingHoursLabel,constraints);

constraints.gridx = 1;
mypanel.add(WorkingHoursTxt, constraints);

constraints.gridx = 0;
constraints.gridy = 7;
mypanel.add(DepartmentLabel,constraints);

constraints.gridx = 1;
mypanel.add(DepartmentTxt, constraints);
```

```
  constraints.gridx = 0;
constraints.gridy = 8;
mypanel.add(YearOfExperienceLabel,constraints);

constraints.gridx = 1;
mypanel.add(YearOfExperienceTxt, constraints);

constraints.gridx = 0;
constraints.gridy = 9;
mypanel.add(GradedScoreLabel,constraints);

constraints.gridx = 1;
mypanel.add(GradedScoreTxt, constraints);

constraints.gridx = 0;
constraints.gridy = 10;
mypanel.add(SalaryLabel,constraints);

constraints.gridx = 1;
mypanel.add(SalaryTxt, constraints);

constraints.gridx = 0;
constraints.gridy = 11;
mypanel.add(SpecilizationLabel,constraints);

constraints.gridx = 1;
mypanel.add(SpecilizationTxt, constraints);
```

```
constraints.gridx = 0;
constraints.gridy = 12;
mypanel.add(AcademicQualificationLabel,constraints);

constraints.gridx = 1;
mypanel.add(AcademicQualificationTxt, constraints);

constraints.gridx = 0;
constraints.gridy = 13;
mypanel.add(PerformanceIndexLabel,constraints);

constraints.gridx = 1;
mypanel.add(PerformanceIndexTxt, constraints);

constraints.gridx = 2;
constraints.gridy = 1;
constraints.gridwidth = 2;
constraints.gridheight = 1;
constraints.anchor = GridBagConstraints.CENTER;
mypanel.add(addTheLecturerButton, constraints);

  constraints.gridx = 2;
constraints.gridy = 3;
constraints.gridwidth = 3;
constraints.gridheight = 1;
constraints.anchor = GridBagConstraints.CENTER;
mypanel.add(addTheTutorButton, constraints);
```

```
constraints.gridx = 2;
constraints.gridy = 5;
constraints.gridwidth = 3;
constraints.gridheight = 1;
constraints.anchor=GridBagConstraints.CENTER;
mypanel.add(setButton,constraints);

constraints.gridx = 2;
constraints.gridy = 7 ;
constraints.gridwidth = 3;
constraints.gridheight = 1;
constraints.anchor=GridBagConstraints.CENTER;
mypanel.add(ClearButton,constraints);

 constraints.gridx = 2;
constraints.gridy = 9;
constraints.gridwidth = 3;
constraints.gridheight = 1;
constraints.anchor = GridBagConstraints.CENTER;
mypanel.add(DisplayButton, constraints);

constraints.gridx = 2;
constraints.gridy = 11;
constraints.gridwidth = 3;
constraints.gridheight = 1;
constraints.anchor=GridBagConstraints.CENTER;
mypanel.add(GradeButton,constraints);
```

```
        constraints.gridx = 2;
        constraints.gridy = 13;
        constraints.gridwidth = 3;
        constraints.gridheight = 1;
        constraints.anchor=GridBagConstraints.CENTER;
        mypanel.add(removeButton,constraints);



        myframe.getContentPane().add(mypanel,BorderLayout.CENTER);
        myframe.setVisible(true);

    }

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