

ITAHARI
INTERNATIONAL
COLLEGE



LONDON
METROPOLITAN
UNIVERSITY

Module Code & Module Title:

CS4001NT Programming

Assessment Weightage & Type:

30% Individual Coursework

Year and Semester:

2023 Autumn

Student Name: Bidur Siwakoti

London Met ID: BIS0388

College ID: NP05CP4A230013.

Assignment Due Date: 10 May 2024

Word Count:9431

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

Table of Contents

1	Introduction	1
1.1	About coursework	1
1.2	Tools used in this coursework.	2
2	Class Diagram.....	3
3	Pseudocode	5
4	Method Description	17
4.1	TeacherGUI()	17
4.2	Action performed()	17
4.3	AddTutor().....	17
4.4	AddLecturer()	18
4.5	GradeAssignment().....	18
4.6	SetSalary ().....	19
4.7	RemoveTutor().....	19
4.8	DisplayInfo().....	19
4.9	Clear ()	20
4.10	FindTutor () and FindLecturer ()	20
4.11	GetTextFromFields ().....	20
4.12	IsIdUnique ()	20
5	Testing	21
5.1	Test 1	21
5.2	Test 2.....	23
5.2.1	Adding the lecturer	23
5.2.2	Adding tutor to array list.	26
5.2.3	Grading assignment by lecturer	29
5.2.4	Set Salary for Tutor	32
5.2.5	Remove tutor.....	35
5.3	Test 3.....	37
5.3.1	Adding teacher with duplicate teacher ID	37
5.3.2	Grading assignment by the lecturer which has not been added.....	38
5.3.3	Setting Salary for Lecturer	39
5.3.4	Setting salaries for tutor which has not been added to array list.	40
6	Error	41
6.1	Syntax Error	41
6.2	Logical Error	42
6.3	Runtime error	44
7	Conclusion	45
8	References.....	46
9	Appendix	47
10	Plagiarism Test	78

Table of Figure:

Figure 1: Java.....	1
Figure 2: BlueJ.	2
Figure 3: Class diagram in BlueJ.....	3
Figure 4: Class diagram Of Teacher Gui	4
Figure 5: Opening Gui from CMD	21
Figure 6: Output GUI	22
Figure 7: Filling required Fields	24
Figure 8: Confirmation.	25
Figure 9: Successfully Added Lecture.	25
Figure 10: Filling required details and conformation.	27
Figure 11: Successfully added tutor	28
Figure 12: Filling information for grading assignment	29
Figure 13: Displaying Grade information	30
Figure 14: Displaying Success message.....	31
Figure 15: Grade score displayed in terminal	31
Figure 16: Filling all the details for setting salary	32
Figure 17: Success message.	33
Figure 18: Display Salary Information.....	33
Figure 19: setting salary for same tutor	34
Figure 20: Entering tutor id for removing.	35
Figure 21: Conformation message.	36
Figure 22: Successful removal of tutor	36
Figure 23: Displaying error message while use same ID.....	37
Figure 24: Grading assignment without adding lecturer.....	38
Figure 25: Setting Salary for Lecturer.....	39
Figure 26: Setting salary without adding lecturer.....	40
Figure 27: Syntax error.	41
Figure 28: Correction of Syntax error.....	42
Figure 29: Logical Error	42
Figure 30: Correction of logical error.	43
Figure 31: Runtime error.....	44
Figure 32: Correction of runtime error.....	44

Table of Tables:

Table 1: Executing program form terminal.....	21
Table 2: To add lecture in array list.	23
Table 3:Adding tutor to array list.....	26
Table 4: Grading assignment by lecturer	29
Table 5: Set Salary for tutor.....	32
Table 6: Remove Tutor	35
Table 7: Adding Teacher with Duplicates teacher ID.....	37
Table 8: Grading assignment by the lecturer which has not been added	38
Table 9: Setting salary for Lecturer.....	39
Table 10: Setting salary for unknown tutor	40

1 Introduction

Java is a general-purpose and high-level programming language which is used for developing different applications such as desktop applications, network applications, web applications and mobile applications. It is also object-oriented programming language and software platforms that runs on billions of devices,



Figure 1: Java.

including notebook computers, mobile devices, gaming consoles, medical devices and many other. The rules and syntax of java are based on the C and C++ languages. (IBM.com, 2023). The major advantage of developing software with java is its portability. Once a code is wrote it can run in any devices. When the language was invented in 1991 by James Gosling of sun Microsystems, the primary goal was to be able to “write once, run anywhere”. (ibm.com, 2023)

1.1 About coursework

In the previous coursework three distinct classes Teacher, Lecturer and Tutor have been already designed. So, in this coursework we are required to design the Gui for the implementation of previous coursework and the GUI system will stores the details of teacher in array list. It also requires to implements functionalities like adding lecturer, adding tutor, grading assignments, setting salaries, and removing tutors. Additionally, error handling using try-catch blocks with appropriate message dialogs is also required. After the completion of programming part, we need to make the structured documentation for the coursework. Finally, this coursework provides a structured approach to manage teacher related information. Such type of system can be applicable in an educational institution, specifically in college or university, where the systematic organization of staff information is required for efficient management.

1.2 Tools used in this coursework.

- **BlueJ**

BlueJ is a user- friendly Integrated Development Environment (IDE) used for developing java programs. This software application helps to provide a more precise interface for creating projects and coding for java. The main purpose of BlueJ's development was to support object-oriented programming for beginner. Visual views of classes and coded object are supported by BlueJ. (Rouse, 2013). The inbuilt debugger that comes with BlueJ enables user to see their code, examine variables, and understand the flow of program execution. So, I use BlueJ as a code editor for my coursework.



Figure 2: BlueJ.

- **Ms-Word**

Microsoft word, a widely used word processing software developed by Microsoft Corporation, offers a comprehensive set of tools for creating, editing, formatting, and sharing documents. With its rich formatting options, spell checking and collaboration tools words provides more flexibility and efficiency needed to produce quality documents. Its friendly user interface and rich features make it easy to use. So, I also use Ms-word for making documentation of this coursework.

2 Class Diagram

A class diagram, a fundamental aspect of Unified Modelling Language (UML), offers a graphical depiction of the interconnections and interdependencies among classes within an object-oriented system. It serves as a blueprint for understanding the structure of a software application by illustrating the relationships, attributes, and methods associated with each class. Through boxes representing classes and lines denoting associations, class diagrams facilitate the visualization and design of complex software systems, aiding developers in comprehending the architecture and organization of their codebase. (By Tech Target, 2019)

The class diagram for GUI:

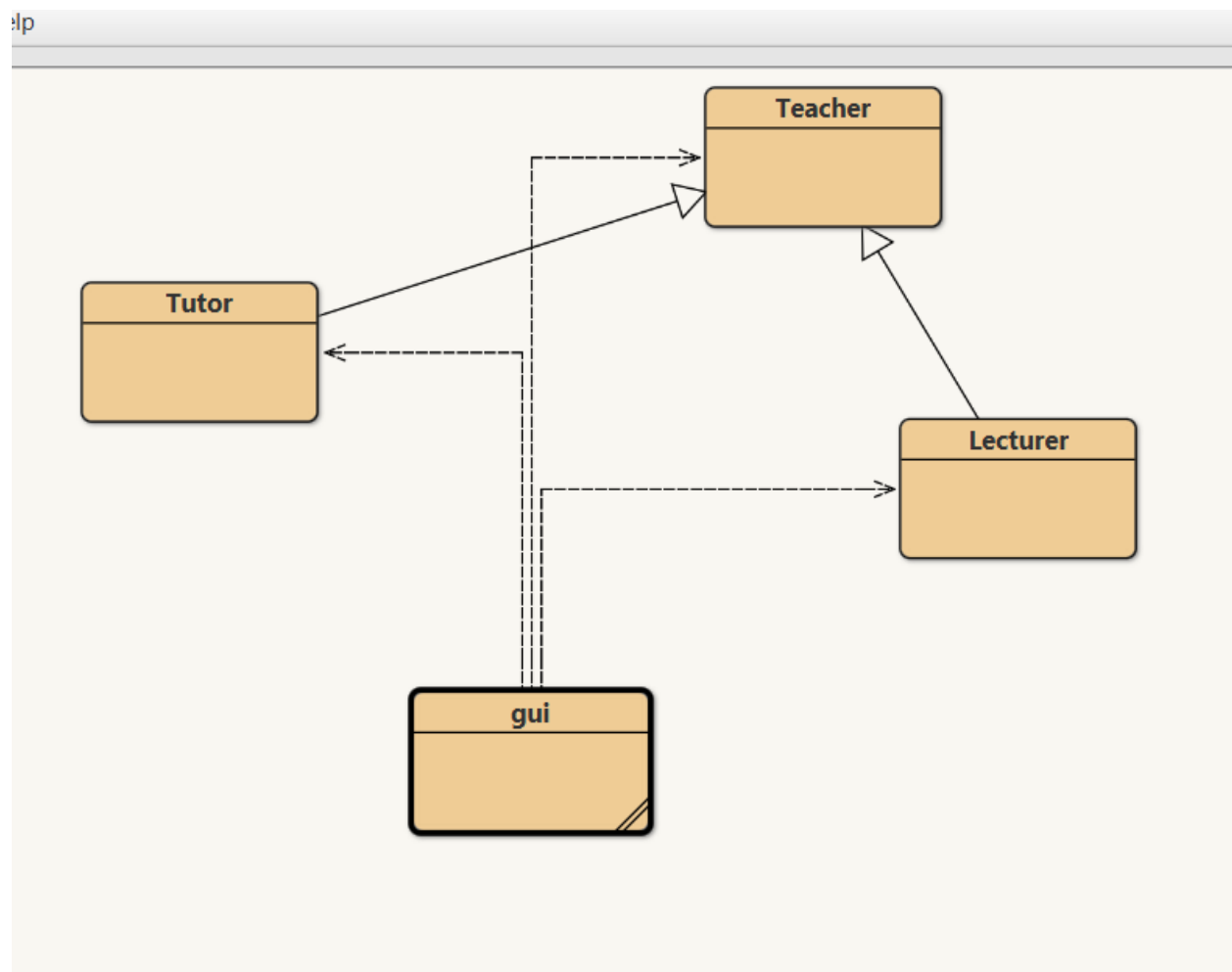


Figure 3: Class diagram in BlueJ

TeacherGUI
<pre> - teacherArrayList: ArrayList<Teacher> = new ArrayList<>() - frame: JFrame - panel: JPanel - pannelLecturer: JPanel - tutorPannel: JPanel - btnPannel: JPanel - teacherIDL: JLabel - teacherNameL: JLabel - addressL: JLabel - workingTypeL: JLabel - employmentStatusL: JLabel - workingHourL: JLabel - departmentL: JLabel - yearOfExperienceL: JLabel - gradeScoreL: JLabel - hasGradedL: JLabel - salaryL: JLabel - specializationL: JLabel - academicQualificationL: JLabel - performanceIndexL: JLabel - isCertifiedL: JLabel - salaryF: JTextField - specializationF: JTextField - academicQualificationF: JTextField - performanceIndexF: JTextField - isCertifiedF: JTextField - teacherIDF: JTextField - teacherNameF: JTextField - addressF: JTextField - workingTypeF: JTextField - employmentStatusF: JTextField - workingHourF: JTextField - departmentF: JTextField - yearOfExperienceF: JTextField - gradeScoreF: JTextField - hasGradedF: JTextField - addLecturerBtn: JButton - gradeAssignmentBtn: JButton - setSalaryBtn: JButton - removeTutorBtn: JButton - clearBtn: JButton - displayBtn: JButton - addtutorbtn: JButton myFont: Font = new Font("Impact", Font.ITALIC, 40) + «create» TeacherGUI () + addTutor () + addLecturer () + gradeAssignment () + setSalary () + displayInfo () + clear () + removeTutor () - getText (F : JTextField): String - isIdUnique (teacherId : int): boolean - getTextFromTF (field : JTextField): String - findLecturer (teacherId : int): Lecturer + getTeacherById (id : int): Teacher - findTutor (teacherId : int): Tutor + main (args : String) </pre>

Figure 4: Class diagram Of Teacher Gui

3 Pseudocode

Pseudocode is an informal way of programming description that does not require any strict programming language syntax or underlying technological considerations. It is used for creating an outline or a rough draft of a program. It summarizes a program's flow. Pseudocode is used for ensuring that programmer understands a software project's requirements and craft code accordingly. (The Economic Times, n.d.)

The pseudocode for Teacher GUI class

CREATE a parent class TeacherGui

DO

DECLARE teacherArrayList as ArrayList of Teacher

DECLARE frame as private JFrame

DECLARE pannel, pannelLecturer, tutorPannel, btnpanel as private JPanel

DECLARE teacherIDL, teacherNameL, addressL as private JLabel

DECLARE workingTypeL, employmentStatusL, workingHoursL as private JLabel

DECLARE departmentL, yearOfExperiencesL, gradeScoreL, salaryL, specializationL, academicQualificationL, performanceIndexL as private JLabel

DECLARE teacherIDF, teacherNameF, addressF, workingTypeF, employmentStatusF, workingHoursF, department, yearOfExperiencesF, gradeScoreF, salaryF, specializationF, academicQualificationF, performanceIndexF as private JTextFields

DECLARE myFont as new Font

CREATE Teachergui class

DO

INITIALIZE frame as a new frame with title

SET frame size

SET frame font to myFont

SET Default close operation to Exit on close

SET frame layout to null

SET frame location to centre

INITIALIZE panel as a new JPanel

SET panel bounds to (20, 20, 400, 300)

SET panel background color to gray

SET panel layout to null

CREATE titledBorder using BorderLayout.createTitledBorder with title "Teacher Class" and title color blue

SET panel border to titledBorder

SET panel font to myFont

INITIALIZE teacherIDL as a new JLabel with text "Teacher ID: "

SET teacherIDL bounds to (65, 30, 100, 25)

INITIALIZE teacherNameL as a new JLabel with text "Teacher Name: "

SET teacherNameL bounds to (46, 70, 100, 25)

INITIALIZE addressL as a new JLabel with text "Address: "

SET addressL bounds to (80, 110, 100, 25)

INITIALIZE workingTypeL as a new JLabel with text "Working Type: "

SET workingTypeL bounds to (48, 150, 100, 25)

INITIALIZE employmentStatusL as a new JLabel with text "Employment Status: "

SET employmentStatusL bounds to (20, 190, 130, 25)

INITIALIZE workingHourL as a new JLabel with text "Working Hours: "

SET workingHourL bounds to (45, 230, 100, 25)

INITIALIZE teacherIDF as a new JTextField

SET teacherIDF bounds to (150, 30, 100, 25)

INITIALIZE teacherNameF as a new JTextField

SET teacherNameF bounds to (150, 70, 100, 25)

INITIALIZE addressF as a new JTextField

SET addressF bounds to (150, 110, 100, 25)

INITIALIZE workingTypeF as a new JTextField

SET workingTypeF bounds to (150, 150, 100, 25)

INITIALIZE employmentStatusF as a new JTextField

SET employmentStatusF bounds to (150, 190, 100, 25)

INITIALIZE workingHourF as a new JTextField

SET workingHourF bounds to (150, 230, 100, 25)

INITIALIZE pannelLecturer as a new JPanel

SET pannelLecturer bounds to (450, 20, 400, 300)

SET pannelLecturer background color to gray

SET pannelLecturer layout to null

CREATE titledBorder1 using BorderFactory.createTitledBorder with title "Lecturer Class" and title color blue

SET pannelLecturer border to titledBorder1

INITIALIZE departmentL as a new JLabel with text "Department: "

SET departmentL bounds to (65, 30, 100, 25)

INITIALIZE yearOfExperienceL as a new JLabel with text "Year of Experience: "

SET yearOfExperienceL bounds to (20, 65, 130, 25)

INITIALIZE gradeScoreL as a new JLabel with text "Graded score: "

SET gradeScoreL bounds to (53, 105, 100, 25)

INITIALIZE addLecturerBtn as a new JButton with text "Add Lecture"

SET addLecturerBtn bounds to (50, 180, 120, 35)

SET addLecturerBtn background color to HSB color (98, 105, 223)

INITIALIZE gradeAssignmentBtn as a new JButton with text "Graded Assignment".

SET gradeAssignmentBtn bounds to (200, 180, 180, 35)

SET gradeAssignmentBtn background color to HSB color (98, 105, 223)

INITIALIZE tutorPannel as a new JPanel

SET tutorPannel bounds to (20, 350, 400, 300)

SET tutorPannel background color to gray

SET tutorPannel layout to null

CREATE titledBorder2 using BorderFactory.createTitledBorder with title "Tutor Class" and title color blue

SET tutorPannel border to titledBorder2

INITIALIZE salaryL as a new JLabel with text "Salary: "

SET salaryL bounds to (105, 30, 100, 25)

INITIALIZE specializationL as a new JLabel with text "Specializations: "

SET specializationL bounds to (53, 70, 100, 25)

INITIALIZE academicQualificationL as a new JLabel with text "Academic Qualification: "

SET academicQualificationL bounds to (13, 110, 135, 25)

INITIALIZE performanceIndexL as a new JLabel with text "Performance Index: "

SET performanceIndexL bounds to (35, 150, 130, 25)

INITIALIZE salaryF as a new JTextField

SET salaryF bounds to (155, 30, 130, 25)

INITIALIZE specializationF as a new JTextField

SET specializationF bounds to (155, 70, 130, 25)

INITIALIZE academicQualificationF as a new JTextField

SET academicQualificationF bounds to (155, 110, 130, 25)

INITIALIZE performanceIndexF as a new JTextField

SET performanceIndexF bounds to (155, 150, 130, 25)

INITIALIZE addtutorbtn as a new JButton with text "Add Tutor"

SET addtutorbtn bounds to (30, 210, 100, 30)

SET addtutorbtn background color to HSB color (98, 105, 223)

INITIALIZE setSalaryBtn as a new JButton with text "Set Salary"

SET setSalaryBtn bounds to (140, 210, 100, 30)

SET setSalaryBtn background color to HSB color (98, 105, 223)

INITIALIZE removeTutorBtn as a new JButton with text "Remove Tutor"

SET removeTutorBtn bounds to (250, 210, 100, 30)

SET removeTutorBtn background color to HSB color (98, 105, 223)

INITIALIZE btnPannel as a new JPanel

SET btnPannel bounds to (450, 400, 300, 100)

SET btnPannel background color to gray

SET btnPannel layout to null

CREATE titledBorder3 using BorderLayout.createTitledBorder with title "Buttons" and title color blue

SET btnPannel border to titledBorder3

INITIALIZE displayBtn as a new JButton with text "Display"

SET displayBtn bounds to (20, 30, 120, 25)

SET displayBtn background color to HSB color (98, 105, 223)

INITIALIZE clearBtn as a new JButton with text "Clear"

SET clearBtn bounds to (160, 30, 120, 25)

SET clearBtn background color to HSB color (98, 105, 223)

ADD components to respective panels

ADD panels to the frame

SET frame visible to true

ADD action listener to addLecturerBtn

DO

CALL addLecturer() method

END DO

ADD action listener to gradeAssignmentBtn

DO

CALL gradeAssignment() method

END DO

ADD action listener to addtutorbtn

DO

CALL addTutor() method

END DO

ADD action listener to setSalaryBtn

DO

CALL setSalary() method

END DO

ADD action listener to removeTutorBtn

DO

CALL removeTutor() method

```
END DO
ADD action listener to clearBtn
DO
    CALL clear() method
END DO
ADD action listener to displayBtn
DO
    CALL displayInfo() method
END DO
DEFINE main method with parameters String[] args
DO
    INITIALIZE Teachergui as a new instance of Teachergui class
END DO
DEFINE addTutor method
DO
    TRY
DO
    SET teacherId to the text retrieved from teacherIDF field
    SET teacherName to the text retrieved from teacherNameF field
    SET address to the text retrieved from addressF field
    SET workingType to the text retrieved from workingTypeF field
    SET employmentStatus to the text retrieved from employmentStatusF field
    SET workingHours to the text retrieved from workingHourF field
    SET salary to the text retrieved from salaryF field
    SET specialization to the text retrieved from specializationF field
    SET academicQualification to the text retrieved from
academicQualificationF field
    SET performanceIndex to the text retrieved from performanceIndexF field
```

PARSE teacherId to integer and store it in teacherID

PARSE workingHours to integer and store it in workingHour

PARSE salary to double and store it in salaries

PARSE performanceIndex to integer and store it in performanceInd

IF salaries is less than 0 **AND** workingHour is less than 20

THEN

DISPLAY error message "Salary cannot be smaller than 1 and \n Working hours must be greater than 20" using **JOptionPane.ERROR_MESSAGE**

RETURN

END IF

IF performanceInd is less than 5 **OR** performanceInd is greater than 10

THEN

DISPLAY error message "Performance index must be between 5 and 10" using **JOptionPane.ERROR_MESSAGE**

RETURN

END IF

SET confirm to the result of **JOptionPane.showConfirmDialog** with parameters tutorPannel, "Please confirm the entered information", "Confirmation", and **JOptionPane.INFORMATION_MESSAGE**

IF teacherID is not unique

THEN

DISPLAY error message "Teacher ID already exists. Please choose a different ID." using **JOptionPane.ERROR_MESSAGE**

RETURN

END IF

IF confirm is not equal to JOptionPane.YES_OPTION

THEN

RETURN

END IF

TRY

INstantiate a new Tutor object named tutor with parameters
teacherID, teacherName, address, workingType, employmentStatus,
workingHour, salaries, specialization, academicQualification,
performanceInd

ADD tutor to teacherArrayList

DISPLAY success message "Tutor added successfully!" using
JOptionPane.INFORMATION_MESSAGE

CATCH NumberFormatException

DISPLAY error message "Invalid input. Please check the inputs
and try again." using **JOptionPane.ERROR_MESSAGE**

CATCH IllegalArgumentException

DISPLAY error message "Please fill in all required fields." using
JOptionPane.ERROR_MESSAGE

END TRY

DEFINE gradeAssignment method

INITIALIZES all the variables

CREATE ActionListener gradeBtnListener

DO

IMPLEMENT actionPerformed method

DO

TRY

GET teacher ID from teacherIdAF text field

GET graded score from gradedScoresAF text field

GET department from departmentAF text field

GET years of experience from yearOfExpAF text field

Parse teacher ID and graded score to integers

if lecturer is not found

PRINT error message

ELSE

Display error message indicating lecturer not found

END IF

CATCH NumberFormatException

Display error message indicating invalid input for ID, graded score, or years of experience

CATCH IllegalArgumentException

Display error message indicating missing required fields

END TRY-CATCH

END DO

END DO

DEFINE setsalary method

INITIALIZES all the variables

CREATE ActionListener setSalaryBtnListener

DO

IMPLEMENT actionPerformed method

DO

TRY

Get all the input

IF new salary < 0

THEN

Display error message indicating invalid salary (negative value)

RETURN

END IF

IF new performance index < 0 OR new performance index > 10

THEN

Display error message indicating invalid performance index

RETURN

END IF

IF new performance index > 5 AND tutor.getWorkingHours() > 20

THEN

Call set salary method

Display success message indicating salary for tutor successfully u

END IF

Display message indicating teacher ID, new salary, and new

RETURN

END IF

END

END DO

HANDEL Exception

CREATE ActionListener clearBtnListener

DO

IMPLEMENT actionPerformed method

DO

Clear text fields teacherIdSF, newSalarySF, and newPerformanceSF

END DO

END DO

CREATE METHOD displayInfo

DO

INITIALIZE StringBuilder displayText

IF teacherArrayList IS EMPTY

THEN

Display message indicating no teachers to display in array list

RETURN

END IF

FOR EACH teacher IN teacherArrayList

DO

IF teacher IS AN INSTANCE OF Lecturer

THEN

DISPLAY lecturer information in frame.

PRINT lecturer information in console

ELSE IF teacher IS AN INSTANCE OF Tutor

DISPLAY tutor information in frame.

PRINT tutor information in frame.

END FOR

END DO

CREATE method clear.

DO

SET the text fields to empty

END DO

4 Method Description

A method is a block of code within a class that performs a specific task or operations. Methods are used to encapsulate functionality, making it reusable and modular. They can accept input parameters, perform actions based on those parameters, and optionally return a value. Methods can be invoked by other parts of the program to execute the code they contain. In object-oriented programming, methods are associated with object and are often used to manipulate the object's state or behaviour. The method description for the TeacherGUI class is briefly described below:

4.1 TeacherGUI()

TeacherGUI is a constructor method. It initializes all necessary components such as frames, panels, labels, text fields for creating a GUI. The components are positioned by setting bounds and then added to their respective panel. This method also includes action listener to perform specific action when the user interacts with the buttons. At last, the constructor adds the panels and components to the frame and sets its visibility to true, making the GUI visible and interactive to the user. Overall, the teacherGui() constructor method serves as the foundation for the GUI of the teacher management system.

4.2 Action performed()

The actionPerformed() method is an implementation of the action listener interface which is used to interact with buttons used in graphical user interface(GUI). This method is invoked automatically when a button is clicked. Some of the functionalities of this method include identifying the source, performing action like adding lecturer and tutor, grading assignments etc.

4.3 AddTutor()

This method is responsible for adding new tutor to the array list of teacher management system. This method begins by retrieving input data entered by the user from the text fields in the GUI which includes attributes likes teacher ID, name, address, working type, employment status, working hours, salary, specialization etc. After that input validation is done to ensure that the entered data is in correct format and meets the necessary requirements. After retrieving and validation of input data a confirmation dialog is displayed to user to proceed it for adding new tutor. Finally, the new 'Tutor' object is

instantiated with provided data when the user confirms the entered information and stored the information in teacher array list. This method also includes error handling to catch and handle exceptions that may occur during the execution of program.

4.4 AddLecturer()

The method is same as addtutor() method but it is responsible for adding new lecturer to the array list of teacher management system. The method begins by retrieving input data entered by the user from the text fields in the GUI that includes attributes likes teacher ID, name, address, working type, employment status, working hours, department, year of experience etc. Similar to the addTutor() method input validation is performed to ensure that the entered data is of the correct format and meets necessary requirements and creates new 'Lecturer' object and store in teacher array list.

.

4.5 GradeAssignment()

The GradeAssignment() method is responsible for grading assignments for lecturers in the teacher management system. The method begins with creating new frame to provide required information to grade assignments. With in the frame, labels and text fields like teacher ID, graded scores, department, and year of experiences are initialized with grade and clear buttons. Action listeners are attached to the grading button and clear button. While clicking on grade button, the method retrieves the input values from the text fields. After than it validates the input data, checking that it contains valid data. Once the input data is validated, the method attempts to find the lecturer in the system using provided teacher ID. If the lecturer is found, the "**GradeAssignment()**" method of the lecturer class is invoked passing the required parameters form the text fields. Error handling is implemented to catch and handle exceptions that may occur during the grading process.

4.6 SetSalary ()

The SetSalary () method handles the process of setting new salary for tutors in the teacher management system. similar to GradeAssignment () method, the setSalary () method starts by creating a new frame to provide dedicated graphical interface for setting salaries for tutor. Labels and text fields are added to the frame to allow the user to enter the necessary information for assigning new salary. After than button is added to the frame to perform actions such as setting the salary and clearing the input fields. While clicking the 'set salary' button the method retrieves the input values from the text fields and it also validates the user's input, ensuring the entered data are correct. Once the validation is completed the method attempts to find the tutor of respective ID. If the tutor is found new salary will set for the tutor if the condition meets. Finally, the method also includes error handling to catch and handle exception that may occur during the salary setting process. If the invalid input is detected or the tutor is not found it will display appropriate error message to guide user.

4.7 RemoveTutor()

The RemoveTutor() method is responsible for removing the tutor from teacher array list. This method begins with displaying the input dialog box for entering the ID of tutor they want to remove. After than input validation is done, it will check for null inputs. If the input is valid, the method proceeds to parse the input strings into integer. After than the ID is used to find the corresponding tutor object in the system by calling the findtutor() method. If the tutor is found, the method checks whether the tutor is certified or not. If the tutor is certified it display an error message and if it is not certified the method prompts the user for confirmation to remove tutor from the array list. After the successful removal, a success message is displayed to notify the user. At last, the method also includes error handling to catch and handle exceptions that may occurs during the removal process.

4.8 DisplayInfo()

The displayInfo() method is responsible for displaying information of teacher stored in the teacher array list. Before displaying any information, the method checks whether there is teacher information in array list. If not, it will display proper error messages. If the teacher

array list is not empty, the method iterates through each object stored in the list using for-each loop. After than it checks the instance of teacher and tutor and display accordingly. After appending information for teacher, the method prints the information to the console. Overall, this method provides a means to visualize the information stored in the teacher array list, ensuring that user can easily view and verify the data within the teacher management system.

4.9 Clear ()

The clear() method is responsible for clearing all the text fields in the GUI. The method first clear the text fields related to the basic information of the teacher such as ID, name, address, working type, employment status, and working hours. After than it will clear the text fields related to lecturer class which includes fields like department, year of experience grade score. At final it will clear the text fields related to tutor class which includes fields like salary, specialization, academic qualifications and performance index.

4.10 FindTutor () and FindLecturer ()

The methods findtutor () and findlecturer () are used for locating specific types of teachers either tutor or lecturer within a given list of teachers based on their unique identifier. These methods are designed to search the instances of various teacher form the array list name 'teacherArrayList'. It verifies the instance of lecturer and tutor. If no teacher with the specified ID is found the method return null value.

4.11 GetTextFromFields ()

This method is designed for extracting text input from a jtext fields components with basic validation. When invoked, the method first accesses the text entered into the provided jtext fields. If the fields is empty it throws Illegal argument exception.

4.12 IsIdUnique ()

This method is a private method that checks whether a given teacher id is unique among the existing teacher id stored in the teacher array list. It takes an integer parameter. If the teacher id parameter matches the ID of any teacher in the list, it returns false indicating that the ID is not unique.

5 Testing

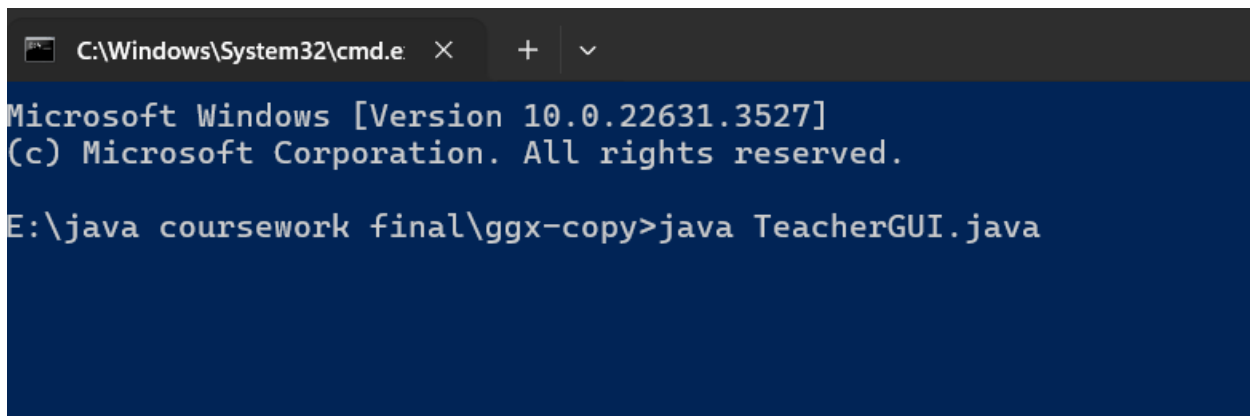
Testing is the process of making the software application or system error and bug free also to ensure that it behaves as expected and meets the given requirements. It involves running the program with various inputs to detect any defect or errors. It is a important part in software development lifecycle. Some of the importance of testing include bug detection, quality assurance, customer satisfaction etc. The testing carried out for this coursework are listed below:

5.1 Test 1

Table 1: Executing program form terminal.

Objective:	To open the GUI through command prompt.
Action performed	The folder is open in CMD, and the Teacher GUI program is executed from command prompt.
Excepted Result	The program should be compiled without any error.
Actual Result	The program was compiled without any error.
Test Result	Test successful.

The screenshots involve in this testing:



```

C:\Windows\System32\cmd.e  X  +  v
Microsoft Windows [Version 10.0.22631.3527]
(c) Microsoft Corporation. All rights reserved.

E:\java coursework final\ggx-copy>java TeacherGUI.java
  
```

Figure 5: Opening Gui from CMD

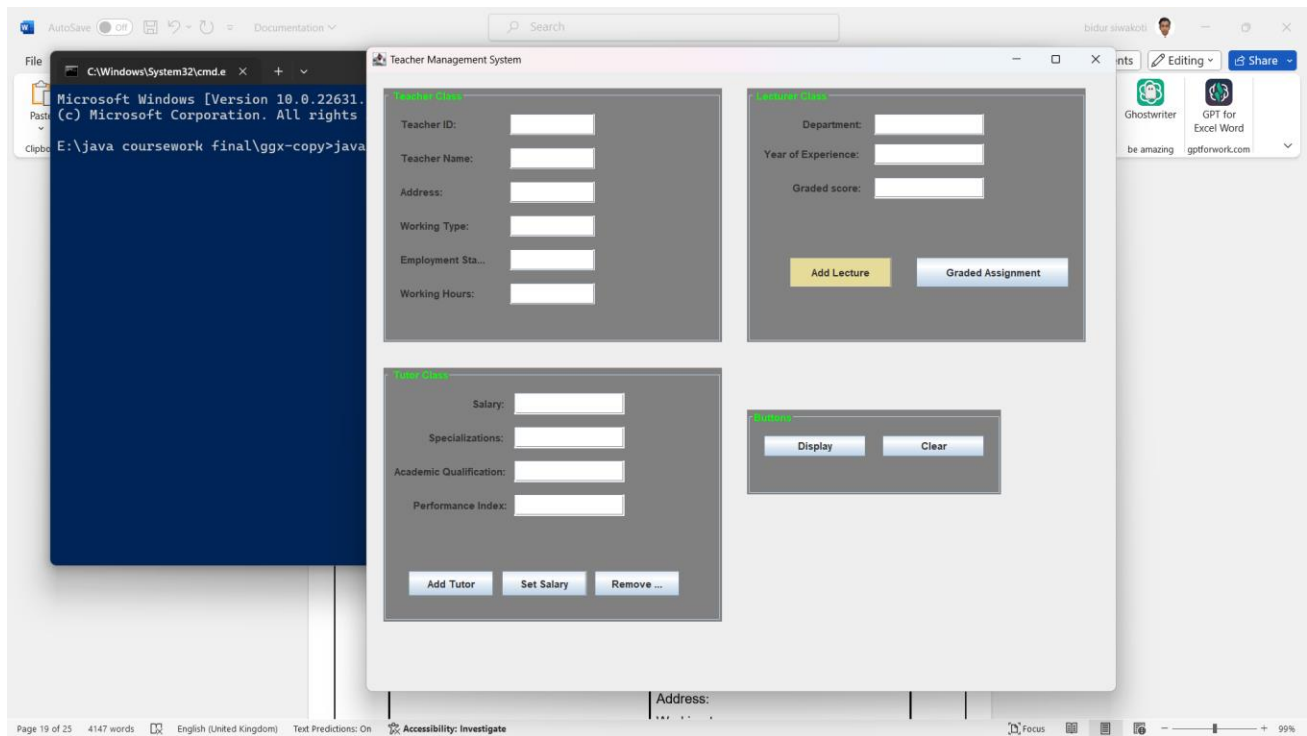


Figure 6: Output GUI

5.2 Test 2

5.2.1 Adding the lecturer

Table 2: To add lecture in array list.

Objective:	To add lecturer to array list.
Action performed	Text fields are filled with below values, Teacher Id: 12 Teacher Name: Bikram Poudel Address: Itahari Working type: Full Time Employment Status: Leave Working Hours: 23 Department: BIT Year of Experiences:6 Grade scores: 92. After that Add Lecturer button is clicked
Excepted Result	The information of lecturer should store in array list and before storing in array list it should show conformation message.
Actual Result	The information of lecturer is successfully stored in array list.
Test Result	Test successful.

The screenshot shows a Java Swing window titled "Teacher Management System" with standard window controls (minimize, maximize, close). The window contains four panels:

- Teacher Class:** A panel with six text input fields. The values entered are: Teacher ID: 12, Teacher Name: Bikram Poudel, Address: Itahari, Working Type: Full Time, Employment Status: Leave, and Working Hours: 23.
- Lecturer Class:** A panel with three text input fields. The values entered are: Department: BIT, Year of Experience: 6, and Graded score: 92. Below these fields are two buttons: "Add Lecture" and "Graded Assignment".
- Tutor Class:** A panel with five text input fields. The fields are labeled: Salary, Specializations, Academic Qualification, Performance Index, and Is certified. All fields are currently empty. Below these fields are three buttons: "Add Tutor", "Set Salary", and "Remove Tutor".
- Buttons:** A separate panel containing two buttons: "Display" and "Clear".

Figure 7: Filling required Fields

The screenshot displays the 'Teacher Management System' window. It contains three main form panels: 'Teacher Class', 'Lecturer Class', and 'Tutor Class'. The 'Teacher Class' panel has fields for Teacher ID (12), Teacher Name (Bikram Poudel), Address (Itahari), Working Type (Full Time), Employment Status (Leave), and Working Hours (23). The 'Lecturer Class' panel has fields for Department (BIT) and Year of Experience (6). A 'Confirmation' dialog box is overlaid on the 'Lecturer Class' panel, asking 'Please confirm the entered information' with 'Yes', 'No', and 'Cancel' buttons. The 'Tutor Class' panel has fields for Salary, Specializations, Academic Qualification, Performance Index, and Is certified, with 'Add Tutor', 'Set Salary', and 'Remove Tutor' buttons. A 'Buttons' panel at the bottom right has 'Display' and 'Clear' buttons.

Figure 8: Confirmation.

The screenshot displays the 'Teacher Management System' window. The 'Lecturer Class' panel now includes a 'Graded score' field with the value 92. A 'Success' dialog box is overlaid on the window, displaying an information icon and the message 'Lecturer added successfully!' with an 'OK' button. The other panels and buttons remain the same as in Figure 8.

Figure 9: Successfully Added Lecture.

5.2.2 Adding tutor to array list.

Table 3: Adding tutor to array list.

Objective:	To add tutor to array list
Action performed	Text fields are filled with below values, Teacher Id:13 Teacher Name: Nikesh Bhattra Address: Dulari Working type: Full Time Employment Status: Active Working Hours:32 Salary:454545 Specialization: Java Academic Qualification: BIT Performance Index: 6 And clicks Add Tutor button.
Excepted Result	The information of tutor should store in array list and before storing in array list it should conformation.
Actual Result	The information of tutor is successfully stored in array list.
Test Result	Test successful.

The screenshot displays the 'Teacher Management System' window, which is divided into three main sections: 'Teacher Class', 'Lecturer Class', and 'Tutor Class'. Each section contains a form with various input fields and buttons.

Teacher Class:

- Teacher ID: 13
- Teacher Name: Nimesh Bhattra
- Address: Dulari
- Working Type: Full Time
- Employment Status: Active
- Working Hours: 32

Lecturer Class:

- Department: [Empty]
- Year of Experience: [Empty]
- Graded score: [Empty]
- Buttons: Add Lecture, Graded Assignment

Tutor Class:

- Salary: 454545
- Specializations: Java
- Buttons: Add Tutor, Set Salary, Remove Tutor

A confirmation dialog box is overlaid on the Tutor Class form, displaying a green question mark icon and the text: 'Please confirm the entered information'. The dialog has three buttons: Yes, No, and Cancel.

Buttons:

- Display
- Clear

Figure 10: Filling required details and conformation.

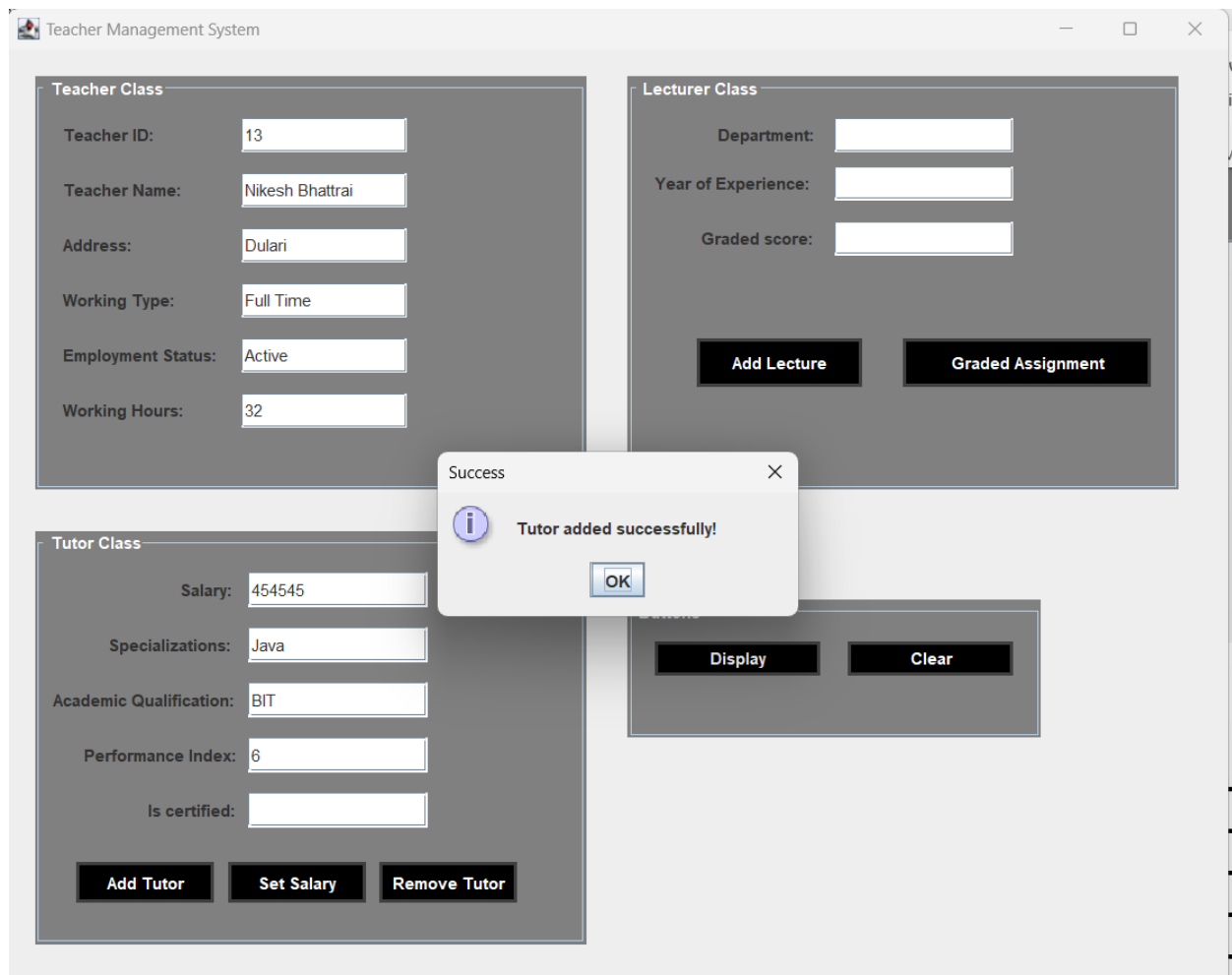


Figure 11: Successfully added tutor

5.2.3 Grading assignment by lecturer

Table 4: Grading assignment by lecturer

Objective:	To grade the assignment.
Action performed	Fill all the required details for grading assignment and click grade button.
Expected Result	Assignment should be graded and graded score should be displayed in terminal with success message.
Actual Result	Assignment is graded and success message is displayed.
Test Result	This test is successful.

The screenshots used in this test are presented below as evidence:

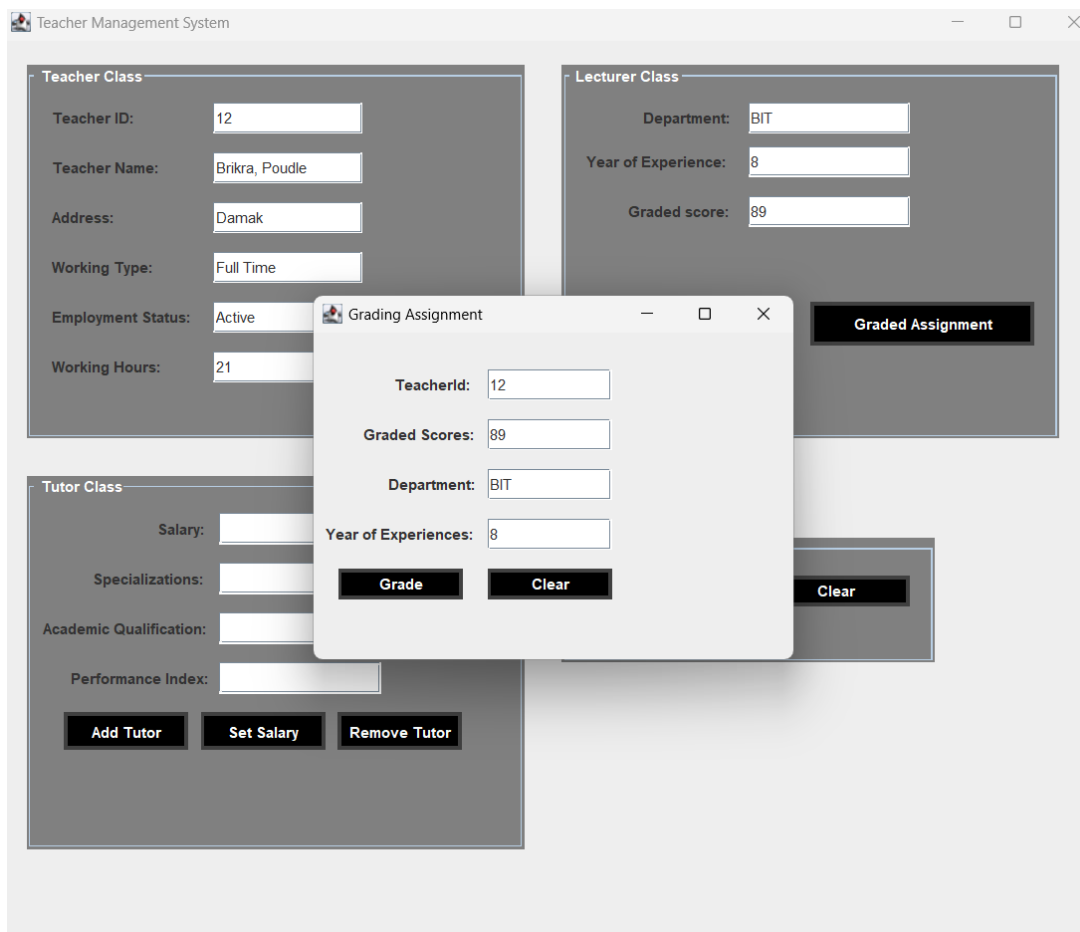


Figure 12: Filling information for grading assignment

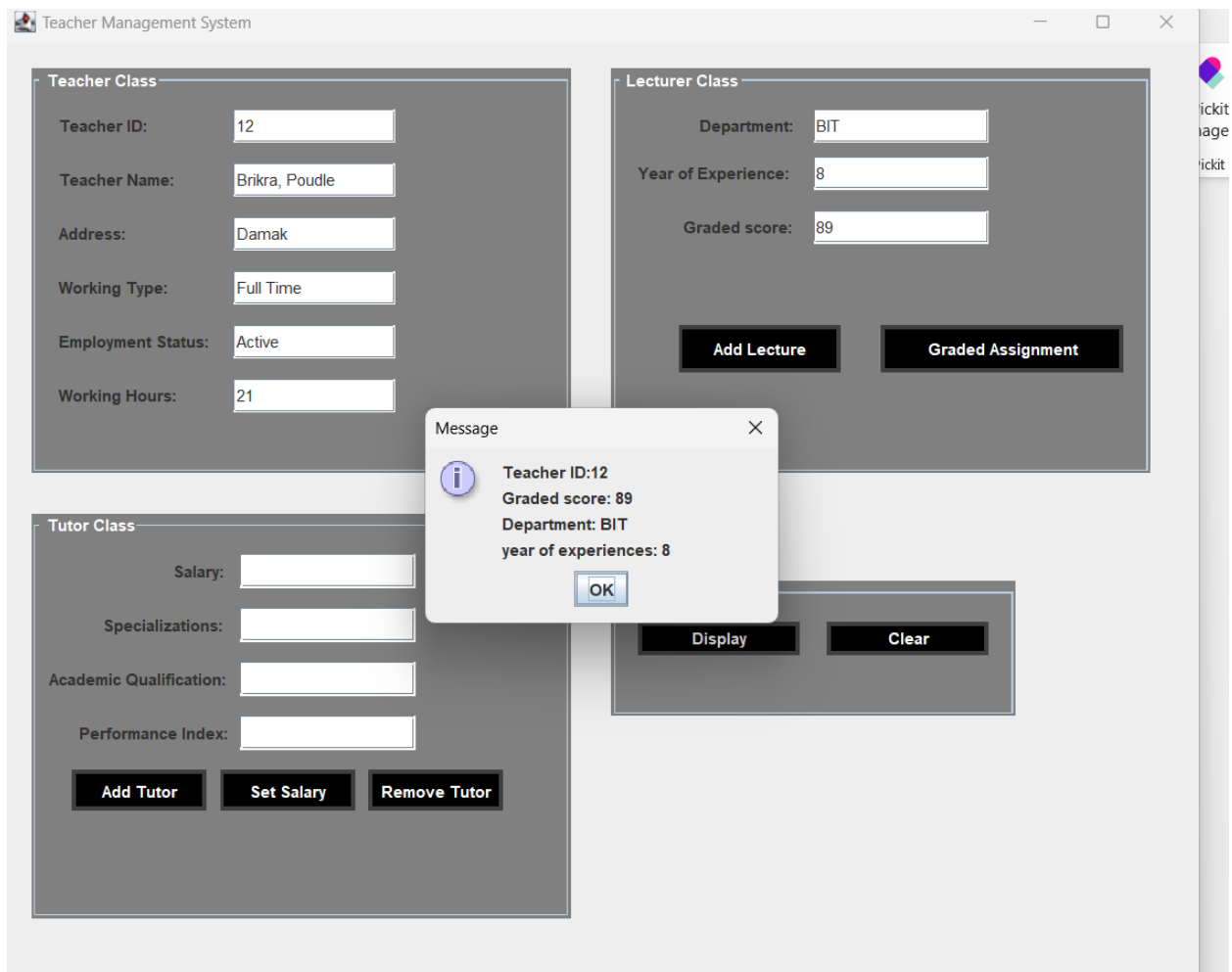


Figure 13: Displaying Grade information.

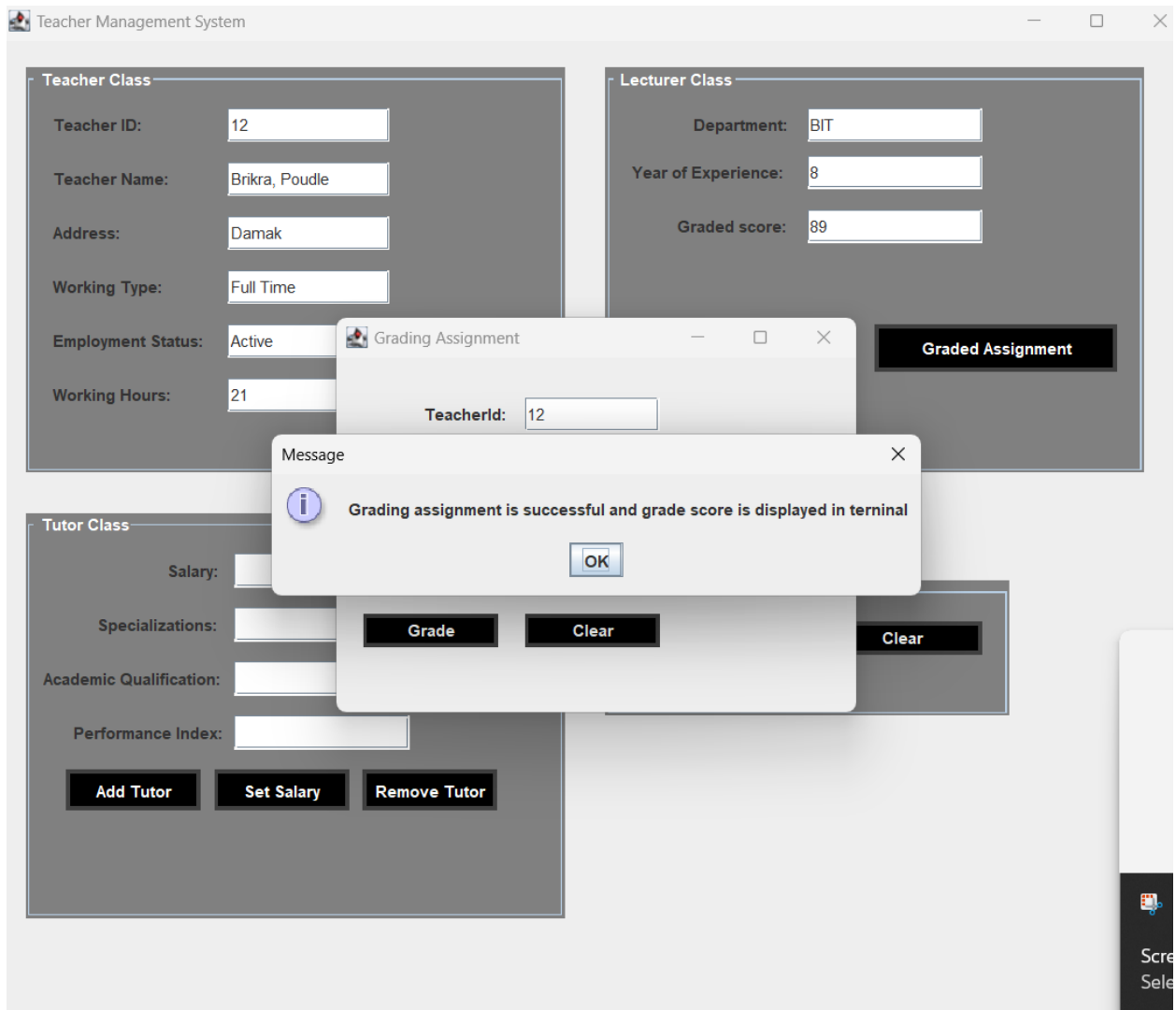


Figure 14: Displaying Success message

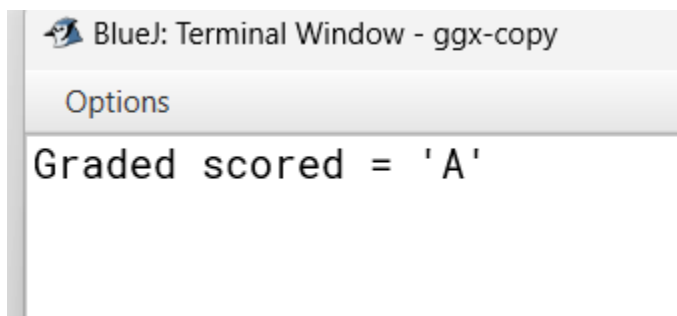


Figure 15: Grade score displayed in terminal

5.2.4 Set Salary for Tutor

Table 5: Set Salary for tutor

Objective:	To set the salary for tutor.
Action performed	Fill all the required details with valid values for setting salary and click set salary button.
Excepted Result	It should set the salary.
Actual Result	Salary for tutor is successfully updated.
Test Result	Test 5.2.4 is successful.

The screenshot involved in this test are provided below as evidence:

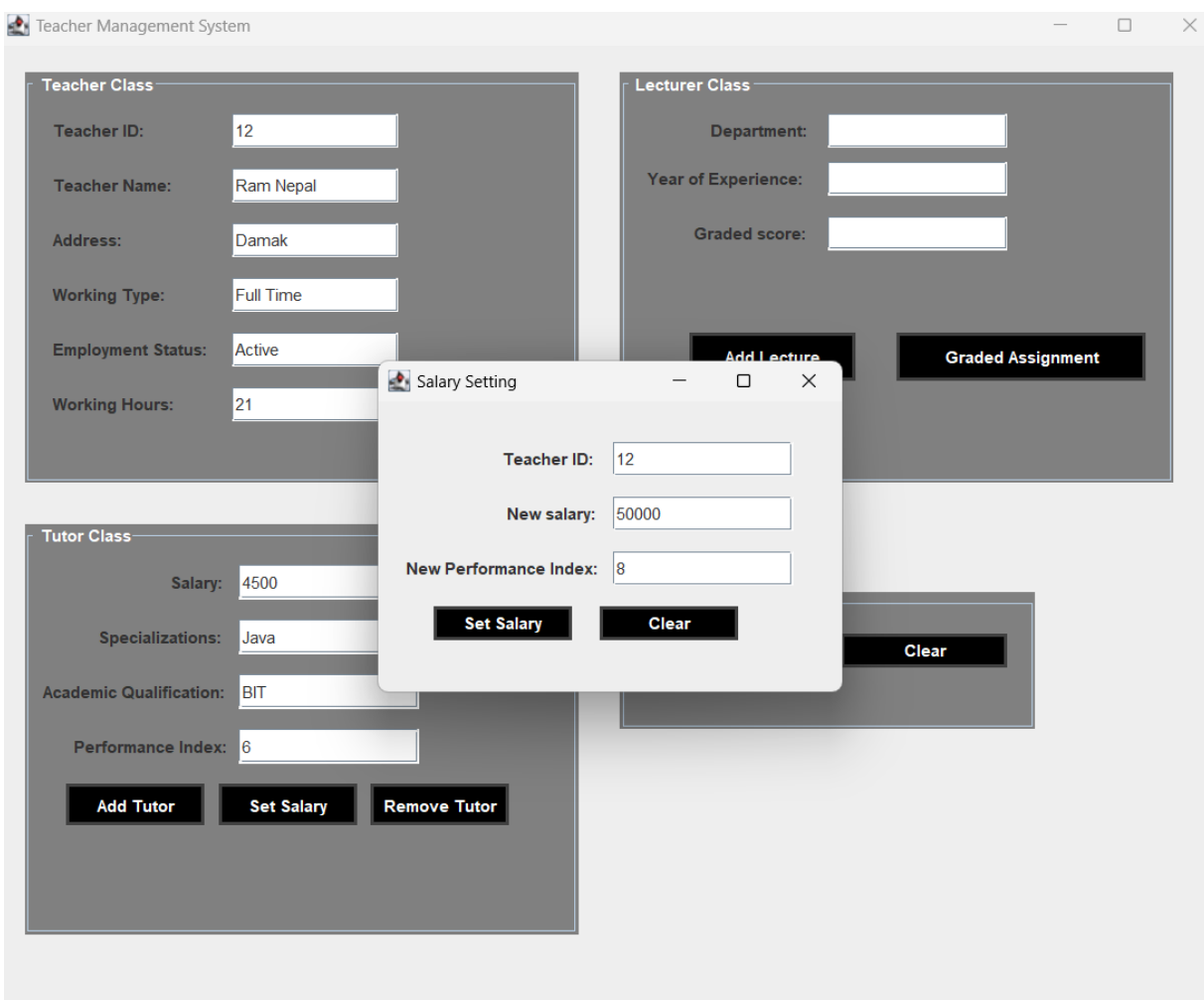


Figure 16: Filling all the details for setting salary

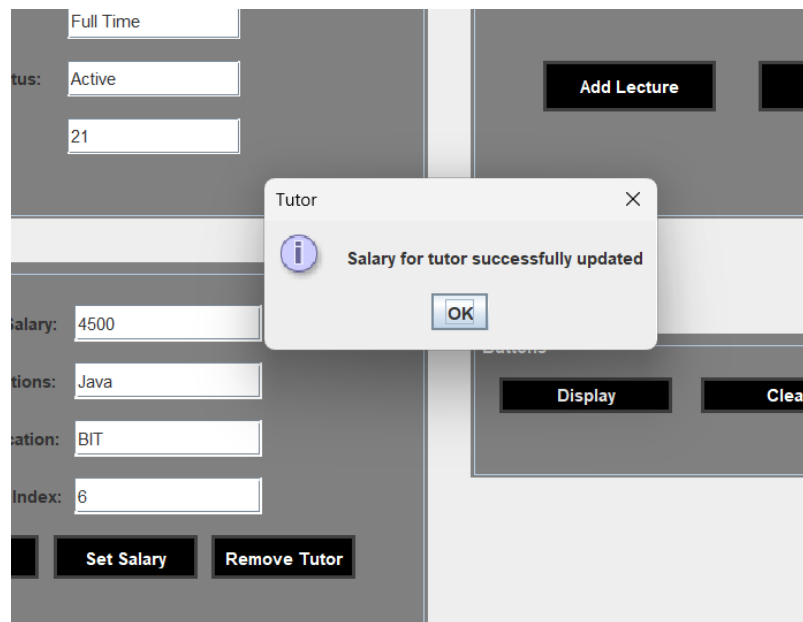


Figure 17: Success message.

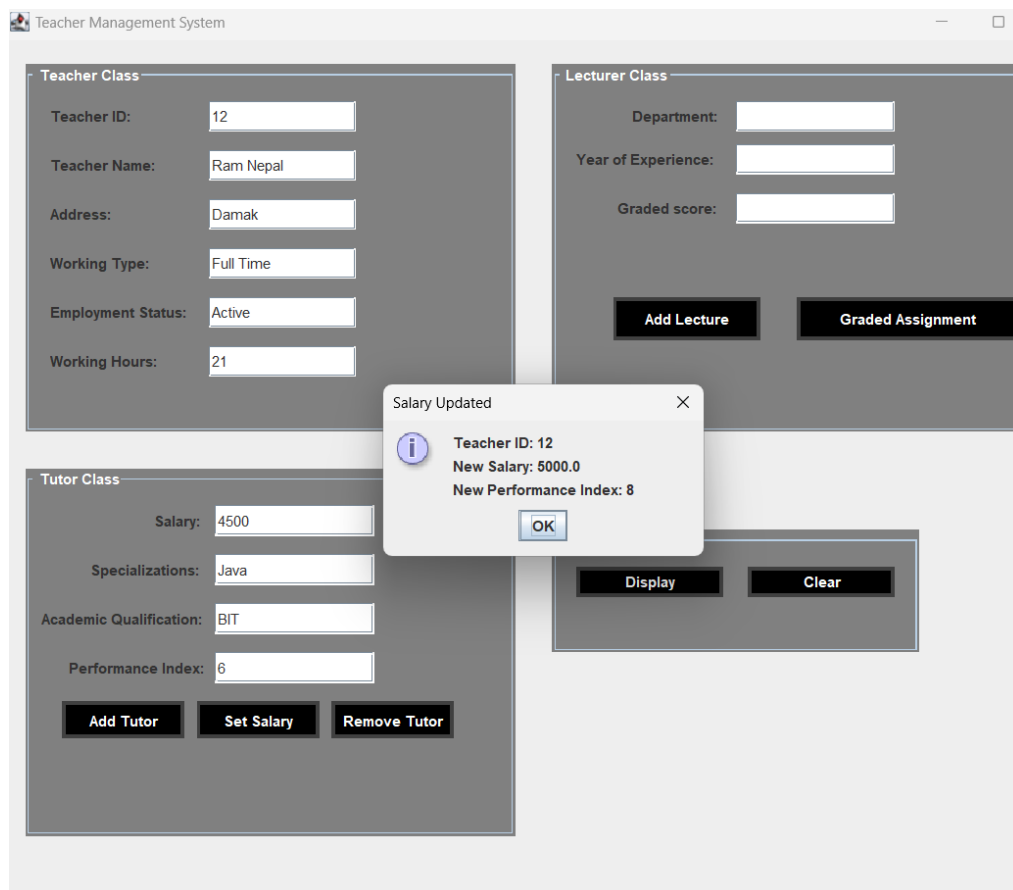


Figure 18: Display Salary Information

The image shows a Java application interface with three main panels and an error dialog box.

- Teacher Class Panel:** Contains input fields for Teacher ID (12), Teacher Name (Ram Nepal), Address (Damak), Working Type (Full Time), Employment Status (Active), and Working Hours (21).
- Lecturer Class Panel:** Contains input fields for Department, Year of Experience, and Graded score. It also has buttons for 'Add Lecture' and 'Graded Assignment'.
- Tutor Class Panel:** Contains input fields for Salary (4500), Specializations (Java), Academic Qualification (BIT), and Performance Index (6). It has buttons for 'Add Tutor', 'Set Salary', and 'Remove Tutor'.
- Error Dialog Box:** Titled 'Certified Tutor', it displays a red 'X' icon and the message 'Tutor is already certified. Salary cannot be modified'. It has an 'OK' button.

Figure 19: setting salary for same tutor

5.2.5 Remove tutor.

Table 6: Remove Tutor

Objective:	To remove tutor from array list.
Action performed	Enter the tutor id that has been stored in the array list and clicks remove button.
Excepted Result	It should remove tutor and should display success message.
Actual Result	Tutor is removed and success message is displayed.
Test Result	This test is successful.

The screenshots used in this test are presented below as evidence:

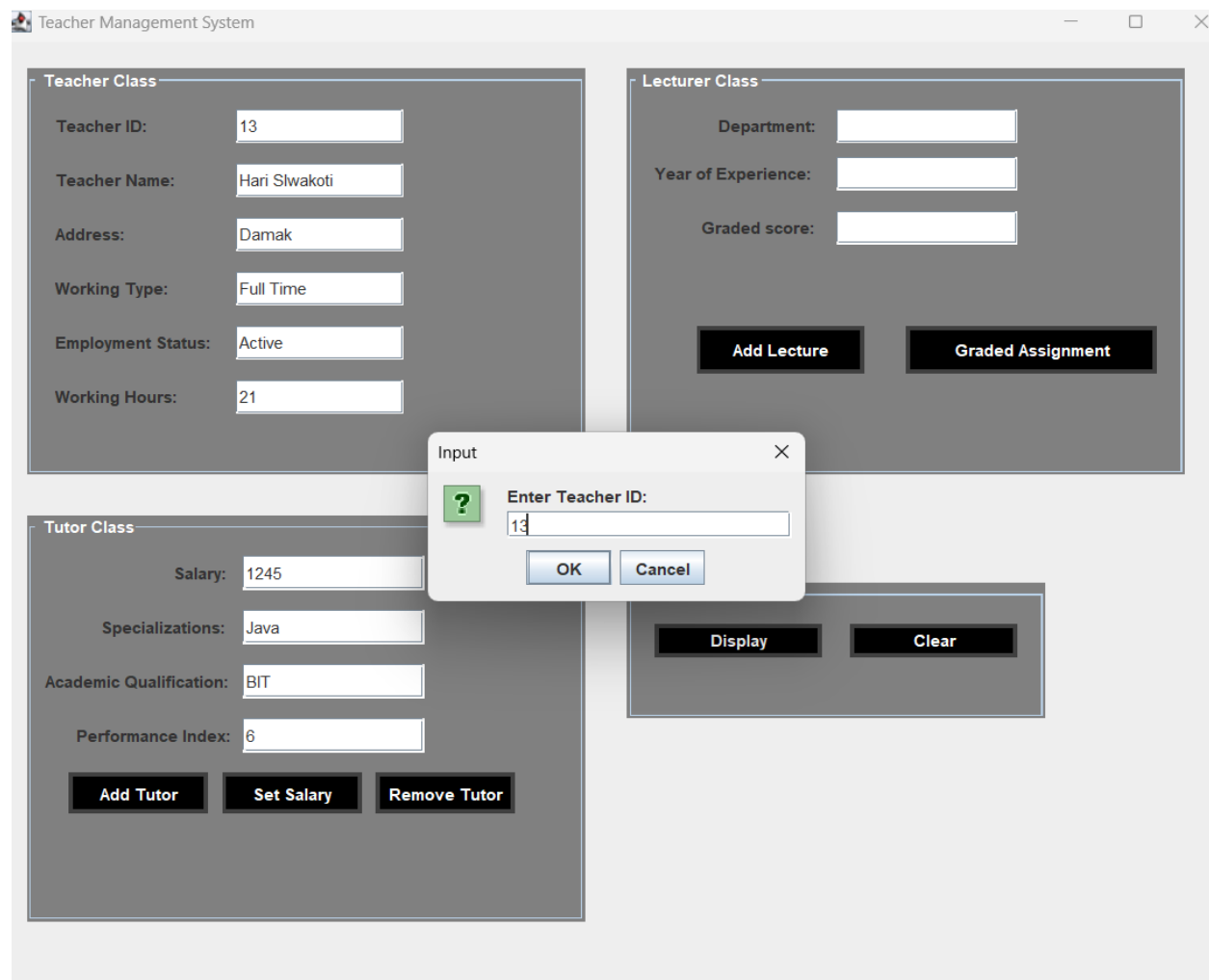


Figure 20: Entering tutor id for removing.

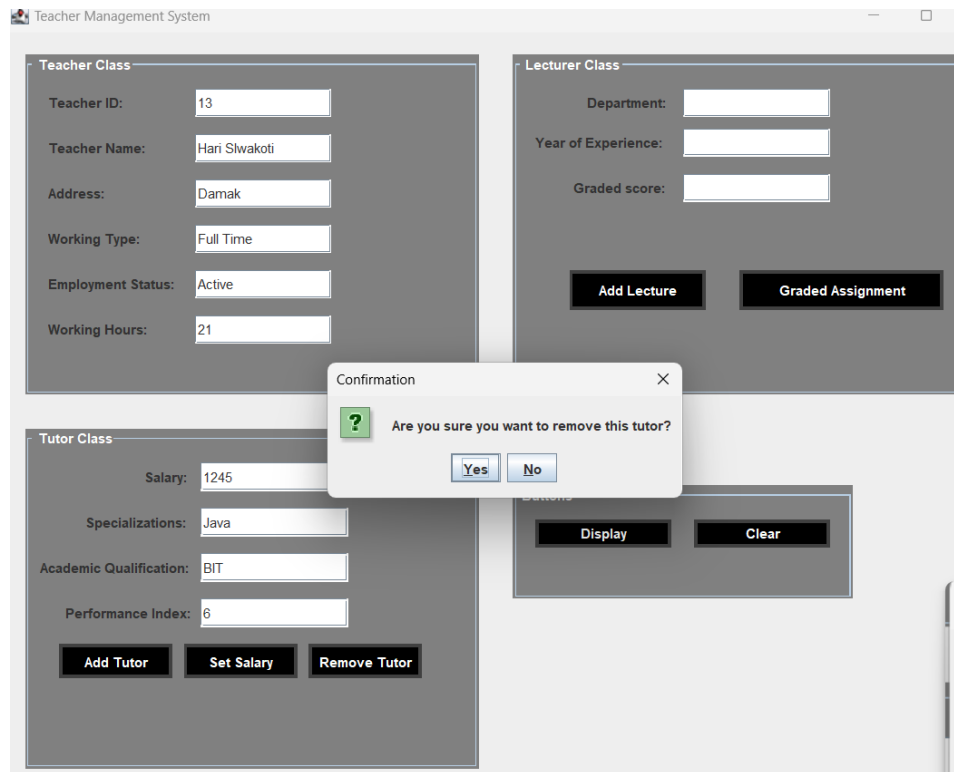


Figure 21: Conformation message.

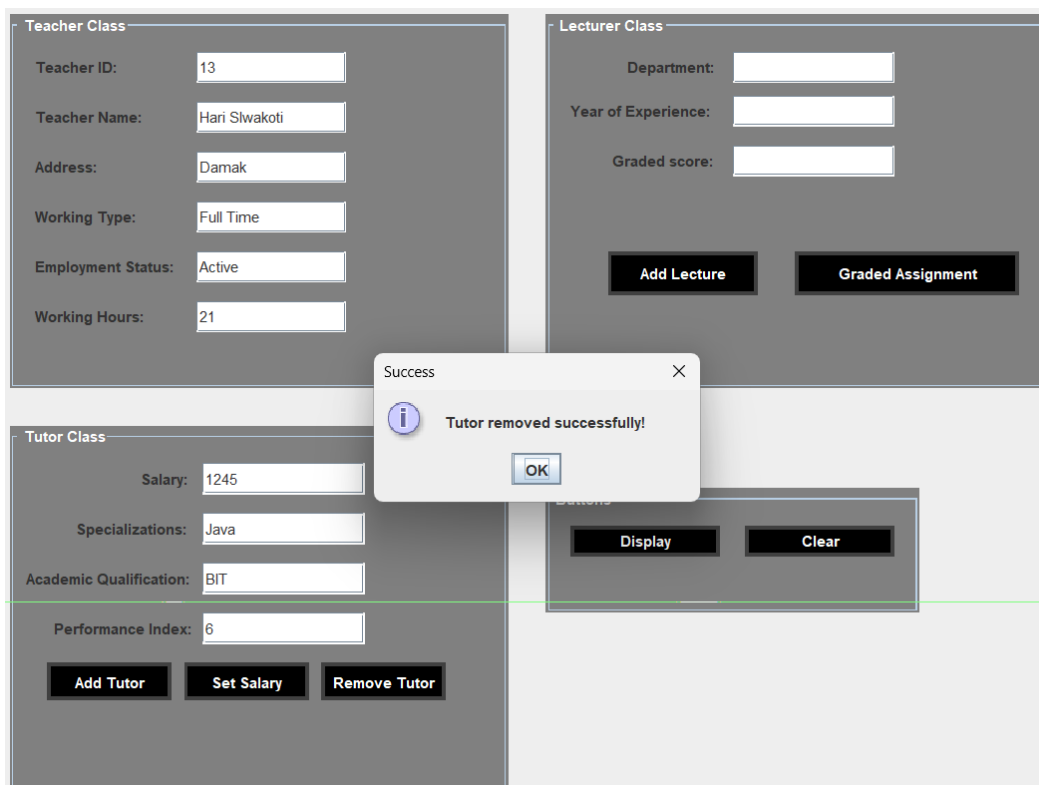


Figure 22: Successful removal of tutor

5.3 Test 3

5.3.1 Adding teacher with duplicate teacher ID

Table 7: Adding Teacher with Duplicates teacher ID

Objective:	To add teacher with duplicate teacher id.
Action performed	Entering the teacher id with duplicate id and click add button and use that same id for another teacher.
Excepted Result	While adding the teacher with duplicate id it should show error message.
Actual Result	It shows error message.
Test Result	This test is successful.

The screenshots used in this test are presented below as evidence:

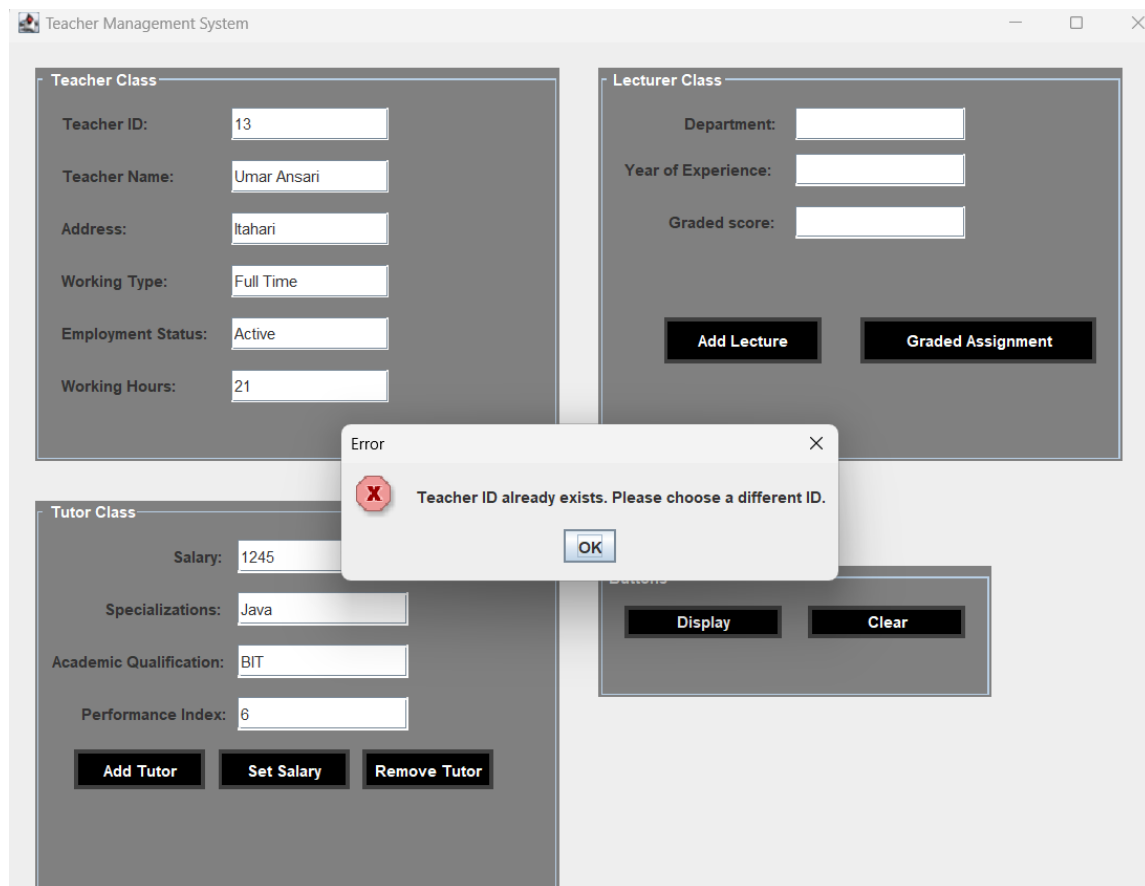


Figure 23: Displaying error message while use same ID.

5.3.2 Grading assignment by the lecturer which has not been added.

Table 8: Grading assignment by the lecturer which has not been added

Objective:	To grade the assignment by the lecturer which has not been added to array list
Action performed	Enter a valid information in text fields while grading assignment but use the teacher id that has not been added to array list and click grade button.
Excepted Result	The assignment should not graded and error message should be displayed.
Actual Result	The assignment is not graded and error message is displayed,
Test Result	This test is successful.

The screenshots used in this test are presented below as evidence:

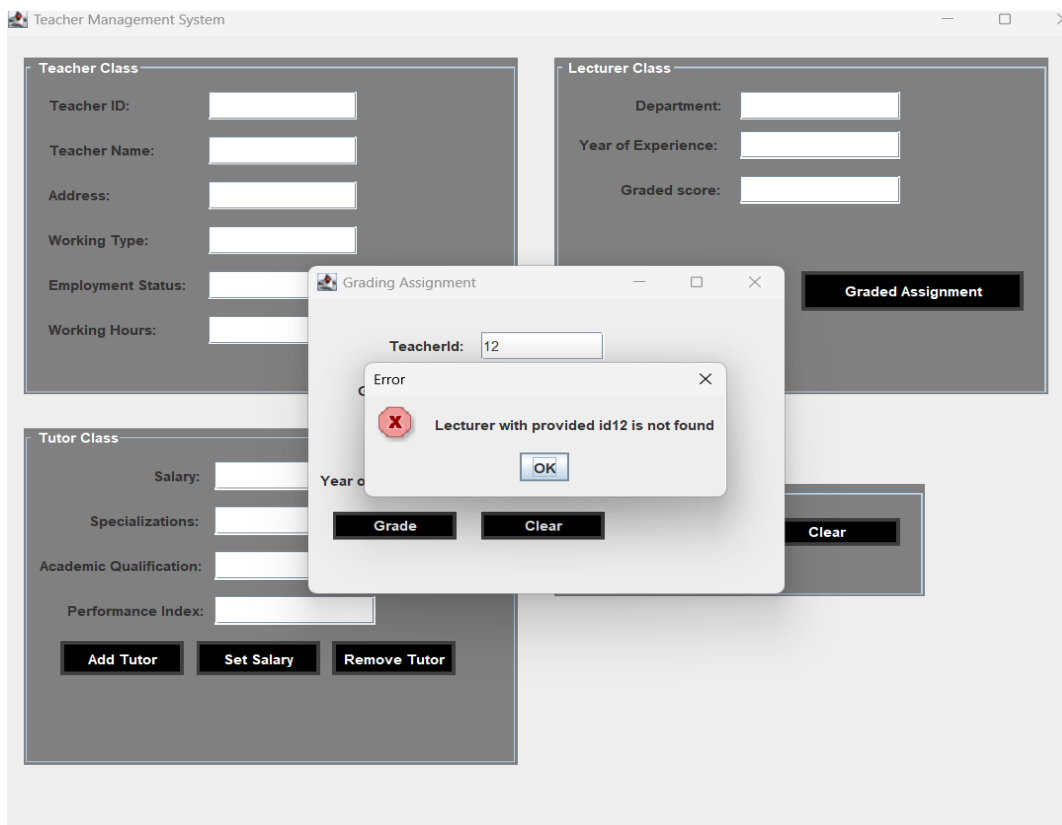


Figure 24: Grading assignment without adding lecturer.

5.3.3 Setting Salary for Lecturer

Table 9: Setting salary for Lecturer

Objective:	To set the salary for lecturer.
Action performed	While setting salary used lecturer id for tutor id and click set salary button.
Excepted Result	Salary should not updated and error message should displayed.
Actual Result	Salary is not update and error message is displayed.
Test Result	This test is successful.

The screenshots used in this test are presented below as evidence:

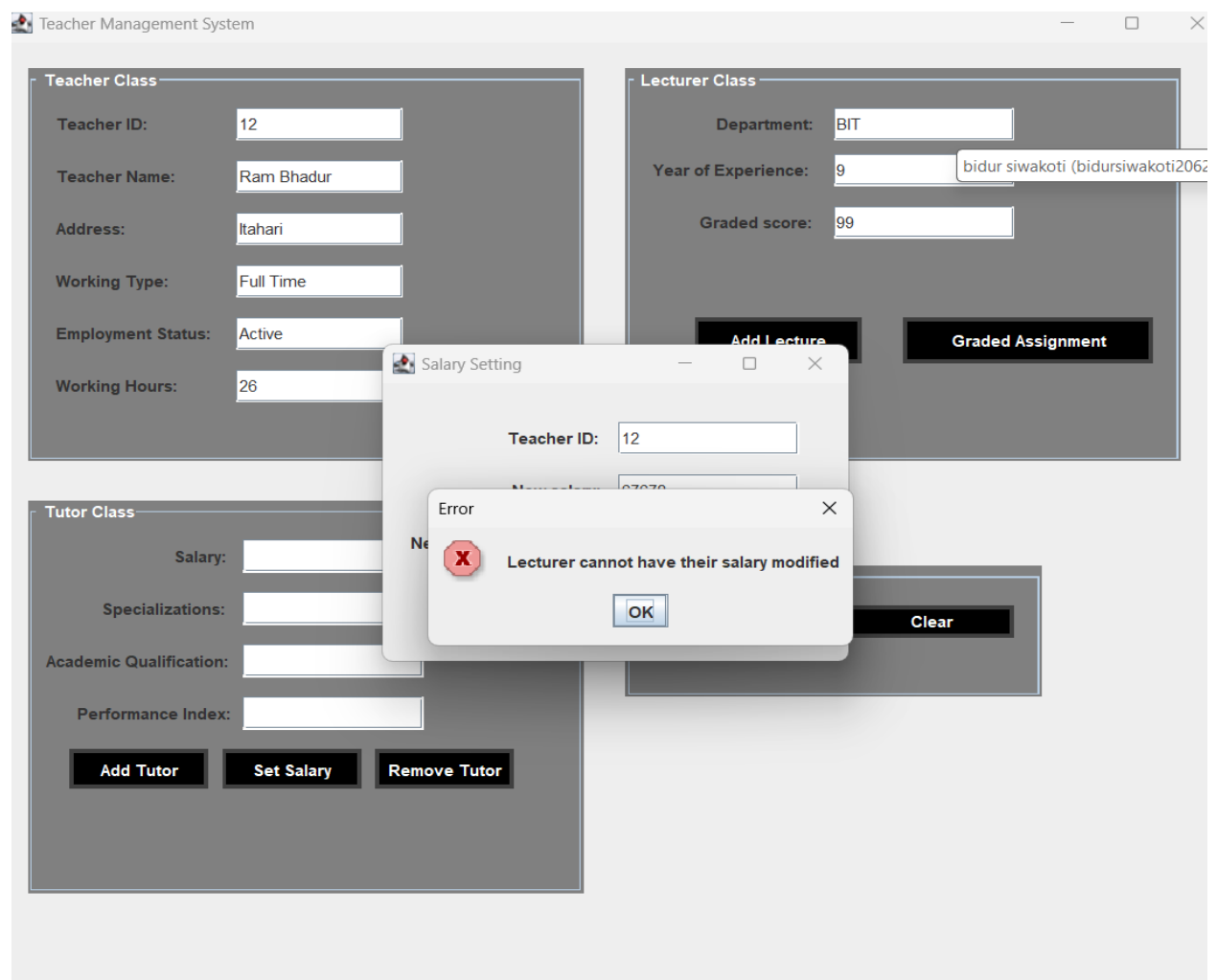


Figure 25: Setting Salary for Lecturer.

5.3.4 Setting salaries for tutor which has not been added to array list.

Table 10: Setting salary for unknown tutor

Objective:	To set the salary for tutor which has not been added to array list.
Action performed	While setting salary use tutor id that has not been added to the array list and click set salary button.
Excepted Result	The salary should not update, and error message should be displayed.
Actual Result	The salary is not updated, and error message is displayed.
Test Result	This test is successful.

The screenshots are presented below as evidence:

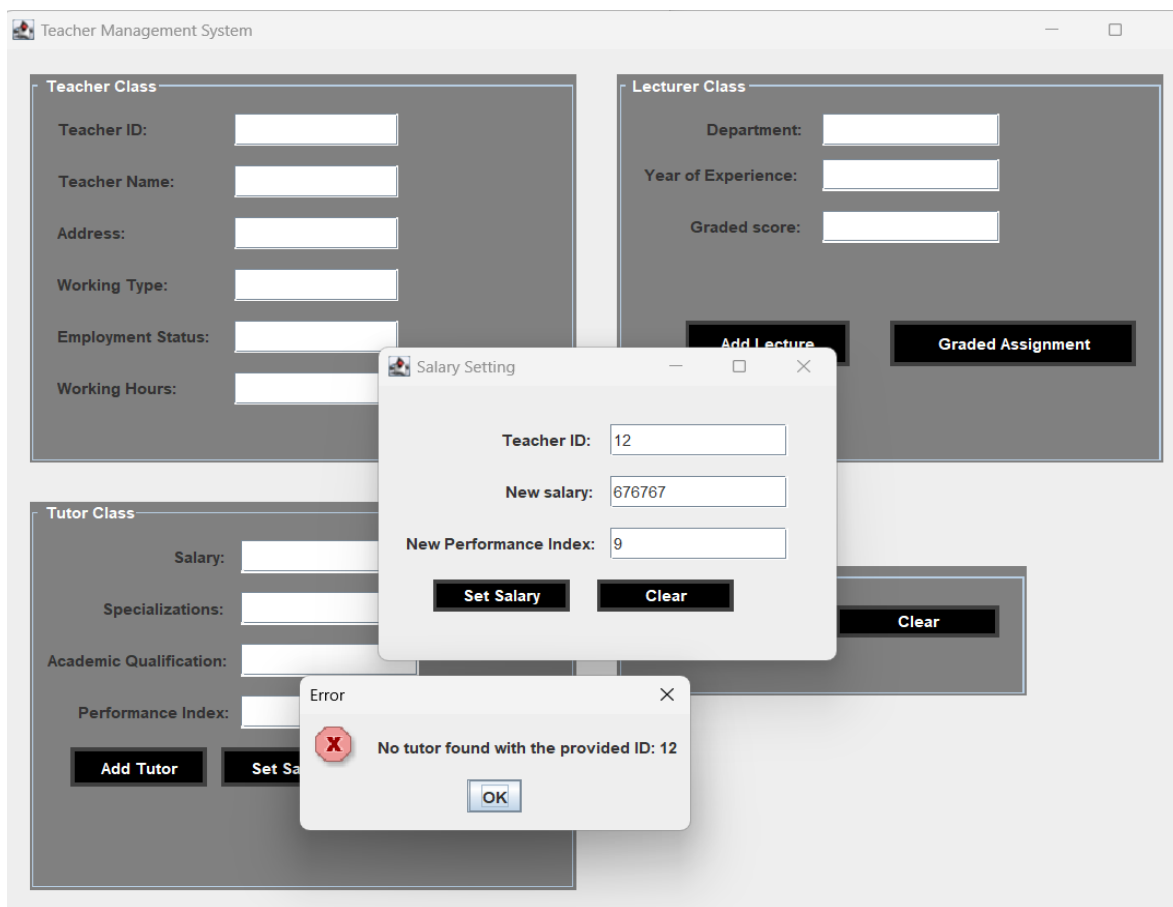


Figure 26: Setting salary without adding lecturer.

6 Error

6.1 Syntax Error

A syntax error occurs when the rules of programming languages are not followed correctly in code. This means that the code does not meet the grammar and structure of respective programming languages. Some of the common syntax errors include missing semicolons, mismatched parentheses and brackets. Misspelled keywords or identifier, improper use of operator etc. are also some of the common syntax errors. Such errors are usually detected by the compiler.

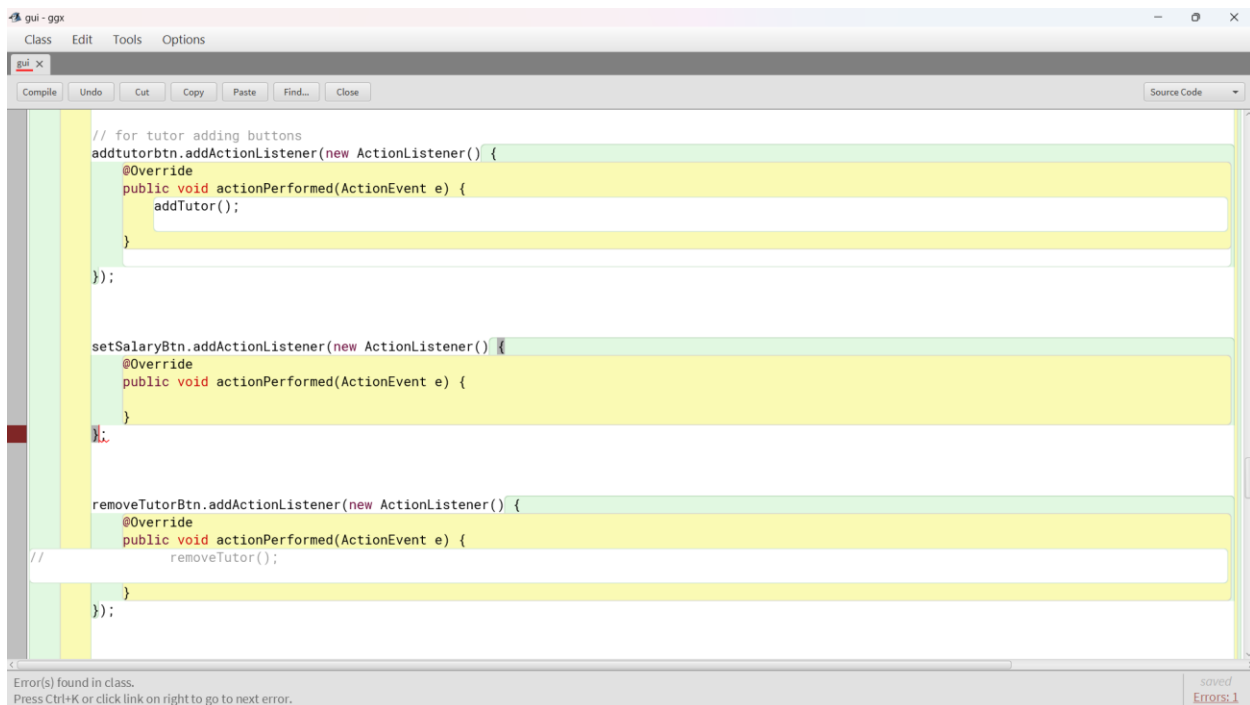


Figure 27: Syntax error.

In this code, there is no small bracket that's why the compiler shows this error. We can fix this error as:

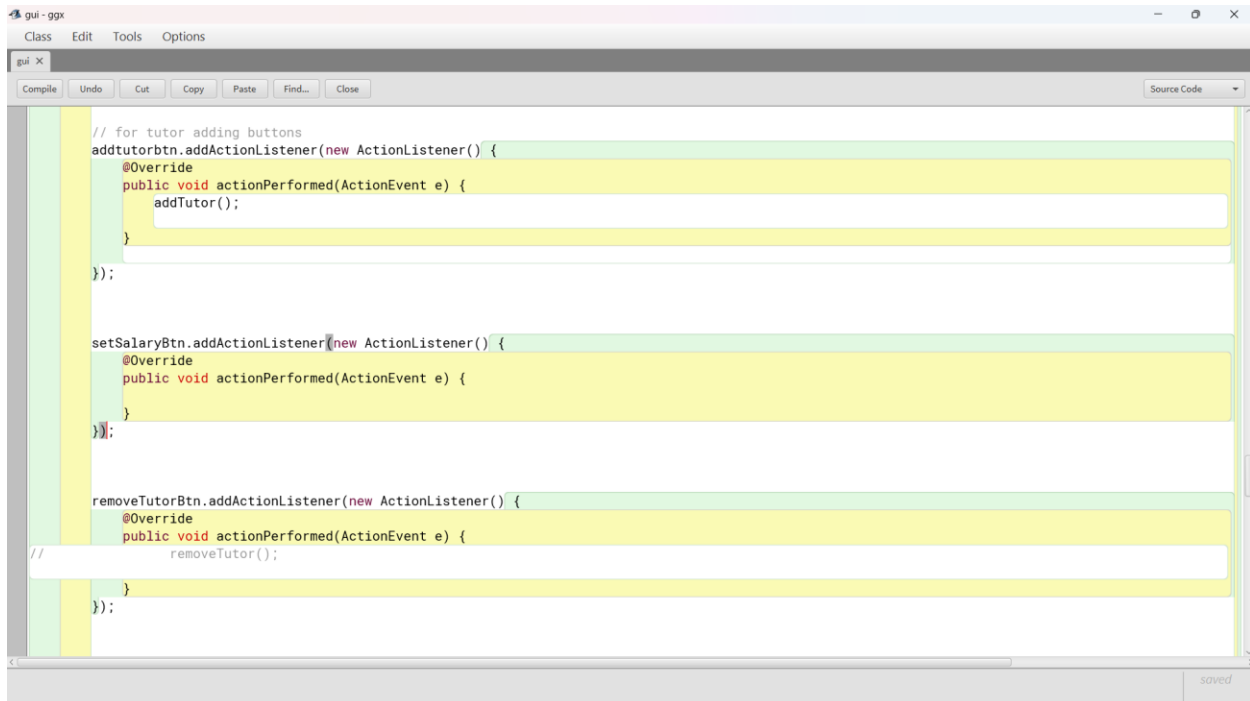


Figure 28: Correction of Syntax error.

6.2 Logical Error

Logical error are those errors that occurs when the code compiles and runs without any syntax error, but it does not produce the expected or required output due to incorrect logic implementation. Some of the logical errors are incorrect assumptions about how certain function work, incorrect use of data structures, flawed logic in conditional statements and loops. Unlike syntax errors, logical error does not cause the program to crash or prevent it from compiling, but they result in the program behaving differently. Such types of error are challenging to identify because such error is not usually detected by the compiler.

```
int id = Integer.parseInt(teacherId);
int yearOfExperience = Integer.parseInt(yearsOfExperience);
int workingHour = Integer.parseInt(workingHours);
if (yearOfExperience < 5 || workingHour < 20){
    JOptionPane.showMessageDialog(frame, "Year of experiences must be greater than 5 and \n Working hours must be gre
    return;
}
```

Figure 29: Logical Error

In above code its checking either for year of experience or working hours to meets the conditions but we need to make sure for to meet the condition. So, by using and instead of or we can fix this error.

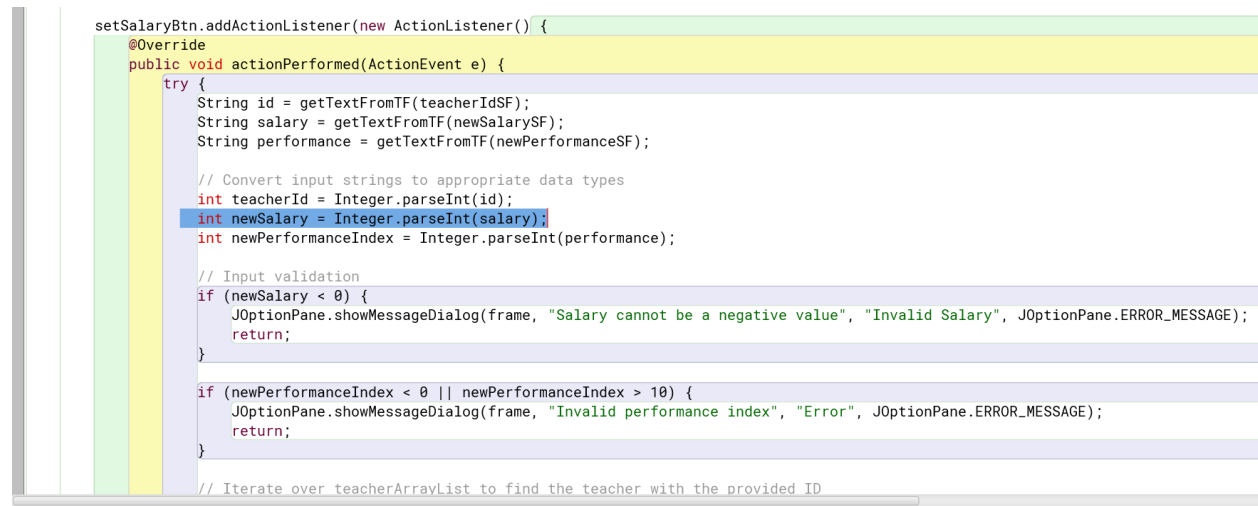
```
int id = Integer.parseInt(teacherId);
int yearOfExperience = Integer.parseInt(yearsOfExperience);
int workingHour = Integer.parseInt(workingHours);
if (yearOfExperience < 5 && workingHour < 20) {
    JOptionPane.showMessageDialog(frame, "Year of experiences must be greater than 5 and \n Working hours must
    return;
}

int confirm = JOptionPane.showConfirmDialog(pannelLecturer, "Please confirm the entered information", "Confirm
if (!isIdUnique(id)) {
    JOptionPane.showMessageDialog(frame, "Teacher ID already exists. Please choose a different ID.", "Error",
    return;
}
if (confirm != JOptionPane.YES_OPTION) {
    return;
}
Lecturer lecturer = new Lecturer(id, teacherName, address, workingType, employmentStatus, department, yearOfEx
```

Figure 30: Correction of logical error.

6.3 Runtime error

A runtime error, also known as an exception, occurs when a program is running and encounters an unexpected condition that prevents it from continuing its normal execution. Unlike syntax errors, which are detected by the compiler before the program runs, runtime errors occur during the execution of the program. Some of the common runtime errors are division by zero, null pointer exception, array index out of bounds, type casting error, file not found, etc. Such types of error can be handle using try catch blocks to catch exceptions.



```
setSalaryBtn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        try {
            String id = getTextFromTF(teacherIdSF);
            String salary = getTextFromTF(newSalarySF);
            String performance = getTextFromTF(newPerformanceSF);

            // Convert input strings to appropriate data types
            int teacherId = Integer.parseInt(id);
            int newSalary = Integer.parseInt(salary);
            int newPerformanceIndex = Integer.parseInt(performance);

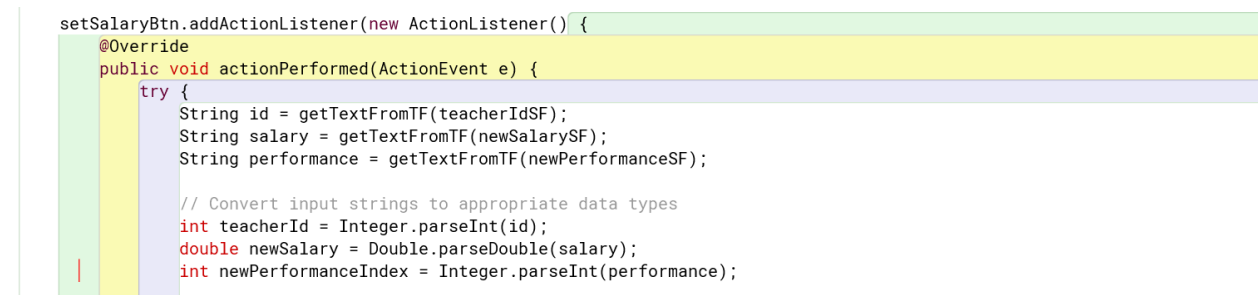
            // Input validation
            if (newSalary < 0) {
                JOptionPane.showMessageDialog(frame, "Salary cannot be a negative value", "Invalid Salary", JOptionPane.ERROR_MESSAGE);
                return;
            }

            if (newPerformanceIndex < 0 || newPerformanceIndex > 10) {
                JOptionPane.showMessageDialog(frame, "Invalid performance index", "Error", JOptionPane.ERROR_MESSAGE);
                return;
            }

            // Iterate over teacherArrayList to find the teacher with the provided ID
        } catch (NumberFormatException e) {
            // Runtime error: Cannot convert double to integer
        }
    }
});
```

Figure 31: Runtime error.

The error occurs because the new salary is stored as an integer data type, causing an issue when entering a salary as double. To resolve this error, we can change this as:



```
setSalaryBtn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        try {
            String id = getTextFromTF(teacherIdSF);
            String salary = getTextFromTF(newSalarySF);
            String performance = getTextFromTF(newPerformanceSF);

            // Convert input strings to appropriate data types
            int teacherId = Integer.parseInt(id);
            double newSalary = Double.parseDouble(salary);
            int newPerformanceIndex = Integer.parseInt(performance);
        } catch (NumberFormatException e) {
            // Runtime error: Cannot convert double to integer
        }
    }
});
```

Figure 32: Correction of runtime error.

7 Conclusion

Finally, I reached to the conclusion part of this coursework, the most interesting part of this coursework. This is the closing section of this coursework and documentation. The report is the documentation of a program developed for the module 'Programming Java'. At the starting of this coursework seemed bit confusing and irritating for me. I cannot manage time for this coursework but with the help of Gantt chart I did. In this coursework we must designed Gui for the coursework that was given in first semester. We must implement GUI system for Teacher, Tutor, and Lecturer class.

At the beginning I did not even understand the question for this coursework but with the help of my module leader, tutor, and other online means I got it. Since this was the first programming in GUI, I must tackle with multiple problems like GUI design, input validation, exception handling etc. I encountered many bugs and errors while implementing action listener to the buttons but with the helps of my friends and teacher I did it.

Finally, it was very nice and important coursework that taught me many things that I had not learned before. From this coursework I got platform to use theoretical programming knowledge in real life scenarios which helps me to explore the fundamental concepts in java's GUI, AWT and Swing module. From the first semester this module has been a great journey for me. I have gained a lot of knowledge that I never knew before. I am very thankful for the teacher and this module. At last, I appreciate the entire course for providing me with this valuable learning experience.

8 References

By Tech Target, 2019. *Class Diagram*. [Online]

Available at: <https://www.techtarget.com/searchapparchitecture/definition/class-diagram>
[Accessed 16 04 2024].

ibm.com, 2023. *what is java?*. [Online]

Available at:

<https://www.ibm.com/topics/java#:~:text=Java%20is%20a%20widely%20used,the%20C%20and%20C%2B%2B%20languages.>

[Accessed 16 4 2024].

IBM.com, 2023. *What is Java?*. [Online]

Available at:

<https://www.ibm.com/topics/java#:~:text=Java%20is%20a%20widely%20used,the%20C%20and%20C%2B%2B%20languages.>

[Accessed 16 4 2024].

Rouse, M., 2013. *What Does BlueJ Mean?*. [Online]

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

[Accessed 16 4 2024].

The Economic Times, n.d. *What is pseudocode*. [Online]

Available at: <https://economictimes.indiatimes.com/definition/pseudocode>
[Accessed 18 4 2024].

9 Appendix

```
/**
 * Write a description of class TeacherGUI here.
 *
 * @author BIDUR SIWAKOTI
 * @Final version/ 2024-03-05
 */

import com.sun.source.tree.NewArrayTree;

import javax.swing.*;
import javax.swing.border.Border;
import javax.swing.border.TitledBorder;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import javax.swing.text.JTextComponent;

public class TeacherGUI {
    private ArrayList<Teacher> teacherArrayList = new ArrayList<>();
    private JFrame frame;

    private JPanel panel, pannelLecturer, tutorPannel, btnPannel;
    private JLabel teacherIDL, teacherNameL, addressL, workingTypeL,
    employmentStatusL, workingHourL, departmentL,
        yearOfExperienceL, gradeScoreL, hasGradedL;
```

```
private JLabel salaryL, specializationL, academicQualificationL, performanceIndexL,  
isCertifiedL;
```

```
private JTextField salaryF, specializationF, academicQualificationF,  
performanceIndexF, isCertifiedF;
```

```
private JTextField teacherIDF, teacherNameF, addressF, workingTypeF,  
employmentStatusF, workingHourF, departmentF,  
yearOfExperienceF, gradeScoreF, hasGradedF;
```

```
private JButton addLecturerBtn, gradeAssignmentBtn, setSalaryBtn, removeTutorBtn,  
clearBtn, displayBtn, addtutorbtn;
```

```
Font myFont = new Font("Impact", Font.ITALIC, 40);
```

```
public TeacherGUI() {
```

```
    frame = new JFrame("Teacher Management System");
```

```
    frame.setSize(900, 770);
```

```
    frame.setFont(myFont);
```

```
    frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
```

```
    frame.setLayout(null);
```

```
    frame.setLocationRelativeTo(null);
```

```
    // attributes for teacher class
```

```
    panel = new JPanel();
```

```
    panel.setBounds(20, 20, 400, 300);
```

```
    panel.setBackground(Color.gray);
```

```
    panel.setLayout(null);
```

```
    TitledBorder titledBorder = BorderFactory.createTitledBorder(" Teacher Class");
```

```
    titledBorder.setTitleColor(Color.WHITE);
```

```
    panel.setBorder(titledBorder);
```

```
    Border border = BorderFactory.createLineBorder(Color.darkGray, 3);
```

```
panel.setFont(myFont);
```

```
teacherIDL = new JLabel("Teacher ID: ");
```

```
teacherIDL.setBounds(20, 30, 100, 25);
```

```
teacherNameL = new JLabel("Teacher Name: ");
```

```
teacherNameL.setBounds(20, 70, 100, 25);
```

```
addressL = new JLabel("Address: ");
```

```
addressL.setBounds(20, 110, 100, 25);
```

```
workingTypeL = new JLabel("Working Type: ");
```

```
workingTypeL.setBounds(20, 150, 100, 25);
```

```
employmentStatusL = new JLabel("Employment Status: ");
```

```
employmentStatusL.setBounds(20, 190, 120, 25);
```

```
workingHourL = new JLabel("Working Hours: ");
```

```
workingHourL.setBounds(20, 230, 100, 25);
```

```
teacherIDF = new JTextField();
```

```
teacherIDF.setBounds(150, 30, 120, 25);
```

```
teacherNameF = new JTextField();
```

```
teacherNameF.setBounds(150, 70, 120, 25);
```

```
addressF = new JTextField();
```

```
addressF.setBounds(150, 110, 120, 25);
```

```
workingTypeF = new JTextField();
workingTypeF.setBounds(150, 150, 120, 25);

employmentStatusF = new JTextField();
employmentStatusF.setBounds(150, 190, 120, 25);

workingHourF = new JTextField();
workingHourF.setBounds(150, 230, 120, 25);

// attributes for lecture clas
pannelLecturer = new JPanel();
pannelLecturer.setBounds(450, 20, 400, 300);
pannelLecturer.setBackground(Color.gray);
pannelLecturer.setLayout(null);
TitledBorder titledBorder1 = BorderFactory.createTitledBorder(" Lecturer Class");
titledBorder1.setTitleColor(Color.WHITE);
pannelLecturer.setBorder(titledBorder1);

departmentL = new JLabel("Department: ");
departmentL.setBounds(65, 30, 100, 25);

yearOfExperienceL = new JLabel("Year of Experience: ");
yearOfExperienceL.setBounds(20, 65, 130, 25);

gradeScoreL = new JLabel("Graded score: ");
gradeScoreL.setBounds(53, 105, 100, 25);
```

```
hasGradedL = new JLabel("Has Graded: ");
hasGradedL.setBounds(57, 145, 100, 25);

departmentF = new JTextField();
departmentF.setBounds(150, 30, 130, 25);

yearOfExperienceF = new JTextField();
yearOfExperienceF.setBounds(150, 65, 130, 25);

gradeScoreF = new JTextField();
gradeScoreF.setBounds(150, 105, 130, 25);

hasGradedF = new JTextField();
hasGradedF.setBounds(150, 145, 130, 25);

addLecturerBtn = new JButton("Add Lecture");
addLecturerBtn.setBounds(50, 190, 120, 35);
addLecturerBtn.setBackground(Color.BLACK);
addLecturerBtn.setForeground(Color.WHITE);
addLecturerBtn.setBorder(border);

gradeAssignmentBtn = new JButton("Graded Assignment");
gradeAssignmentBtn.setBounds(200, 190, 180, 35);
gradeAssignmentBtn.setBackground(Color.BLACK);
gradeAssignmentBtn.setForeground(Color.WHITE);
gradeAssignmentBtn.setBorder(border);

// panel for tutor
tutorPannel = new JPanel();
```

```
tutorPannel.setBounds(20, 350, 400, 300);  
tutorPannel.setBackground(Color.GRAY);  
tutorPannel.setLayout(null);  
TitledBorder titledBorder2 = BorderFactory.createTitledBorder(" Tutor Class");  
titledBorder2.setTitleColor(Color.WHITE);  
tutorPannel.setBorder(titledBorder2);
```

```
salaryL = new JLabel("Salary: ");  
salaryL.setBounds(105, 30, 100, 25);
```

```
specializationL = new JLabel("Specializations: ");  
specializationL.setBounds(53, 70, 100, 25);
```

```
academicQualificationL = new JLabel("Academic Qualification: ");  
academicQualificationL.setBounds(13, 110, 135, 25);
```

```
performanceIndexL = new JLabel("Performance Index: ");  
performanceIndexL.setBounds(35, 150, 130, 25);
```

```
isCertifiedL = new JLabel("Is certified: ");  
isCertifiedL.setBounds(82, 190, 100, 25);
```

```
salaryF = new JTextField();  
salaryF.setBounds(155, 30, 130, 25);
```

```
specializationF = new JTextField();  
specializationF.setBounds(155, 70, 130, 25);
```



```
academicQualificationF = new JTextField();  
academicQualificationF.setBounds(155, 110, 130, 25);
```

```
performanceIndexF = new JTextField();  
performanceIndexF.setBounds(155, 150, 130, 25);
```

```
isCertifiedF = new JTextField();  
isCertifiedF.setBounds(155, 190, 130, 25);
```

```
addtutorbtn = new JButton("Add Tutor");  
addtutorbtn.setBounds(30, 190, 100, 30);  
addtutorbtn.setBackground(Color.BLACK);  
addtutorbtn.setForeground(Color.WHITE);  
addtutorbtn.setBorder(border);
```

```
setSalaryBtn = new JButton("Set Salary");  
setSalaryBtn.setBounds(140, 190, 100, 30);  
setSalaryBtn.setBackground(Color.BLACK);  
setSalaryBtn.setForeground(Color.WHITE);  
setSalaryBtn.setBorder(border);
```

```
removeTutorBtn = new JButton("Remove Tutor");  
removeTutorBtn.setBounds(250, 190, 100, 30);  
removeTutorBtn.setBackground(Color.BLACK);  
removeTutorBtn.setForeground(Color.WHITE);  
removeTutorBtn.setBorder(border);
```

```
btnPannel = new JPanel();
```

```
btnPannel.setBounds(450, 400, 300, 100);
btnPannel.setBackground(Color.gray);
btnPannel.setLayout(null);
TitledBorder titledBorder3 = BorderFactory.createTitledBorder("Buttons");
titledBorder3.setTitleColor(Color.WHITE);

btnPannel.setBorder(titledBorder3);

displayBtn = new JButton("Display");
displayBtn.setBounds(20, 30, 120, 25);
displayBtn.setBackground(Color.BLACK);
displayBtn.setForeground(Color.WHITE);
displayBtn.setBorder(border);

clearBtn = new JButton("Clear");
clearBtn.setBounds(160, 30, 120, 25);
clearBtn.setBackground(Color.BLACK);
clearBtn.setForeground(Color.WHITE);
clearBtn.setBorder(border);

panel.add(teacherIDL);
panel.add(teacherIDF);
panel.add(teacherNameL);
panel.add(teacherNameF);
panel.add(addressL);
panel.add(addressF);
panel.add(workingTypeL);
panel.add(workingTypeF);
```

```
panel.add(employmentStatusL);  
panel.add(employmentStatusF);  
panel.add(workingHourL);  
panel.add(workingHourF);
```

```
pannelLecturer.add(departmentL);  
pannelLecturer.add(departmentF);  
pannelLecturer.add(yearOfExperienceL);  
pannelLecturer.add(yearOfExperienceF);  
pannelLecturer.add(gradeScoreL);  
pannelLecturer.add(gradeScoreF);  
// pannelLecturer.add(hasGradedL);  
// pannelLecturer.add(hasGradedF);  
pannelLecturer.add(addLecturerBtn);  
pannelLecturer.add(gradeAssignmentBtn);
```

```
tutorPannel.add(salaryL);  
tutorPannel.add(salaryF);  
tutorPannel.add(specializationL);  
tutorPannel.add(specializationF);  
tutorPannel.add(academicQualificationL);  
tutorPannel.add(academicQualificationF);  
tutorPannel.add(performanceIndexL);  
tutorPannel.add(performanceIndexF);  
// tutorPannel.add(isCertifiedL);  
// tutorPannel.add(isCertifiedF);  
tutorPannel.add(setSalaryBtn);  
tutorPannel.add(removeTutorBtn);
```

```
tutorPannel.add(addtutorbtn);

btnPannel.add(displayBtn);
btnPannel.add(clearBtn);

Component teacher = frame.add(panel);
Component lecturer = frame.add(pannellLecturer);
Component tutor = frame.add(tutorPannel);
Component bnpannl = frame.add(btnPannel);
frame.setVisible(true);

addLecturerBtn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        addLecturer();
    }
});

gradeAssignmentBtn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        gradeAssignment();
    }
});

// for tutor adding buttons
addtutorbtn.addActionListener(new ActionListener() {
```

```
@Override
public void actionPerformed(ActionEvent e) {
    addTutor();

}

});

setSalaryBtn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        setSalary();
    }
});

removeTutorBtn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        removeTutor();
    }
});

clearBtn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        clear();
    }
});
```

```
displayBtn.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        displayInfo();  
    }  
});  
}  
  
public void addTutor() {  
    try {  
        String teacherId = getTextFromTF(teacherIDF);  
        String teacherName = getTextFromTF(teacherNameF);  
        String address = getTextFromTF(addressF);  
        String workingType = getTextFromTF(workingTypeF);  
        String employmentStatus = getTextFromTF(employmentStatusF);  
        String workingHours = getTextFromTF(workingHourF);  
        String salary = getTextFromTF(salaryF);  
        String specialization = getTextFromTF(specializationF);  
        String academicQualification = getTextFromTF(academicQualificationF);  
        String performanceIndex = getTextFromTF(performanceIndexF);  
  
        int teacherID = Integer.parseInt(teacherId);  
        int workingHour = Integer.parseInt(workingHours);  
  
        Double salaries = Double.parseDouble(salary);  
        int performanceInd = Integer.parseInt(performanceIndex);  
        if (salaries < 0 && workingHour < 20) {
```

```
        JOptionPane.showMessageDialog(frame,
            "Salary cannot be smaller than 1 and \n Working hours must be greater
than 20", "Invalid",
            JOptionPane.ERROR_MESSAGE);
        return;
    }
    if (performanceInd < 5 || performanceInd > 10) {
        JOptionPane.showMessageDialog(frame, "Performance index must be
between 5 and 10", "Invalid",
            JOptionPane.ERROR_MESSAGE);
        return;
    }

    int confirm = JOptionPane.showConfirmDialog(tutorPannel, "Please confirm the
entered information",
        "Confirmation", JOptionPane.INFORMATION_MESSAGE);
    if (!isIdUnique(teacherID)) {
        JOptionPane.showMessageDialog(frame, "Teacher ID already exists. Please
choose a different ID.",
            "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }
    if (confirm != JOptionPane.YES_OPTION) {
        return;
    }

    Tutor tutor = new Tutor(teacherID, teacherName, address, workingType,
employmentStatus, workingHour,
        salaries, specialization, academicQualification, performanceInd);
    teacherArrayList.add(tutor);
```

```
        JOptionPane.showMessageDialog(frame, "Tutor added successfully!",
    "Success",
        JOptionPane.INFORMATION_MESSAGE);
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(frame, "Invalid input. Please check the inputs
    and try again.", "Error",
        JOptionPane.ERROR_MESSAGE);
    } catch (IllegalArgumentException e) {
        JOptionPane.showMessageDialog(frame, "Please fill in all required fields.",
    "Error",
        JOptionPane.ERROR_MESSAGE);
    }
}

public void addLecturer() {
    try {
        String teacherId = getTextFromTF(teacherIDF);
        String teacherName = getTextFromTF(teacherNameF);
        String address = getTextFromTF(addressF);
        String workingType = getTextFromTF(workingTypeF);
        String employmentStatus = getTextFromTF(employmentStatusF);
        String workingHours = getTextFromTF(workingHourF);
        String department = getTextFromTF(departmentF);
        String yearsOfExperience = getTextFromTF(yearOfExperienceF);
        String gradeScore = getTextFromTF(gradeScoreF);
        int score = Integer.parseInt(gradeScore);
        if (score < 0 || score >= 100) {
            JOptionPane.showMessageDialog(frame, "Grade score must be greater than
    0 and \n smaller  than 100",
                "Invalid", JOptionPane.ERROR_MESSAGE);
        }
    }
}
```



```
        return;
    }

    int id = Integer.parseInt(teacherId);
    int yearOfExperience = Integer.parseInt(yearsOfExperience);
    int workingHour = Integer.parseInt(workingHours);
    if (yearOfExperience < 5 && workingHour < 20) {
        JOptionPane.showMessageDialog(frame,
            "Year of experiences must be greater than 5 and \n Working hours must
            be greater than 20",
            "Invalid", JOptionPane.ERROR_MESSAGE);
        return;
    }

    int confirm = JOptionPane.showConfirmDialog(pannelLecturer, "Please confirm
    the entered information",
        "Confirmation", JOptionPane.INFORMATION_MESSAGE);
    if (!isIdUnique(id)) {
        JOptionPane.showMessageDialog(frame, "Teacher ID already exists. Please
        choose a different ID.",
            "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }
    if (confirm != JOptionPane.YES_OPTION) {
        return;
    }

    Lecturer lecturer = new Lecturer(id, teacherName, address, workingType,
    employmentStatus, department,
        yearOfExperience, workingHour);
    teacherArrayList.add(lecturer);
```

```
        JOptionPane.showMessageDialog(frame, "Lecturer added successfully!",
        "Success",
            JOptionPane.INFORMATION_MESSAGE);
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(frame, "Invalid input. Please check the inputs
        and try again.", "Error",
            JOptionPane.ERROR_MESSAGE);
    } catch (IllegalArgumentException e) {
        JOptionPane.showMessageDialog(frame, "Please fill in all required fields.",
        "Error",
            JOptionPane.ERROR_MESSAGE);
    }
}

public void gradeAssignment() {
    JFrame frame1 = new JFrame("Grading Assignment");
    frame1.setSize(400, 300);
    frame1.setLayout(null);
    frame1.setLocationRelativeTo(null);
    Border border = BorderFactory.createLineBorder(Color.darkGray, 3);

    JLabel teacherIdAL = new JLabel("TeacherId: ");
    teacherIdAL.setBounds(65, 30, 100, 25);
    JLabel gradedScoresAL = new JLabel("Graded Scores: ");
    gradedScoresAL.setBounds(40, 70, 100, 25);
    JLabel departmentAL = new JLabel("Department: ");
    departmentAL.setBounds(60, 110, 100, 25);
    JLabel yearOfExpAL = new JLabel("Year of Experiences: ");
    yearOfExpAL.setBounds(10, 150, 140, 25);
```

```
JTextField teacherIdAF = new JTextField();
teacherIdAF.setBounds(140, 30, 100, 25);
JTextField gradedScoresAF = new JTextField();
gradedScoresAF.setBounds(140, 70, 100, 25);
JTextField departmentAF = new JTextField();
departmentAF.setBounds(140, 110, 100, 25);
JTextField yearOfExpAF = new JTextField();
yearOfExpAF.setBounds(140, 150, 100, 25);
```

```
JButton gradeBtn = new JButton("Grade");
gradeBtn.setBounds(20, 190, 100, 25);
gradeBtn.setBackground(Color.BLACK);
gradeBtn.setForeground(Color.WHITE);
gradeBtn.setBorder(border);
JButton clearBtn = new JButton("Clear");
clearBtn.setBounds(140, 190, 100, 25);
clearBtn.setBackground(Color.BLACK);
clearBtn.setForeground(Color.WHITE);
clearBtn.setBorder(border);
```

```
frame1.add(teacherIdAL);
frame1.add(gradedScoresAL);
frame1.add(departmentAL);
frame1.add(yearOfExpAL);
frame1.add(gradedScoresAF);
frame1.add(teacherIdAF);
frame1.add(departmentAF);
```

```
frame1.add(yearOfExpAF);
frame1.add(gradeBtn);
frame1.add(clearBtn);
frame1.setVisible(true);

gradeBtn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {

        try {

            String ID = getTextFromTF(teacherIdAF);
            String score = getTextFromTF(gradedScoresAF);
            String department = getTextFromTF(departmentAF);
            int teacherId = Integer.parseInt(ID);
            int gradedScore = Integer.parseInt(score);
            int yearsOfExperience = Integer.parseInt(yearOfExpAF.getText());
            // find the lecturer with given id
            Lecturer lecturer = findLecturer(teacherId);
            if (lecturer == null) {
                JOptionPane.showMessageDialog(frame1, "Lecturer with provided id" +
                    teacherId + " is not found",
                    "Error", JOptionPane.ERROR_MESSAGE);
                return;
            }
            if
(!department.strip().toLowerCase().equals(lecturer.getDepartment().strip().toLowerCase
())) {
                JOptionPane.showMessageDialog(frame, "Department must be same to
grade assignment",
```

```
        "error department", JOptionPane.ERROR_MESSAGE);
    return;
}
if (gradedScore > 100 || gradedScore < 0) {
    JOptionPane.showMessageDialog(frame, "Grade score must be
between 0 and 100", "Error grade",
        JOptionPane.ERROR_MESSAGE);
    return;
}
if (yearsOfExperience < 5) {
    JOptionPane.showMessageDialog(frame, "year of experiences must be
grater than 5", "Error",
        JOptionPane.ERROR_MESSAGE);
    return;
}
if (lecturer != null) {
    lecturer.gradeAssignment(gradedScore, department,
yearsOfExperience);
    JOptionPane.showMessageDialog(frame, "Teacher ID:" +
lecturer.getTeacherId()
        + "\n" +
        "Graded score: " + gradedScore + "\n" +
        "Department: " + department + "\n" +
        "year of experiences: " + yearsOfExperience + "\n");
    JOptionPane.showMessageDialog(frame1,
        "Grading assignment is successful and grade score is displayed in
terninal");
    return;
}
} catch (NumberFormatException ex) {
```

```
        JOptionPane.showMessageDialog(frame,
            "Teacher id, grade score, year of experiences must be positive
integer", "Error",
            JOptionPane.ERROR_MESSAGE);
    } catch (IllegalArgumentException ex) {
        JOptionPane.showMessageDialog(frame, "Please fill all the empty fields",
"Error",
            JOptionPane.ERROR_MESSAGE);
    }
}

});

clearBtn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        teacherIdAF.setText("");
        gradedScoresAF.setText("");
        departmentAF.setText("");
        yearOfExpAF.setText("");
    }
});

}

public void setSalary() {
    JFrame frame2 = new JFrame("Salary Setting");
    frame2.setSize(350, 250);
    frame2.setLayout(null);
```

```
frame2.setLocationRelativeTo(null);  
Border border = BorderFactory.createLineBorder(Color.darkGray, 3);  
  
JLabel teacherID = new JLabel("Teacher ID: ");  
teacherID.setBounds(90, 30, 100, 25);  
  
JTextField teacherIdSF = new JTextField();  
teacherIdSF.setBounds(170, 30, 130, 25);  
  
JLabel newSalaryL = new JLabel(" New salary: ");  
newSalaryL.setBounds(90, 70, 100, 25);  
  
JTextField newSalarySF = new JTextField();  
newSalarySF.setBounds(170, 70, 130, 25);  
  
JLabel newPerformance = new JLabel("New Performance Index:");  
newPerformance.setBounds(20, 110, 170, 25);  
  
JTextField newPerformanceSF = new JTextField();  
newPerformanceSF.setBounds(170, 110, 130, 25);  
  
JButton setSalaryBtn = new JButton("Set Salary");  
setSalaryBtn.setBounds(40, 150, 100, 25);  
setSalaryBtn.setBackground(Color.BLACK);  
setSalaryBtn.setForeground(Color.WHITE);  
setSalaryBtn.setBorder(border);  
JButton clearBtn = new JButton("Clear");  
clearBtn.setBounds(160, 150, 100, 25);
```

```
clearBtn.setBackground(Color.BLACK);
clearBtn.setForeground(Color.WHITE);
clearBtn.setBorder(border);

frame2.add(teacherID);
frame2.add(teacherIdSF);
frame2.add(newSalaryL);
frame2.add(newSalarySF);
frame2.add(newPerformance);
frame2.add(newPerformanceSF);
frame2.add(setSalaryBtn);
frame2.add(clearBtn);
frame2.setVisible(true);

setSalaryBtn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        try {
            String id = getTextFromTF(teacherIdSF);
            String salary = getTextFromTF(newSalarySF);
            String performance = getTextFromTF(newPerformanceSF);

            // Convert input strings to appropriate data types
            int teacherId = Integer.parseInt(id);
            double newSalary = Double.parseDouble(salary);
            int newPerformanceIndex = Integer.parseInt(performance);

            // Input validation
```



```
        if (newSalary < 0) {
            JOptionPane.showMessageDialog(frame, "Salary cannot be a negative
value", "Invalid Salary",
                JOptionPane.ERROR_MESSAGE);
            return;
        }

        if (newPerformanceIndex < 0 || newPerformanceIndex > 10) {
            JOptionPane.showMessageDialog(frame, "Invalid performance index",
"Error",
                JOptionPane.ERROR_MESSAGE);
            return;
        }

        // Iterate over teacherArrayList to find the teacher with the provided ID
        for (Teacher teacher : teacherArrayList) {
            if (teacher.getTeacherId() == teacherId) {
                if (teacher instanceof Lecturer) {
                    JOptionPane.showMessageDialog(frame2, "Lecturer cannot have
their salary modified",
                        "Error", JOptionPane.ERROR_MESSAGE);
                    return;
                }

                Tutor tutor = (Tutor) teacher;
                if (tutor.isCertified()) {
                    JOptionPane.showMessageDialog(frame,
                        "Tutor is already certified. Salary cannot be modified",
"Certified Tutor",
                        JOptionPane.ERROR_MESSAGE);
```

```
        return;
    }

    if (newPerformanceIndex > 5 && tutor.getWorkingHours() > 20) {
        tutor.setSalary(newSalary, newPerformanceIndex);
        JOptionPane.showMessageDialog(frame, "Salary for tutor
successfully updated", "Tutor",
        JOptionPane.INFORMATION_MESSAGE);
    } else {
        tutor.setSalary(newSalary, newPerformanceIndex);
        JOptionPane.showMessageDialog(frame, "Salary updated
successfully for tutor", "Tutor",
        JOptionPane.INFORMATION_MESSAGE);
    }
    // Additional message to display updated information
    JOptionPane.showMessageDialog(frame,
        "Teacher ID: " + teacherId + "\nNew Salary: " + newSalary
        + "\nNew Performance Index: " + newPerformanceIndex,
        "Salary Updated", JOptionPane.INFORMATION_MESSAGE);
    return;
}
}

JOptionPane.showMessageDialog(frame2, "No tutor found with the
provided ID: " + teacherId, "Error",
    JOptionPane.ERROR_MESSAGE);
} catch (NumberFormatException exception) {
    JOptionPane.showMessageDialog(frame,
        "Please enter numerical values for Teacher ID, Salary, and
Performance Index", "Error",
```

```
        JOptionPane.ERROR_MESSAGE);
    } catch (IllegalArgumentException exception) {
        JOptionPane.showMessageDialog(frame, "Invalid Inputs, Please fills the
values in text fields",
        "Error", JOptionPane.ERROR_MESSAGE);
    }
}

});

clearBtn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        teacherIdSF.setText("");
        newSalarySF.setText("");
        newPerformanceSF.setText("");
    }
});

}

public void displayInfo() {
    StringBuilder displayText = new StringBuilder();
    if (teacherArrayList.isEmpty()) {
        JOptionPane.showMessageDialog(frame, "No teachers to display in array list.",
"Empty List",
        JOptionPane.INFORMATION_MESSAGE);
        return;
    }
    for (Teacher teacher : teacherArrayList) {
```

```
        if (teacher instanceof Lecturer) {
            displayText.append("Lecturer:\n");
            displayText.append("Teacher ID: ").append(((Lecturer)
teacher).getTeacherId()).append("\n");
            displayText.append("Teacher Name: ").append(((Lecturer)
teacher).getTeacherName()).append("\n");
            displayText.append("Address: ").append(((Lecturer)
teacher).getAddress()).append("\n");
            displayText.append("Working Type: ").append(((Lecturer)
teacher).getWorkingType()).append("\n");
            displayText.append("Employment Status: ").append(((Lecturer)
teacher).getEmploymentStatus())
                .append("\n");
            displayText.append("Working Hours: ").append(((Lecturer)
teacher).getWorkingHours()).append("\n");
            displayText.append("Department: ").append(((Lecturer)
teacher).getDepartment()).append("\n");
            displayText.append("Years of Experience: ").append(((Lecturer)
teacher).getYearsOfExperience())
                .append("\n\n");
            System.out.println("Lecturer Information: ");
            System.out.println(displayText.toString());
        } else if (teacher instanceof Tutor) {
            displayText.append("Tutor:\n");
            displayText.append("Teacher ID: ").append(((Tutor)
teacher).getTeacherId()).append("\n");
            displayText.append("Teacher Name: ").append(((Tutor)
teacher).getTeacherName()).append("\n");
            displayText.append("Address: ").append(((Tutor)
teacher).getAddress()).append("\n");
            displayText.append("Working Type: ").append(((Tutor)
teacher).getWorkingType()).append("\n");
```

```
        displayText.append("Employment Status: ").append(((Tutor)
teacher).getEmploymentStatus()).append("\n");

        displayText.append("Salary: ").append(((Tutor)
teacher).getSalary()).append("\n");

        displayText.append("Specialization: ").append(((Tutor)
teacher).getSpecialization()).append("\n");

        // displayText.append("Academic Qualification: ").append(((Tutor)
// teacher).getacademicQualification()).append("\n");

        displayText.append("Performance Index: ").append(((Tutor)
teacher).getPerformanceIndex())

                .append("\n\n");

        System.out.println(displayText.toString());
    }
}

JOptionPane.showMessageDialog(frame, displayText.toString(), "Teacher
Information",

        JOptionPane.INFORMATION_MESSAGE);
}

public void clear() {
    // Clear text fields for teacher class
    teacherIDF.setText("");
    teacherNameF.setText("");
    addressF.setText("");
    workingTypeF.setText("");
    employmentStatusF.setText("");
    workingHourF.setText("");

    // Clear text fields for lecturer class
    departmentF.setText("");
}
```

```
yearOfExperienceF.setText("");
gradeScoreF.setText("");
hasGradedF.setText("");

// Clear text fields for tutor class
salaryF.setText("");
specializationF.setText("");
academicQualificationF.setText("");
performanceIndexF.setText("");
isCertifiedF.setText("");
}

public void removeTutor() {
    try {
        // Ask the user to enter the Teacher ID
        String teacherIdInput = JOptionPane.showInputDialog(frame, "Enter Teacher
ID:");
        if (teacherIdInput == null) {
            return;
        }

        if (teacherIdInput != null && !teacherIdInput.isEmpty()) { // Check if user entered
a value
            int teacherId = Integer.parseInt(teacherIdInput);
            Tutor tutorToRemove = findTutor(teacherId);
            if (tutorToRemove != null) {
                if (!tutorToRemove.isCertified()) {
                    int confirm = JOptionPane.showConfirmDialog(frame,
                        "Are you sure you want to remove this tutor?", "Confirmation",
```

```
        JOptionPane.YES_NO_OPTION);
    if (confirm == JOptionPane.YES_OPTION) {
        teacherArrayList.remove(tutorToRemove);
        JOptionPane.showMessageDialog(frame, "Tutor removed
successfully!", "Success",
        JOptionPane.INFORMATION_MESSAGE);
    }
    } else {
        JOptionPane.showMessageDialog(frame, "Certified tutor cannot be
removed.", "Error",
        JOptionPane.ERROR_MESSAGE);
    }
    } else {
        JOptionPane.showMessageDialog(frame, "Tutor with ID " + teacherId + "
not found.", "Error",
        JOptionPane.ERROR_MESSAGE);
    }
    } else {
        JOptionPane.showMessageDialog(frame, "Please enter a valid Teacher ID.",
"Error",
        JOptionPane.ERROR_MESSAGE);
    }
    } catch (NumberFormatException ex) {
        JOptionPane.showMessageDialog(frame, "Invalid input. Please enter a valid
Teacher ID.", "Error",
        JOptionPane.ERROR_MESSAGE);
    }
    }

    private String getText(JTextField F) {
```

```
String texts = F.getText().trim();
if (texts.isEmpty()) {
    throw (new IllegalArgumentException());
}
return texts;
}
```

```
private boolean isIdUnique(int teacherId) {
    for (Teacher teacher : teacherArrayList) {
        if (teacher.getTeacherId() == teacherId) {
            return false;
        }
    }
    return true;
}
```

```
private String getTextFromTF(JTextField field) {
    String text = getText(field);
    if (text.isEmpty()) {
        throw (new IllegalArgumentException());
    }
    return text;
}
```

```
private Lecturer findLecturer(int teacherId) {
    for (Teacher teacher : teacherArrayList) {
        if (teacher instanceof Lecturer && teacher.getTeacherId() == teacherId) {
            return (Lecturer) teacher;
        }
    }
}
```



```
    }  
    }  
    return null;  
}  
  
public Teacher getTeacherById(int id) {  
    // Iterate through your list of teachers and find the one with the matching ID  
    for (Teacher teacher : teacherArrayList) {  
        if (teacher.getTeacherId() == id) {  
            return teacher;  
        }  
    }  
    // If no teacher with the given ID is found, return null  
    return null;  
}  
  
private Tutor findTutor(int teacherId) {  
    for (Teacher teacher : teacherArrayList) {  
        if (teacher instanceof Tutor && teacher.getTeacherId() == teacherId) {  
            return (Tutor) teacher;  
        }  
    }  
    return null; // Return null if tutor with given ID is not found  
}  
  
public static void main(String[] args) {  
    TeacherGUI gui = new TeacherGUI();  
}  
}
```

10 Plagiarism Test**Originality report****COURSE NAME**

Programming 2023-24 Autumn

STUDENT NAME

IIC Bldur

FILE NAME Documentation**REPORT CREATED**

May 7, 2024

Summary

Flagged passages	11	3%
Cited/quoted passages	3	0.6%

Web matches

studyx.ai	3	0.8%
ibm.com	3	0.6%
indiatimes.com	1	0.5%
codechef.com	1	0.4%
testbook.com	1	0.4%
naukri.com	1	0.4%
process.st	1	0.3%
studysmarter.co.uk	1	0.3%
vaia.com	1	0.2%

appwrite.io

1

0.1%

1 of 14 passages

Student passage FLAGGED

It is also **object-oriented programming language and software** platforms that runs on billions of devices, including notebook computers, mobile devices, gaming consoles, medical devices and many

Top web match

It is a widely-used **object-oriented programming language and software** platform that runs on billions of devices, including notebook computers, mobile devices, gaming consoles, medical devices, and... [Solved] A server uses _____ application software to support the https://testbook.com/question-answer/aserver-uses-_____-application-software-to-su-6113a4e7b9a64c426e97546b

2 of 14 passages

Student passage CITED

...mobile devices, gaming consoles, medical devices and many other. **The rules and syntax of java are based on the C and C++ languages.** (IBM.com, 2023). The major advantage of developing software with...

Top web match

Java is a widely used object-oriented programming language and software platform that runs on billions of devices, including notebook computers, mobile devices, gaming consoles, medical devices and... What Is Java? - IBM <https://www.ibm.com/topics/java>

3 of 14 passages

Student passage FLAGGED

...on the C and C++ languages. (IBM.com, 2023). The **major advantage of developing software with java is its portability.** Once a code is wrote it can run in...

Top web match

The rules and syntax of Java are based on the C and C++ languages. One **major advantage of developing software with Java is its portability.**

What Is Java? - IBM <https://www.ibm.com/topics/java>

4 of 14 passages

Student passage CITED

...code is wrote it can run in any devices. **When the language was invented in 1991 by James Gosling of sun Microsystems, the primary goal was to be able to “write once, run anywhere”.** (ibm.com, 2023)

Top web match

Once you wrote code for a Java program on a notebook computer, it can be easily moved to a mobile device.

When the language was invented in 1991 by James Gosling of Sun Microsystems (later acquired by...

What Is Java? - IBM <https://www.ibm.com/topics/java>

5 of 14 passages

Student passage FLAGGED

Microsoft word, a widely used word processing software developed by Microsoft Corporation, offers a comprehensive set of tools for creating, editing, formatting, and sharing documents

Top web match

Microsoft Word, a widely used word processing application developed by Microsoft Corporation, provides a comprehensive set of tools for effectively editing documents. Editing in Microsoft Word is like...

How To Edit In Microsoft Word - Process Street <https://www.process.st/how-to/edit-in-microsoft-word/>

6 of 14 passages

Student passage FLAGGED

Pseudocode is an informal way of programming description that does not require any strict programming language syntax or underlying technological considerations. It is used for creating an outline or...

Top web match

Definition: **Pseudocode is an informal way of programming description that does not require any strict programming language syntax or underlying technology considerations. It is used for creating an...**

What is Pseudocode? Definition of ... - The Economic Times <https://economictimes.indiatimes.com/definition/pseudocode>

7 of 14 passages

Student passage QUOTED

DISPLAY error message "Invalid input. **Please check the inputs and try again.**" using
JOptionPane.ERROR_MESSAGE

Top web match

There was an error processing your request. **Please check the inputs and try again.** (400) ...
Developers are encountering a 400 error when trying ...

There was an error processing your request. Please check the
... <https://appwrite.io/threads/1236626677514240122>

8 of 14 passages

Student passage FLAGGED

This method begins by retrieving **input** data entered by the user from the text fields in the GUI which includes attributes likes **teacher ID, name, address, working type, employment status, working...**

Top web match

Add a Tutor When **this** button is pressed, the **input** values of **teacher Id, teacher name, address, working type, employment status, working hours, salary, specialization, academic qualifications** and...

our GUI make it look more modern using awt | StudyX <https://studyx.ai/homework/100214603-our-gui-make-itlook-more-modern-using-awt-swing-should-contain-the-same-components-but>

9 of 14 passages

Student passage FLAGGED

The method begins by retrieving **input** data entered by the user from the text fields in the GUI that includes attributes likes **teacher ID, name, address, working type, employment status, working hours,...**

Top web match

Add a Tutor When **this** button is pressed, **the input** values of **teacher Id, teacher name, address, working type, employment status, working hours, salary, specialization, academic qualifications** and...

our GUI make it look more modern using awt | StudyX <https://studyx.ai/homework/100214603-our-gui-make-itlook-more-modern-using-awt-swing-should-contain-the-same-components-but>

10 of 14 passages

Student passage FLAGGED

...The method first clear the text fields related to **the** basic information **of** the **teacher** such as **ID**, **name**, **address**, **working type**, **employment status**, and **working hours**

Top web match

Add a Tutor When this button is pressed, **the** input values **of teacher Id**, teacher **name**, **address**, **working type**, **employment status**, **working hours**, salary, specialization, academic qualifications and...

our GUI make it look more modern using awt | StudyX <https://studyx.ai/homework/100214603-our-gui-make-itlook-more-modern-using-awt-swing-should-contain-the-same-components-but>

11 of 14 passages

Student passage FLAGGED

A syntax error occurs **when the rules of programming** languages **are not followed correctly** in code. This means that the code does not...

Top web match

Answer: **A syntax error** is a mistake in a computer program or code **when the rules of the programming language are not followed correctly**. To fix syntax errors, a programmer must identify and correct...

Problem 14 What is a syntax error?... [FREE SOLUTION] - Vaia <https://www.vaia.com/en-us/textbooks/computer-science/starting-out-with-java-from-control-structures-through-data-structures-2edition/chapter-1/problem-14-what-is-a-syntax-error/>

12 of 14 passages

Student passage FLAGGED

Logical Error Logical error are those errors **that occurs when the code compiles and runs without any syntax error**, **but it does not produce the expected or required output**

Top web match

Logical error or wrong answer (wa) A **logical error** is an **error** in a program **that occurs when the code compiles and runs without** producing **any error** messages, **but it does not produce the expected or...**

Logical Error or Wrong Answer (WA) - CodeChef <https://www.codechef.com/learn/course/stacks-andqueues/LEARNJS14/problems/LEARNJSP86>

13 of 14 passages

Student passage FLAGGED

Runtime error A runtime error, also known as an exception, occurs **when a program is running and encounters an unexpected condition that** prevents it from continuing its normal execution. Unlike **syntax...**

Top web match

Runtime error Runtime errors in Java occur **when a program is running and encounters an unexpected condition** or action **that** cannot be handled by the program... **Syntax errors** occur when there is a mistake...

Types of Error in Java - Naukri Code 360 <https://www.naukri.com/code360/library/types-of-error-in-java>

14 of 14 passages

Student passage FLAGGED

...runtime errors occur during the execution of the program. **Some** of the **common runtime errors are division by zero, null pointer exception, array index out of bounds**, type casting error, **file**

Top web match

Some common examples of **runtime errors are: Division by zero; Null pointer** dereferencing; **Array index out of bounds**; Resource leaks (e.g., memory or open **file** ...

Common Errors in C Programming: Types, Meaning & Solutions <https://www.studysmarter.co.uk/explanations/computer-science/computer-programming/commonerrors-in-c-programming/>
