CS4001NI Programming

30% Individual Coursework

2023-24 Autumn

Student Name: Bibek Kumar Sahani

London Met ID: 23047566

College ID: NP01NT4A230229

Group: N8

Assignment Due Date: Friday, May 10, 2024

Assignment Submission Date: Sunday, May 5, 2024

Table of Contents

Table of Figures

Table of Tables

Introduction

1.1 Overview of the coursework

The overall goal of this coursework is to create a Teacher Graphic Interface User (GUI) that stores Teacher data in array list. This project is designed to know or view the information of teacher, lecturer, and tutor which information stored in array list. It is created under the reference of BlueJ. BlueJ is an integrated development environment (IDE) for the java programming language. BlueJ implements blue environment design for java programming. BlueJ runs with Java Development Kit (JDK). (geeksforgeeks, 2022)

1.2 Aim and Objective

The aim of the coursework is to extend the functionality of an existing java project by adding Graphical User Interface (GUI). The objectives of this coursework are listed below: -

1. Develop a Graphical User Interface (GUI) for a system that will store details of teachers in array list in new class with main method and class name called "TeacherGUI".
2. Utilize the text fields and buttons for better user experience.
3. To describe and check the functionality of the GUI components like buttons.
4. To create a report focusing on TeacherGUI class.
5. To provide the evidence of testing conducted on program.
6. To discuss error detection and correction.

1.3 About the coursework

The TeacherGUI class is like a basic form with labels Teacher Id, Teacher Name, Address, Working Type, Employment Status, Working Hours, Department, Years of Experience, Graded Score, and buttons Add, Display, clear, Grade assignment, Tutor in Lecturer class GUI. There is also labels and buttons for Tutor class Teacher Id, Teacher Name, Address, Working Type, Employment Status, Working Hours, salary, specialization, Academic Qualifications, Performance Index, and a certification status. It also has buttons Add, Display, clear, Lecturer, Set salary etc. In this GUI we have added some colour code in RGB format for better viewing experience of the user.

1.4 Tools Used

1. BlueJ: It is a user-friendly IDE (Integrated Development Environment) designed for learning programming in the context of JAVA. It is suitable of beginners in programming. It is widely used for educational purpose and small software developments. (geeksforgeeks, 2022)

Figure 1: BlueJ logo and interface

2. MS-Word: It is inbuilt user-friendly software tool of MS-Office for PC that is used for word processing applications, offering rich text editing, formatting tools and some extra features. (byjus, 2024)

Figure 2: Microsoft-word logo

3. Draw.io: It is an online diagramming tool having user-friendly interface which is used for creating flowcharts, different diagrams, and visualization. It supports collaborative work and exports to various formats. (Paraschiv, 2023)

dFigure 3: Draw.io Logo

4. Balsamiq: Balsamiq is a wireframing used for mock-ups and wireframes for websites, web apps, and desktop software. This wireframe tool created using Balsamiq are hand-drawn style. It focuses on structure and content of the product. (intellipaat, 2024)

balsamiq®

Figure 4: Balsamiq Logo

Wireframe

2.1 Lecturer Wireframe

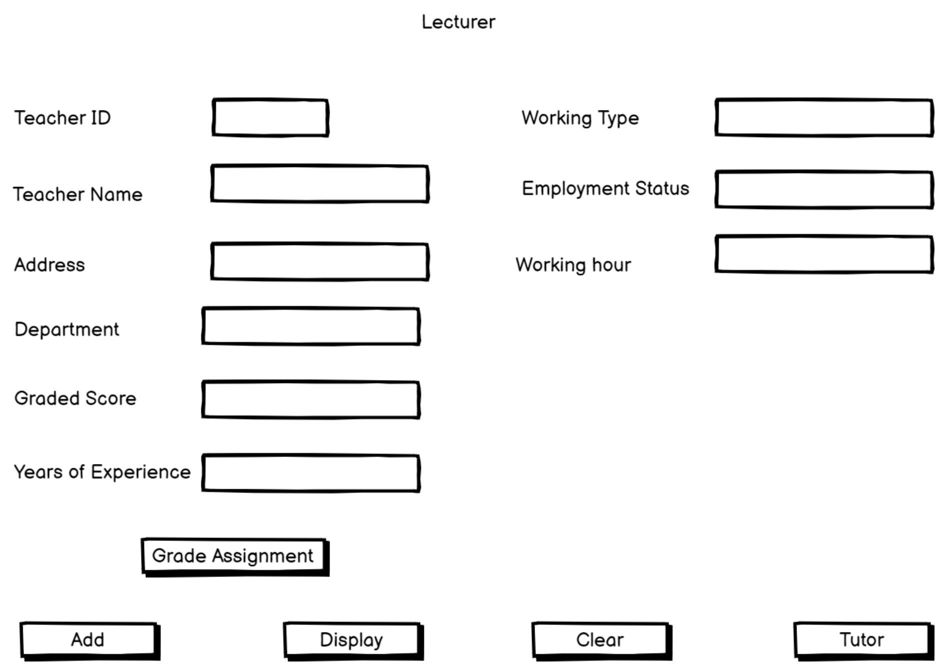


Figure 5: Lecturer Wireframe

2.2 Tutor Wireframe

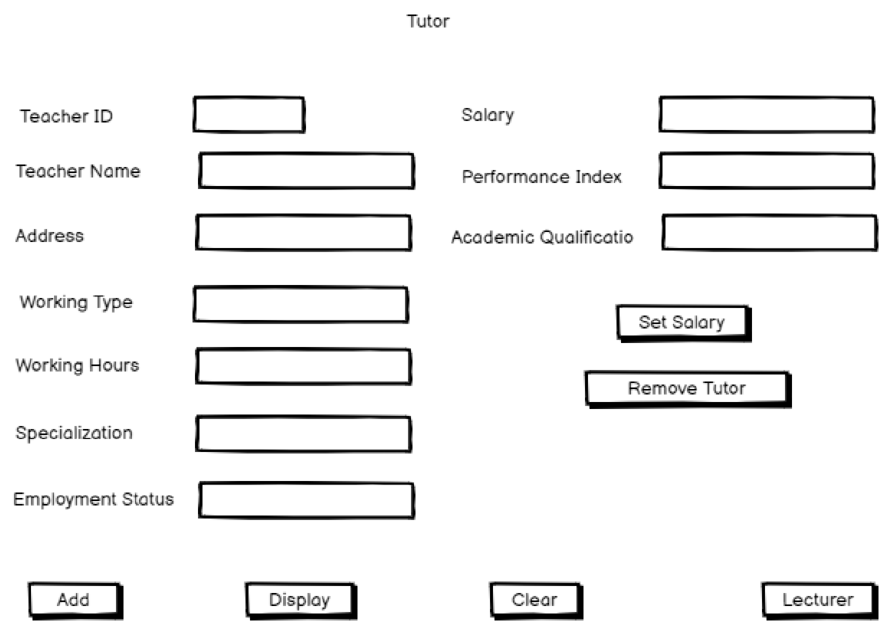


Figure 6: Tutor Wireframe

Class Diagram

3.1 Teacher Class Diagram

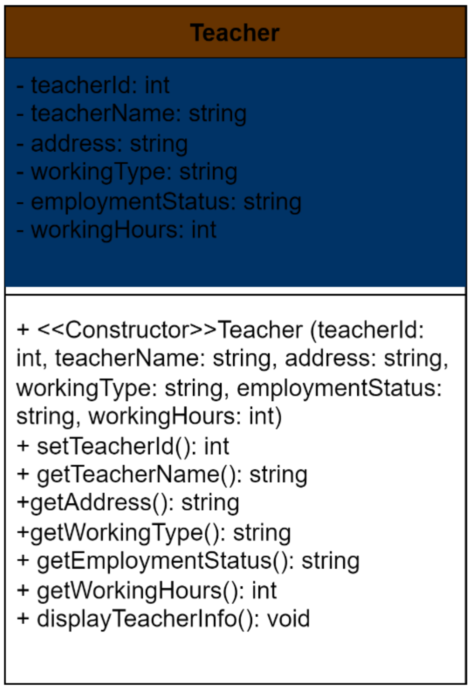

Teacher
- teacherId: int
- teacherName: string
- address: string
- workingType: string
- employmentStatus: string
- workingHours: int
+ <<Constructor>>Teacher (teacherId: int, teacherName: string, address: string, workingType: string, employmentStatus: string, workingHours: int)
+ setTeacherId(): int
+ getTeacherName(): string
+getAddress(): string
+getWorkingType(): string
+ getEmploymentStatus(): string
+ getWorkingHours(): int
+ displayTeacherInfo(): void

Figure 7: Teacher class Diagram

3.2 Tutor Class Diagram

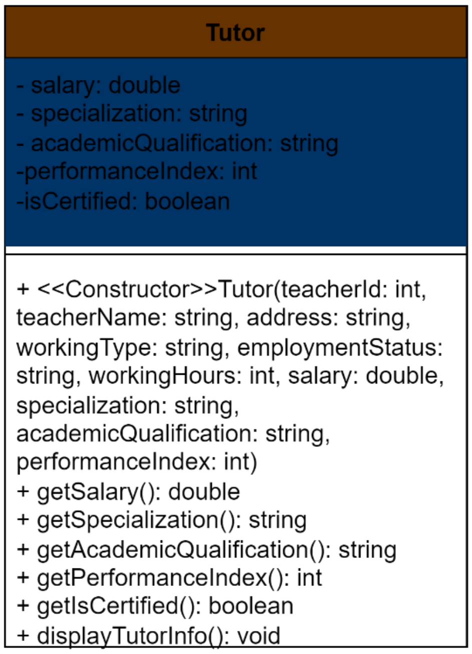

Tutor
- salary: double
- specialization: string
- academicQualification: string
-performanceIndex: int
-isCertified: boolean
+ <<Constructor>>Tutor(teacherId: int, teacherName: string, address: string, workingType: string, employmentStatus: string, workingHours: int, salary: double, specialization: string, academicQualification: string, performanceIndex: int)
+ getSalary(): double
+ getSpecialization(): string
+ getAcademicQualification(): string
+ getPerformanceIndex(): int
+ getIsCertified(): boolean
+ displayTutorInfo(): void

Figure 8: Tutor Class Diagram

3.3 Lecturer Class Diagram

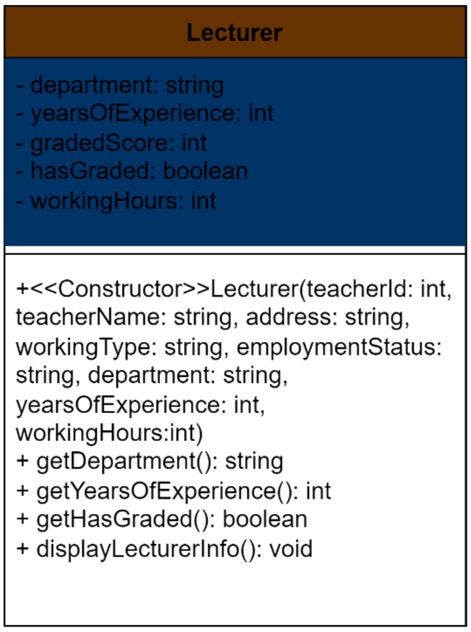

Lecturer
- department: string
- yearsOfExperience: int
- gradedScore: int
- hasGraded: boolean
- workingHours: int
+<<Constructor>>Lecturer(teacherId: int, teacherName: string, address: string, workingType: string, employmentStatus: string, department: string, yearsOfExperience: int, workingHours:int)
+ getDepartment(): string
+ getYearsOfExperience(): int
+ getHasGraded(): boolean
+ displayLecturerInfo(): void

Figure 9: Lecturer Class Diagram

3.4 Teacher GUI Class Diagram

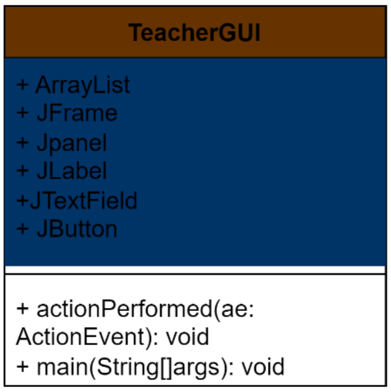

TeacherGUI
+ ArrayList
+ JFrame
+ Jpanel
+ JLabel
+JTextField
+ JButton
+ actionPerformed(ae: ActionEvent): void
+ main(String[]args): void

Figure 10: TeacherGUI Class Diagram

Inheritance Diagram

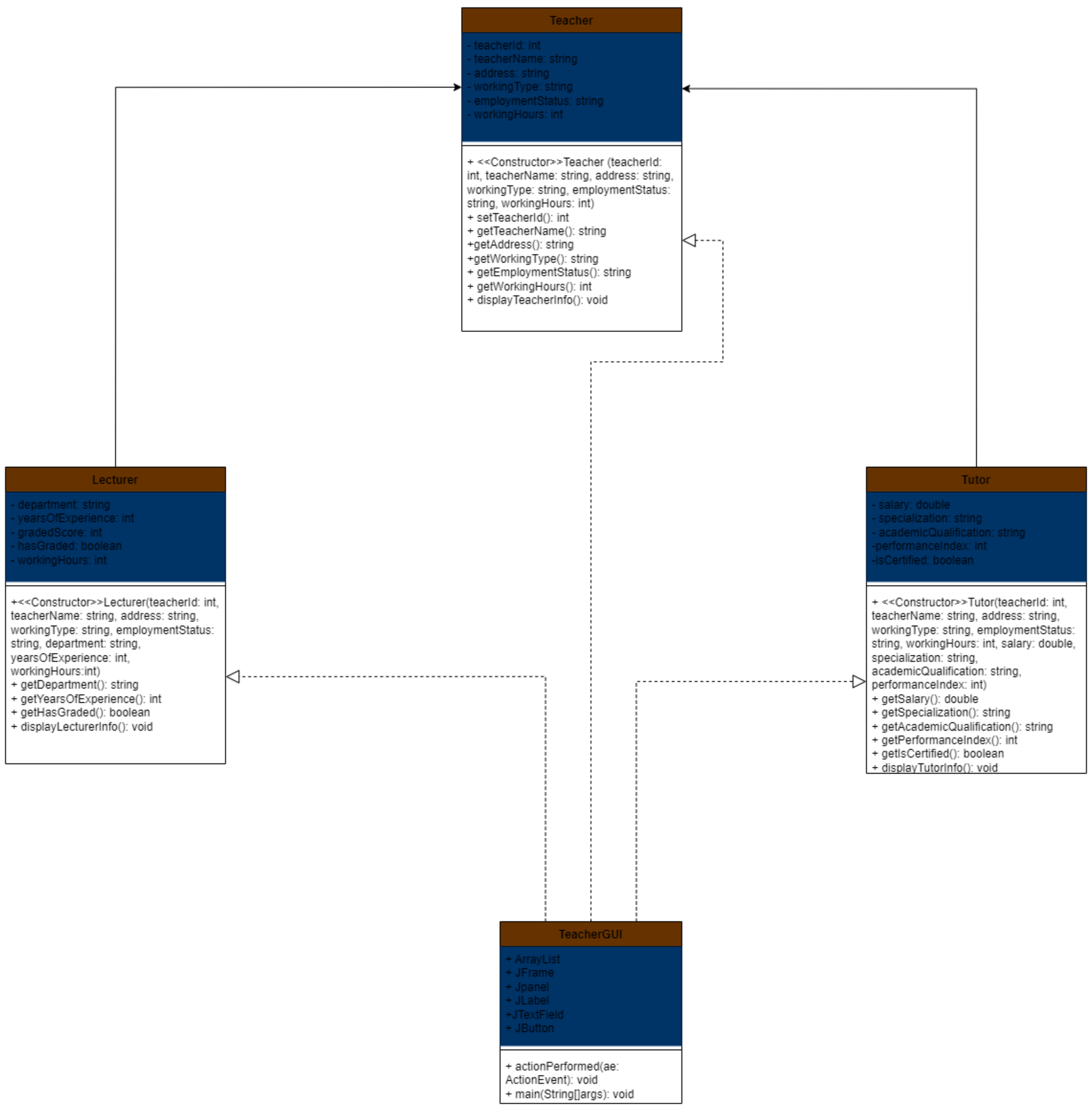

Teacher
- teacherId: int
- teacherName: string
- address: string
- workingType: string
- employmentStatus: string
- workingHours: int
+ <<Constructor>>Teacher (teacherId: int, teacherName: string, address: string, workingType: string, employmentStatus: string, workingHours: int)
+ setTeacherId(): int
+ getTeacherName(): string
+getAddress(): string
+getWorkingType(): string
+ getEmploymentStatus(): string
+ getWorkingHours(): int
+ displayTeacherInfo(): void
Lecturer
- department: string
- yearsOfExperience: int
- gradedScore: int
- hasGraded: boolean
- workingHours: int
+<<Constructor>>Lecturer(teacherId: int, teacherName: string, address: string, workingType: string, employmentStatus: string, department: string, yearsOfExperience: int, workingHours:int)
+ getDepartment(): string
+ getYearsOfExperience(): int
+ getHasGraded(): boolean
+ displayLecturerInfo(): void
Tutor
- salary: double
- specialization: string
- academicQualification: string
-performanceIndex: int
-isCertified: boolean
+ <<Constructor>>Tutor(teacherId: int, teacherName: string, address: string, workingType: string, employmentStatus: string, workingHours: int, salary: double, specialization: string, academicQualification: string, performanceIndex: int)
+ getSalary(): double
+ getSpecialization(): string
+ getAcademicQualification(): string
+ getPerformanceIndex(): int
+ getIsCertified(): boolean
+ displayTutorInfo(): void
TeacherGUI
+ ArrayList
+ JFrame
+ Jpanel
+ JLabel
+JTextField
+ JButton
+ actionPerformed(ae: ActionEvent): void
+ main(String[]args): void

Figure 11: Inheritance Diagram of Four classes

Pseudocode

```
Import java awt
Import java swing
CREATE class TeacherGUI
CREATE ArrayList A1

CREATE JFrame jf1
SET jf1 Title
SET jf1 Size
SET jf1 Layout
SET jf1 Resizable

CREATE JPanel jp1
SET jp1 Bounds
SET jp1 Layout
SET jp1 Background
ADD jp1 TO jf1

CREATE JLabel l1, l2, l3, l4, l5, l6, l7, l8, l9
CREATE JTextfields jtf2, jtf3, jtf4, jtf5, jtf6, jtf7, jtf8, jtf9
CREATE JButtons jb1, jb2, jb3, jb4, jb5

SET bounds of l1, l2, l3, l4, l5, l6, l7, l8, l9
SET bounds of jtf2, jtf3, jtf4, jtf5, jtf6, jtf7, jtf8, jtf9
SET bounds of jb1, jb2, jb3, jb4, jb5

ADD l1, l2, l3, l4, l5, l6, l7, l8, l9 TO jp1
ADD jtf2, jtf3, jtf4, jtf5, jtf6, jtf7, jtf8, jtf9 TO jp1
ADD jb1, jb2, jb3, jb4, jb5  TO jp1
ADD ActionListener TO jb1
    TRY
        GET Teacher Id
        GET Teacher Name
        GET Address
        GET Working Type
        GET Employment Status
        GET Years of Experience
        GET Department
        GET Working Hour
        CREATE Lecturer L1
        ADD L1 to A1
        SHOW Message “Lecturer added successfully!”
    CATCH Exception
        SHOW Message “Invalid Input!”
END ActionListener

ADD ActionListener TO jb2
    TRY
        GET Years of Experience
        GET Department
        GET Graded Score
        Loop through A1
        IF Teacher is instance of Lecturer, then
        DOWNCAST to Lecturer
        CALL L1 To Grade Assignment
        END IF
        SHOW Message “Grade Assigned successfully!”
    CATCH Exception
```

```
                SHOW Message "Invalid Input!"
END ActionListener

ADD ActionListener TO jb3
                Loop through A1
                IF Teacher is instance of Lecturer, then
                DOWNCAST to Lecturer
                CALL L1 To Display Lecturer Information
                END IF
END ActionListener

ADD ActionListener TO jb4
                SET Clear Text Field
                SHOW Message "Text Fields are Cleared!"
END ActionListener

ADD ActionListener TO jb5
        CREATE JPanel jp2
        SET jp2 Bounds
        SET jp2 Layout
        SET jp2 Background
        ADD jp2 TO jf1

        CREATE JLabel  l_1, l_2, l_3, l_4, l_5, l_6, l_7, l_8, l_9, l_10, l_11
        CREATE JTextfields jtf_2, jtf_3, jtf_4, jtf_5, jtf_6, jtf_7, jtf_8, jtf_9, jtf_10, jtf_11
        CREATE JButtons jb_1, jb_2, jb_3, jb_4, jb_5, jb_6

        SET bounds of l_1, l_2, l_3, l_4, l_5, l_6, l_7, l_8, l_9, l_10, l_11
        SET bounds of jtf_2, jtf_3, jtf_4, jtf_5, jtf_6, jtf_7, jtf_8, jtf_9, jtf_10, jtf_11
        SET bounds of jb_1, jb_2, jb_3, jb_4, jb_5, jb_6
```

```
    ADD l_1, l_2, l_3, l_4, l_5, l_6, l_7, l_8, l_9, l_10, l_11 TO jp2
    ADD jtf_2, jtf_3, jtf_4, jtf_5, jtf_6, jtf_7, jtf_8, jtf_9, jtf_10, jtf_11 TO jp2
    ADD jb_1, jb_2, jb_3, jb_4, jb_5, jb_6 TO jp2


ADD ActionListener TO jb_1
    TRY
        GET Teacher Id
        GET Teacher Name
        GET Address
        GET Working Type
        GET Employment Status
        GET Performance Index
        GET Academic Qualification
        GET Working Hour
        GET Salary
        GET Specialization
        CREATE Tutor T1
        ADD T1 to A1
        SHOW Message “Tutor added successfully!”
    CATCH Exception
        SHOW Message “Invalid Input!”
END ActionListener

ADD ActionListener TO jb_2
    TRY
        GET Salary
        GET Performance Index
        Loop through A1
        IF Teacher is instance of Tutor, then
        DOWNCAST to Tutor
        CALL T0 To SET Salary and Performance Index
        END IF
        SHOW Message "Salary and performance index have been set
successfully!"
    CATCH Exception
        SHOW Message "Invalid Input!"
END ActionListener

ADD ActionListener TO jb_3
        Loop through A1
        IF Teacher is instance of Tutor, then
        DOWNCAST to Tutor
        CALL T2 To Remove Tutor
        END IF
        SHOW Message "Tutor has been removed successfully!"
        BREAK
END ActionListener
ADD ActionListener TO jb_4
        Loop through A1
        IF Teacher is instance of Tutor, then
        DOWNCAST to Tutor
        CALL T2 To Display Tutor Information
        END IF
END ActionListener

ADD ActionListener TO jb_5
        SET Clear Text Field
        SHOW Message "Text Fields are Cleared!"
END ActionListener
```

```
ADD ActionListener TO jb_6
        GET jf1 Content Pane
        REMOVE jp2
        ADD jp1 TO jf1
        SET Revalidate
        SET Repaint
END ActionListener

SET jf1 Visible
END ActionListener
```

Method Description

1. Main method: The Java main method is the first method to learn because main method is the entry point for executing a java program. The main method contains code to execute or call other methods. The main method is written in this way “public static void main (String[]args)”

2. Add Lecturer Button (jb1): The button functionality is to add new lecturer to Array list. It is associated with ActionListener which captures the data entered in the text filed and create new lecture object and store it in Array list.

3. Grade Assignment Button (jb2): The button functionality is to grades an assignment for a lecturer. It is associated with ActionListener which retrieves the necessary data of grading assignment from text field and find the corresponding lecturer in the Array list and grade the assignment from the lecturer.

4. Display button (jb3): The button functionality is to display information about the lecturers. It is associated with ActionListener. It identifies the lecturers and display their information.

5. Clear Button (jb4): The button functionality is to clear all the text fields. It is associated with ActionListener which resets all the text field into empty string.

6. Tutor Button (jb5): The button functionality is to switch the lecturer interface to tutor interface. It is associated with ActionListener. It allows the user to change the JPanel between to classes.

7. Add Tutor Button (jb_1): The button functionality is to add new Tutor to Array list. It is associated with ActionListener which captures the data entered in the text filed and create new Tutor object and store it in Array list.

8. Set salary (jb_2): The button functionality is to set the salary and performance index for tutor. It is associated with ActionListener which retrieves the necessary data of Tutor ID from text field and find the corresponding Tutor in the Array list and set the salary and Performance index.

9. Remove Tutor Button (jb_3): The button functionality is to Remove Tutor from the system. It is associated with ActionListener. It retrieves the tutor ID from the text field and find the corresponding tutor in Array List and removes the tutor.

10. Display Tutor Button (jb_4): The button functionality is to display information about the Tutor. It is associated with ActionListener. It identifies the Tutor and display their information.

11. Clear Tutor (jb_5): The button functionality is to clear all the text fields. It is associated with ActionListener which resets all the text field into empty string.

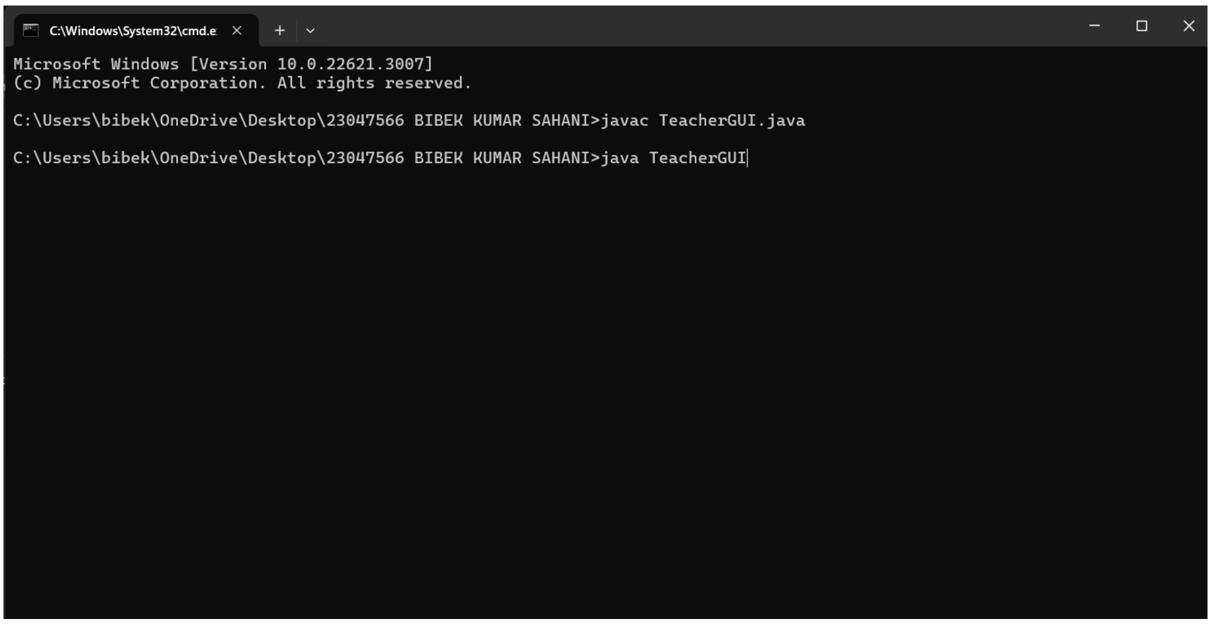
12. Lecturer Button (jb_6): The button functionality is to switch the Tutor interface to Lecturer interface. It is associated with ActionListener. It allows the user to change the JPanel between to classes.

Testing

Table 1: Testing 1

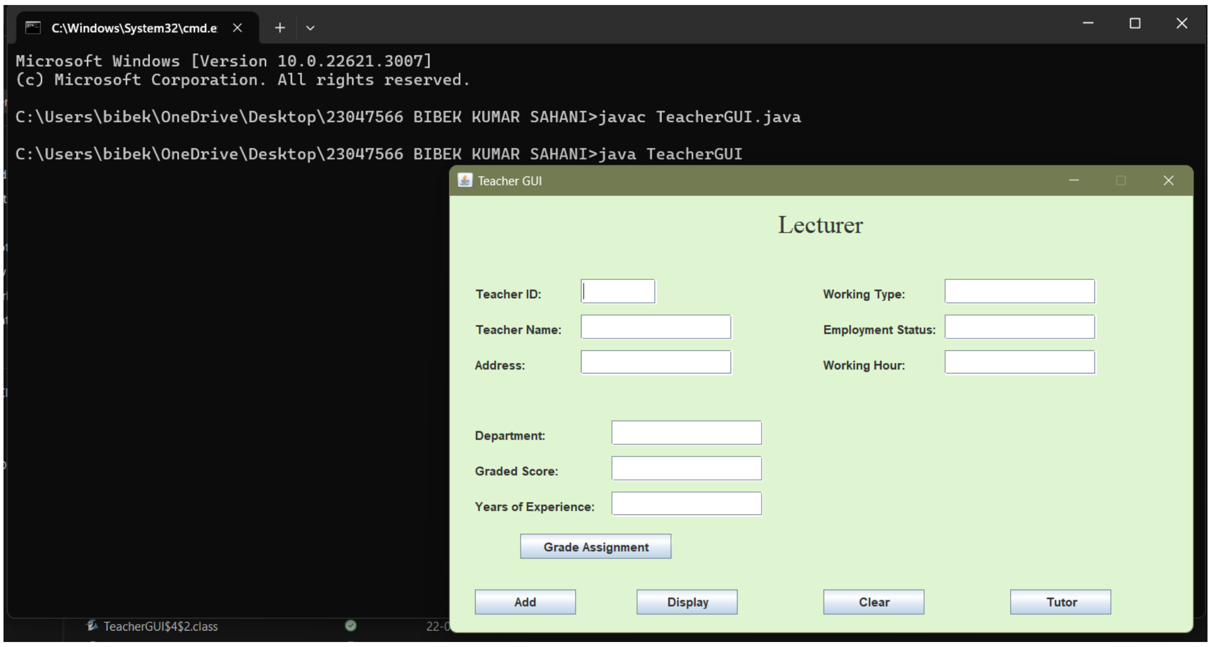
Objective	To test the program can be compiled and run using command prompt.
Action	Open the command prompt and type “javac TeacherGUI.java” then press enter after that type “java TeacherGUI” and press enter.
Expected Result	The GUI of TeacherGUI will be shown.
Actual Result	The TeacherGUI was shown.
Conclusion	The test was successful.

Screenshot of Testing 1



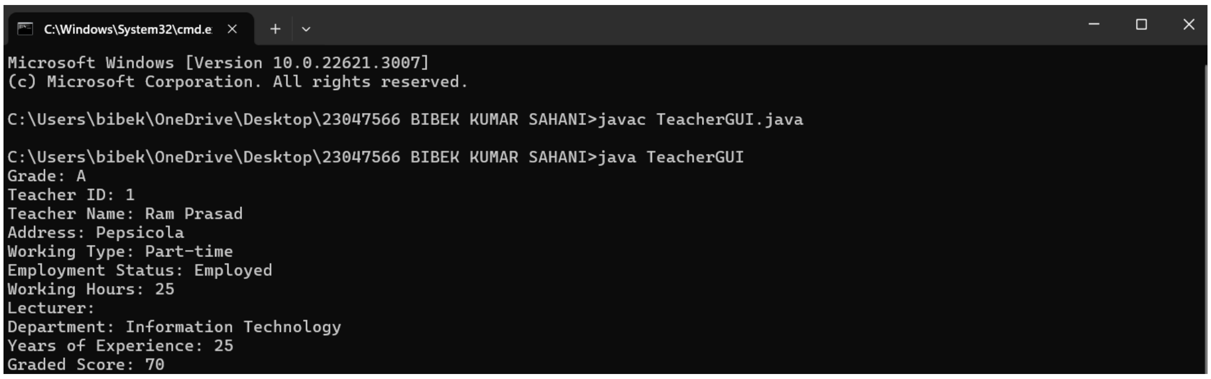
```
Microsoft Windows [Version 10.0.22621.3007]
(c) Microsoft Corporation. All rights reserved.

C:\Users\bibek\OneDrive\Desktop\23047566 BIBEK KUMAR SAHANI>javac TeacherGUI.java

C:\Users\bibek\OneDrive\Desktop\23047566 BIBEK KUMAR SAHANI>java TeacherGUI
```

Figure 12: Testing 1 Command prompt



```
Microsoft Windows [Version 10.0.22621.3007]
(c) Microsoft Corporation. All rights reserved.

C:\Users\bibek\OneDrive\Desktop\23047566 BIBEK KUMAR SAHANI>javac TeacherGUI.java

C:\Users\bibek\OneDrive\Desktop\23047566 BIBEK KUMAR SAHANI>java TeacherGUI
```

Figure 13: Open Teacher GUI from command prompt



```
Microsoft Windows [Version 10.0.22621.3007]
(c) Microsoft Corporation. All rights reserved.

C:\Users\bibek\OneDrive\Desktop\23047566 BIBEK KUMAR SAHANI>javac TeacherGUI.java

C:\Users\bibek\OneDrive\Desktop\23047566 BIBEK KUMAR SAHANI>java TeacherGUI
Grade: A
Teacher ID: 1
Teacher Name: Ram Prasad
Address: Pepsicola
Working Type: Part-time
Employment Status: Employed
Working Hours: 25
Lecturer:
Department: Information Technology
Years of Experience: 25
Graded Score: 70
```

Figure 14: Displaying the Lecturer information in command promt

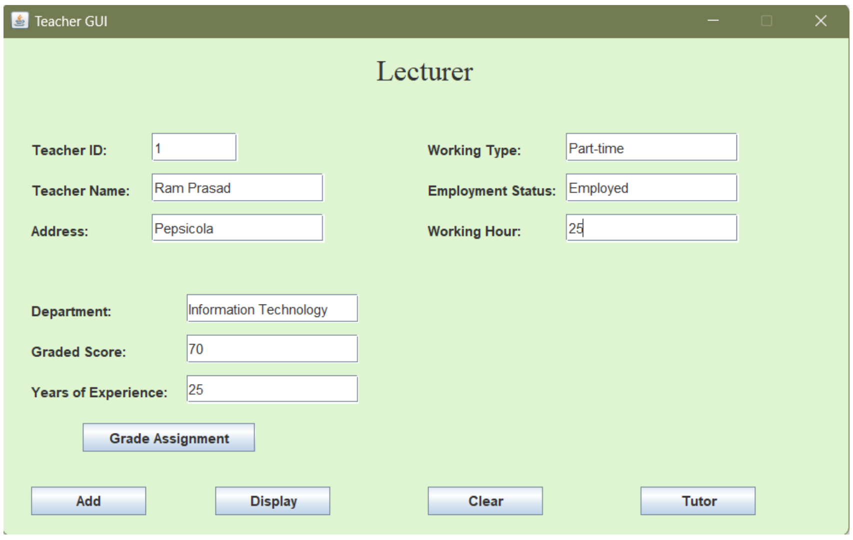
Table 2 : Testing 2

Objective	To add Lecturer, Tutor, Grade Assignment from Lecturer, Set salary, Remove tutor.
Action	a) Assign the value of Lecturer in field box and add by pressing "Add" button. The values assigned are: Teacher ID: 1 Teacher Name: "Ram Prasad" Address: "Pepsicola" Department: "Information Technology" Graded Score: 70 Years of Experience: 25 Working Type: "Part-time" Employment Status: "Employed" Working Hour: 25

	b) Assign the value of Tutor in field box and add by pressing “Add” button. The assigned values are: Teacher ID: 1 Teacher Name: “Shyam Prasad” Address: “Rammechhap” Working Type: “Full-time” Working Hours: 24 Specialization: “IT” Employment Status: ”Employed” Salary: 50000 Performance Index: 8 Academic Qualification: ”Masters in IT” c) After assigning the value of Lecturer press the button “Grade Assignment”. d) After assigning the value of Tutor press the button “Set Salary”. e) After assigning and setting the salary press the button “Remove Tutor”.
Expected Result	a) While pressing the add in Lecturer the message box will appeared displaying “Lecturer added successfully!” then after pressing “Display” button, the Input enter in text field will be displayed. b) While pressing the add button in Tutor the message box will appeared displaying “Tutor added successfully!” then after pressing “Display” button, the input enter in the text field will be displayed. c) While pressing the “Grade Assignment” button the grade should be displayed. d) While pressing the “Set Salary” button the salary should be either approved or not approved. e) While pressing the “Remove Tutor” button the salary which has been set should be removed.
Actual Result	a) The Input value had been displayed successfully after pressing “Display” button in Lecturer. b) The Input value had been displayed successfully after pressing “Display” button in Tutor. c) The grade was assigned successfully. d) The salary was not approved by tutor. e)
Conclusion	The test has been passed.

Screenshots of Testing 2

For a

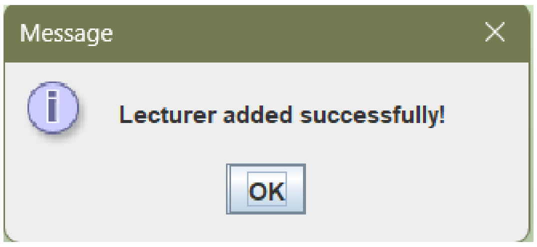

Teacher GUI

Lecturer

Teacher ID:	1	Working Type:	Part-time
Teacher Name:	Ram Prasad	Employment Status:	Employed
Address:	Pepsicola	Working Hour:	25

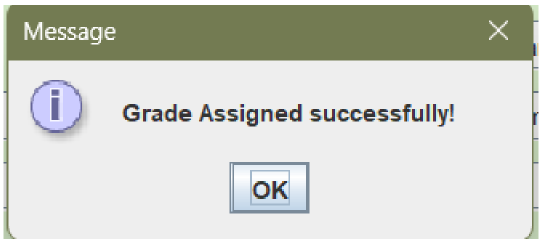
Department:	Information Technology
Graded Score:	70
Years of Experience:	25

Figure 15: Assigned Input for Lecturer


Message

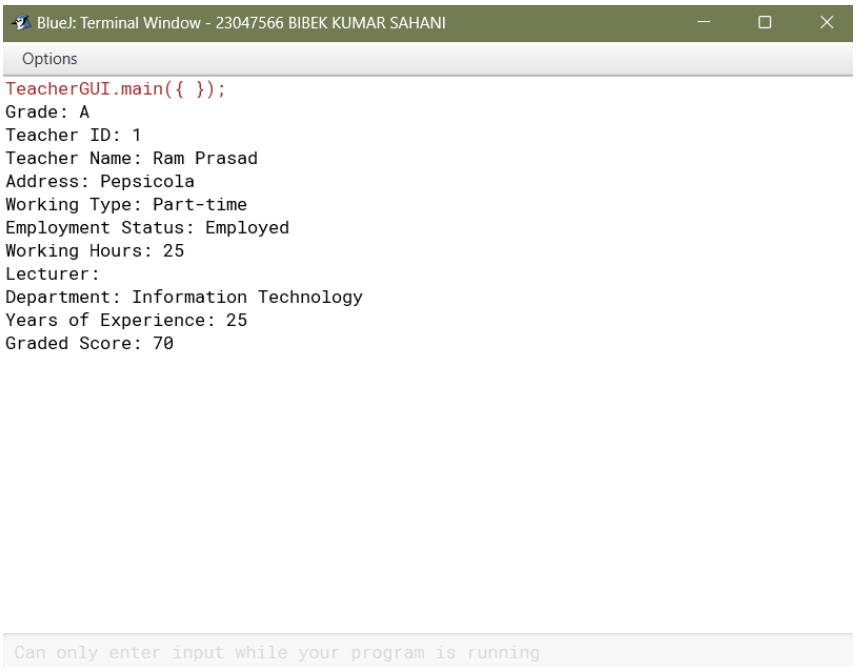
Lecturer added successfully!

Figure 16: Message for confirmation


Message

Grade Assigned successfully!

Figure 17: Message for confirmation of Grade assigned

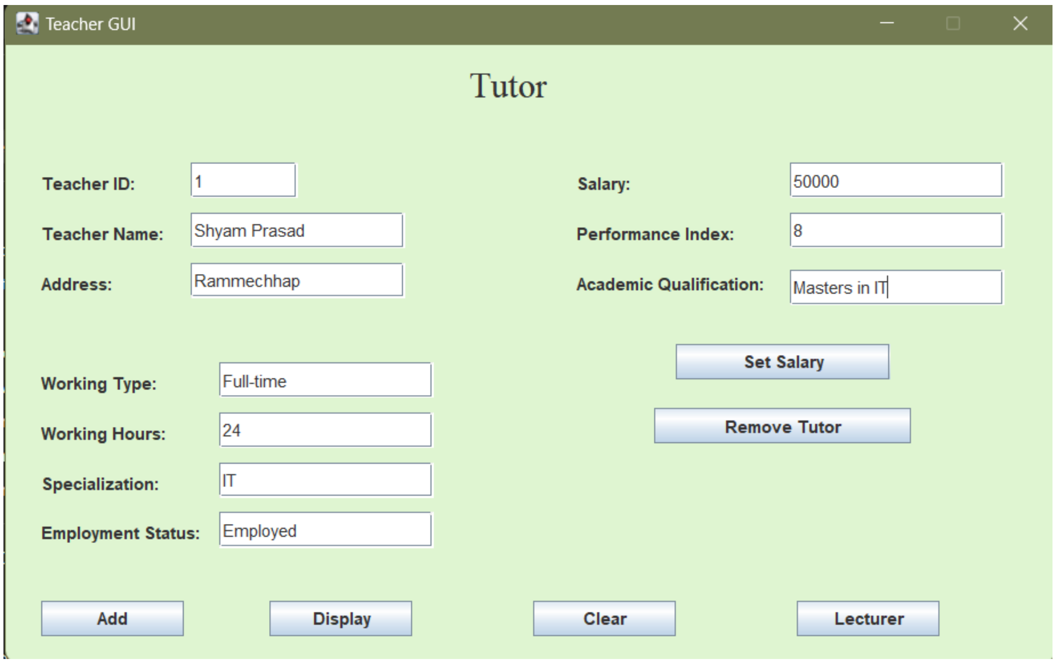

BlueJ: Terminal Window - 23047566 BIBEK KUMAR SAHANI
Options

```
TeacherGUI.main({ });
Grade: A
Teacher ID: 1
Teacher Name: Ram Prasad
Address: Pepsicola
Working Type: Part-time
Employment Status: Employed
Working Hours: 25
Lecturer:
Department: Information Technology
Years of Experience: 25
Graded Score: 70
```

Can only enter input while your program is running

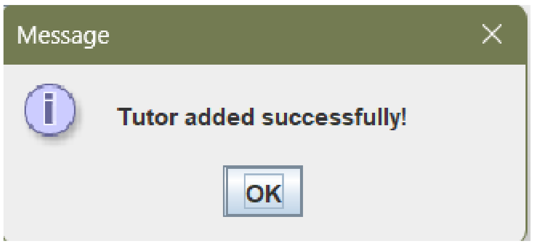
Figure 18: Displaying the assigned Lecturer

For b



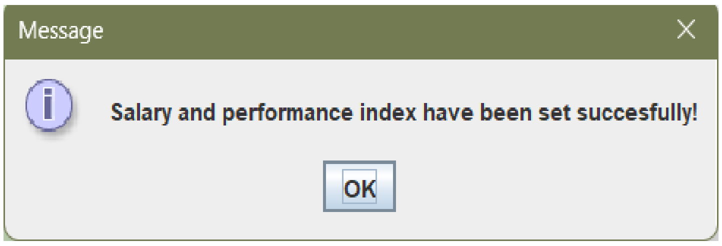
Teacher ID:	1	Salary:	50000
Teacher Name:	Shyam Prasad	Performance Index:	8
Address:	Rammechhap	Academic Qualification:	Masters in IT
Working Type:	Full-time		
Working Hours:	24		
Specialization:	IT		
Employment Status:	Employed		

Figure 19: Assigned Value of Tutor in Text field



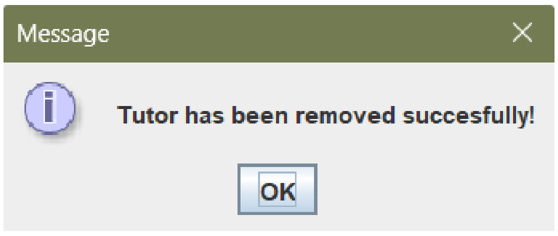
Tutor added successfully!

Figure 20: Message box for confirmation



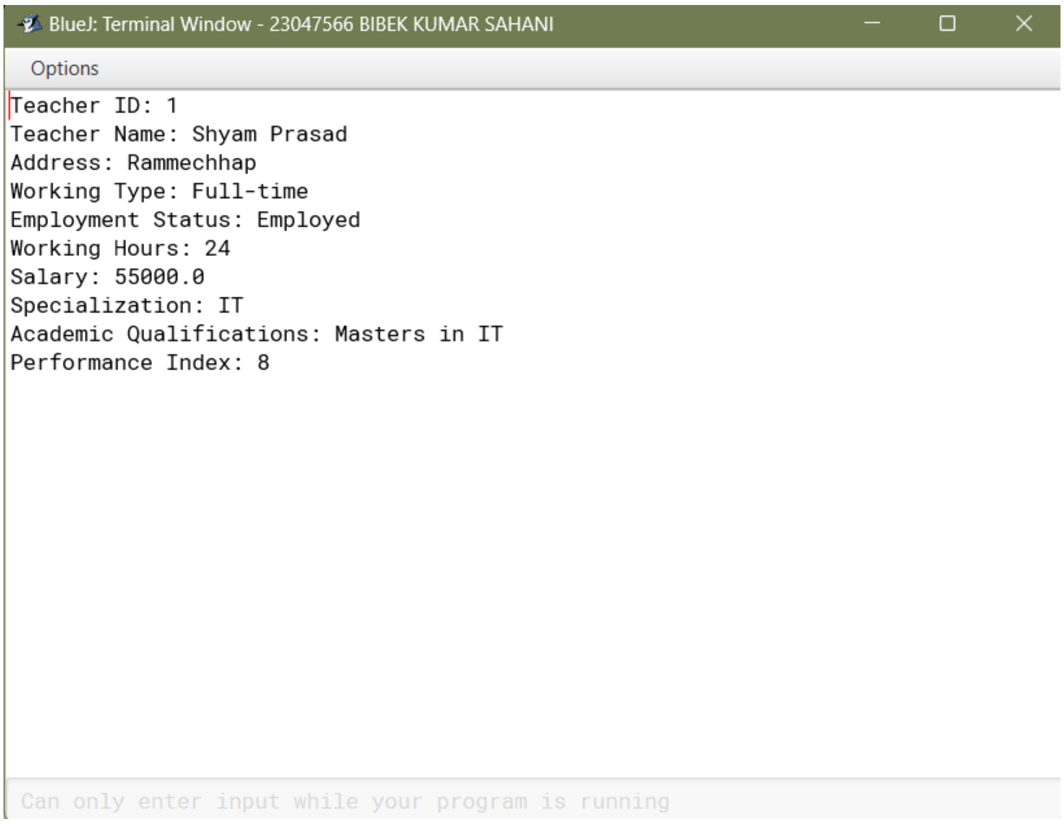
Salary and performance index have been set succesfully!

Figure 21: Message Box for confirmation of Set Salary and Performance Index



Tutor has been removed succesfully!

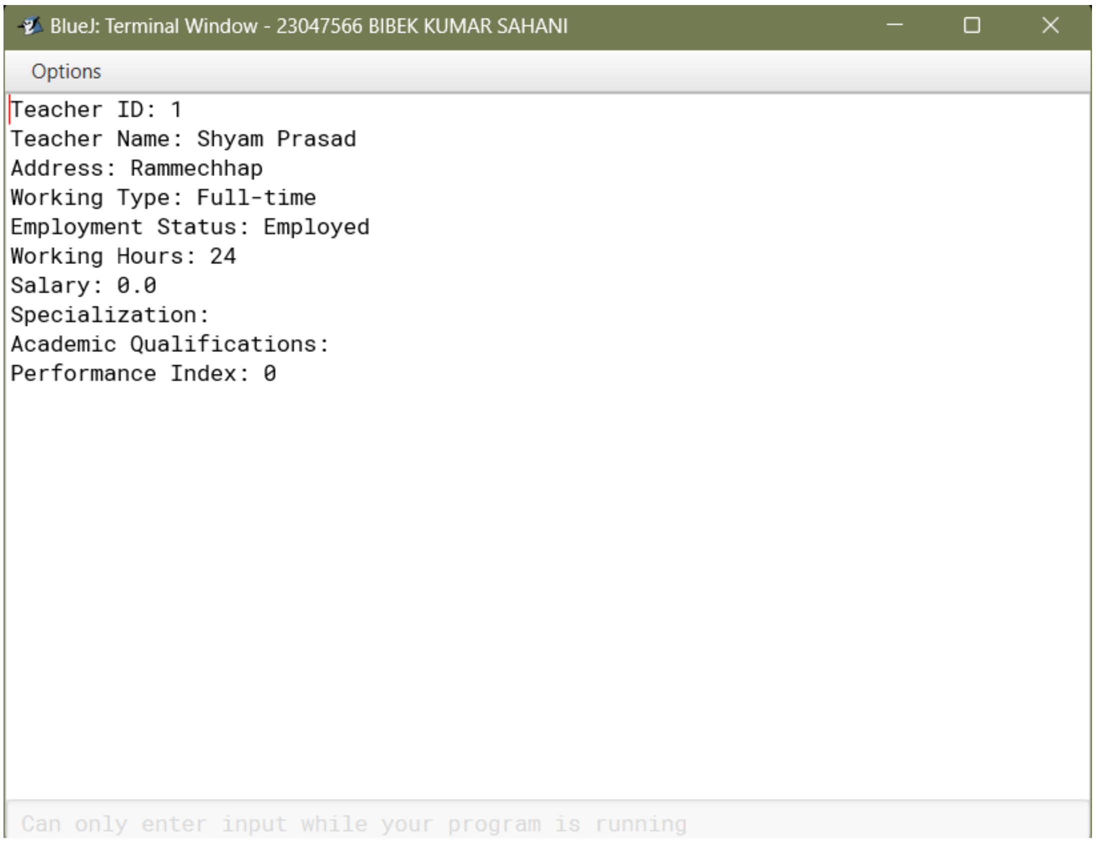
Figure 22: Message box for confirmation of removing Tutor


BlueJ: Terminal Window - 23047566 BIBEK KUMAR SAHANI
Options

```
Teacher ID: 1
Teacher Name: Shyam Prasad
Address: Rammechhap
Working Type: Full-time
Employment Status: Employed
Working Hours: 24
Salary: 55000.0
Specialization: IT
Academic Qualifications: Masters in IT
Performance Index: 8
```

Can only enter input while your program is running

Figure 23: Displaying the Tutor Info


BlueJ: Terminal Window - 23047566 BIBEK KUMAR SAHANI
Options

```
Teacher ID: 1
Teacher Name: Shyam Prasad
Address: Rammechhap
Working Type: Full-time
Employment Status: Employed
Working Hours: 24
Salary: 0.0
Specialization:
Academic Qualifications:
Performance Index: 0
```

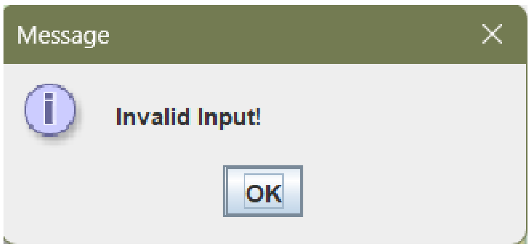
Can only enter input while your program is running

Figure 24: Displaying after Removing Tutor Info

Table 3: Testing 3

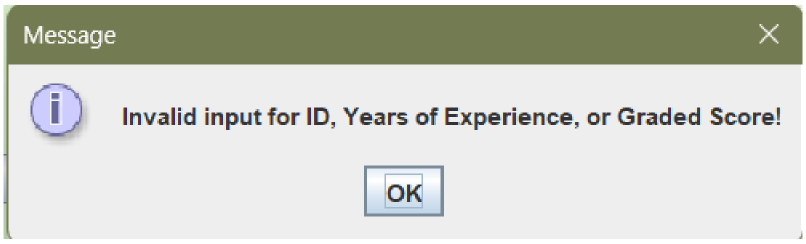
Objective	To test the dialog box, appear when unsuitable values are entered for Teacher ID.
Action	a) Leave the Text Field blank and press "Add" button. b) Write String value in Teacher Id and press "Add" button. c) Leave the Text Field blank of Grade Assignment Parameters and press "Grade Assignment" button. d) Leave The Text Field blank in Set Salary Parameters and press "Set Salary" buttons.
Expected Result	a) The Dialog box will appear saying "Invalid Input!". b) The Dialog box will appear saying "Invalid Input!". c) The Dialog box will appear saying "Invalid Input for ID, Years of Experience and Graded Score!". d) The Dialog box will appear saying "Invalid Input for Salary and Performance Index!".
Actual Result	The Dialog box was appeared with message "Please enter a valid Input!".
Conclusion	The test has been successfully passed.

Screenshot of Testing 3


Message

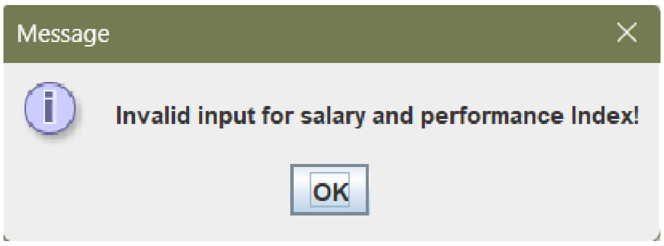
Invalid Input!

Figure 25: Message box of Error Detection


Message

Invalid input for ID, Years of Experience, or Graded Score!

Figure 26: Message box of Error detection in Grade assignment


Message

Invalid input for salary and performance Index!

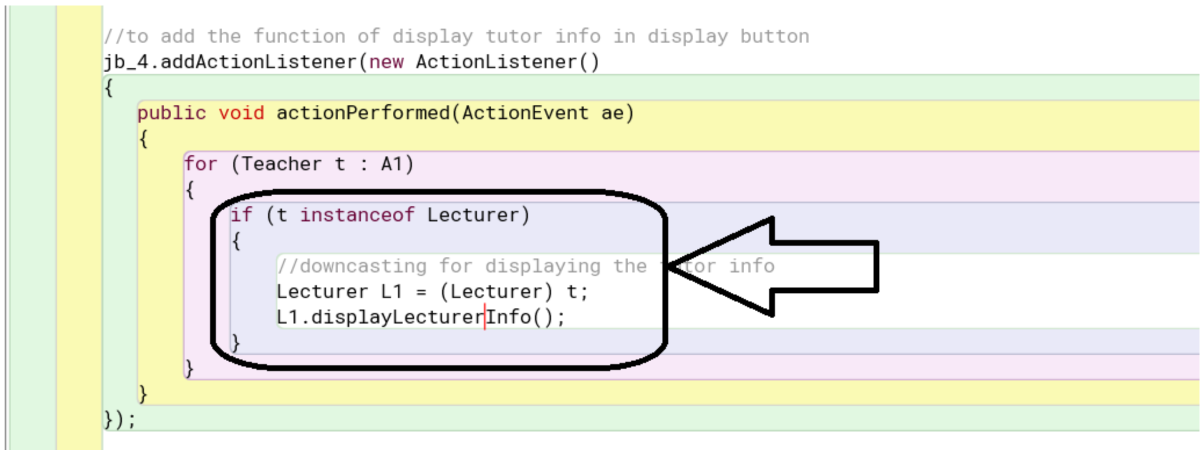
Figure 27: Message box of Error detection in Set salary

Error Detection

6.1 Logical Error

There was a logical error in tutor jb_4 button. While displaying the Tutor info the button was not working.

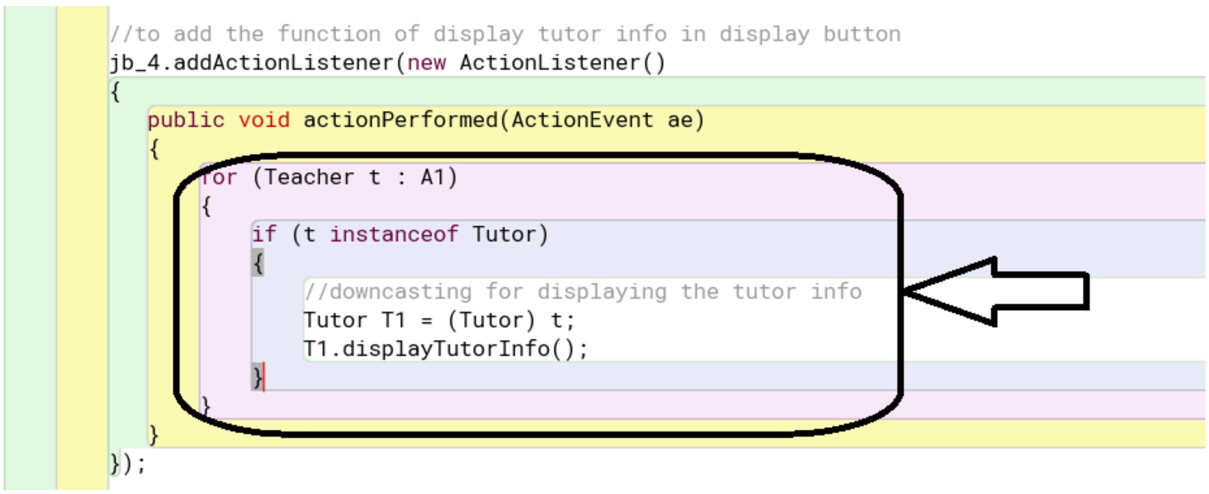
Before:



```
//to add the function of display tutor info in display button
jb_4.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent ae)
    {
        for (Teacher t : A1)
        {
            if (t instanceof Lecturer)
            {
                //downcasting for displaying the tutor info
                Lecturer L1 = (Lecturer) t;
                L1.displayLecturerInfo();
            }
        }
    }
});
```

Figure 28: Logical Error finding

After:



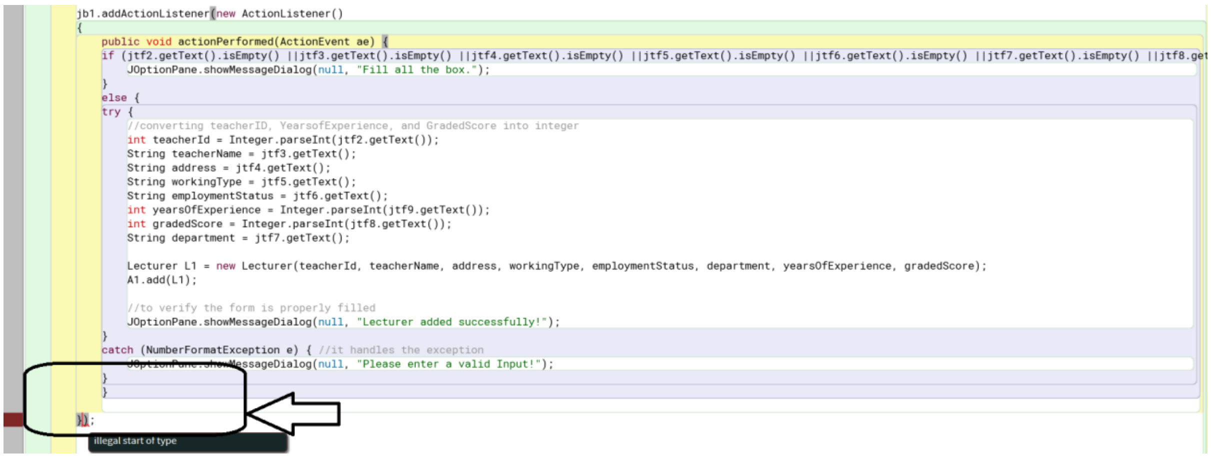
```
//to add the function of display tutor info in display button
jb_4.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent ae)
    {
        for (Teacher t : A1)
        {
            if (t instanceof Tutor)
            {
                //downcasting for displaying the tutor info
                Tutor T1 = (Tutor) t;
                T1.displayTutorInfo();
            }
        }
    }
});
```

Figure 29: Logical Error correction

6.2 Syntax Error:

There was a curly bracket missing in button b1 while using if else condition. While compiling the code there was error shown “illegal start of type”.

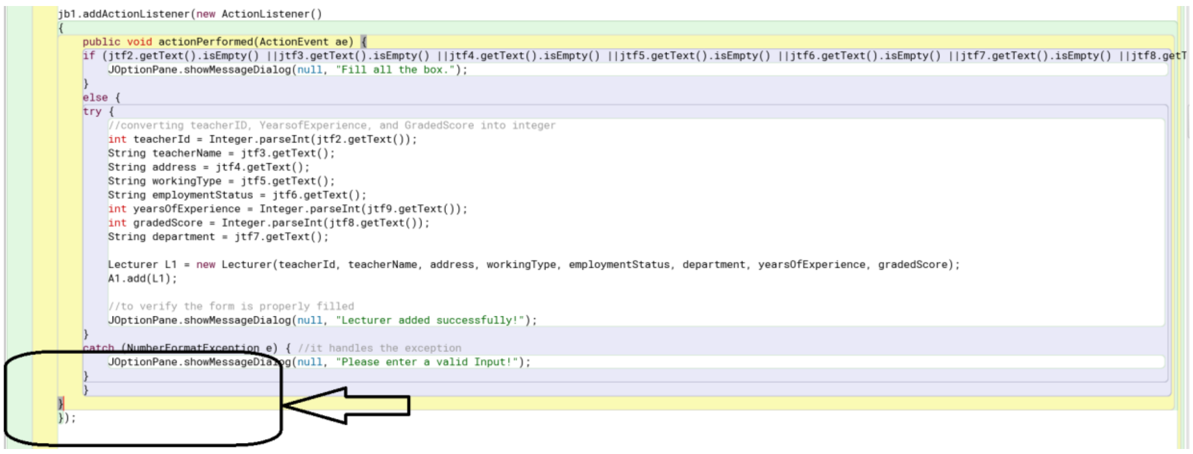
Before:



```
jb1.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent ae) {
    if (jtf2.getText().isEmpty() ||jtf3.getText().isEmpty() ||jtf4.getText().isEmpty() ||jtf5.getText().isEmpty() ||jtf6.getText().isEmpty() ||jtf7.getText().isEmpty() ||jtf8.get
        JOptionPane.showMessageDialog(null, "Fill all the box.");
    }
    else {
    try {
        //converting teacherID, YearsofExperience, and GradedScore into integer
        int teacherId = Integer.parseInt(jtf2.getText());
        String teacherName = jtf3.getText();
        String address = jtf4.getText();
        String workingType = jtf5.getText();
        String employmentStatus = jtf6.getText();
        int yearsOfExperience = Integer.parseInt(jtf9.getText());
        int gradedScore = Integer.parseInt(jtf8.getText());
        String department = jtf7.getText();

        Lecturer L1 = new Lecturer(teacherId, teacherName, address, workingType, employmentStatus, department, yearsOfExperience, gradedScore);
        A1.add(L1);

        //to verify the form is properly filled
        JOptionPane.showMessageDialog(null, "Lecturer added successfully!");
    }
    catch (NumberFormatException e) { //it handles the exception
        JOptionPane.showMessageDialog(null, "Please enter a valid Input!");
    }
    }

});
```

Figure 30: Syntax Error finding.

After:



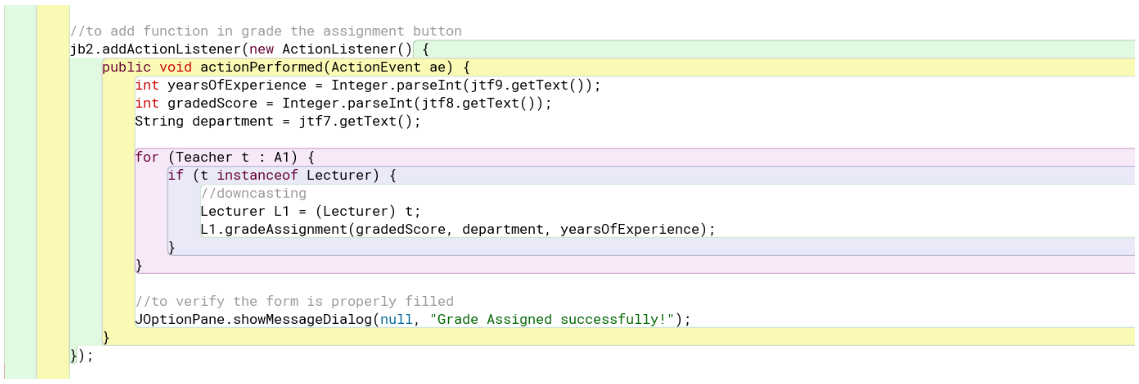
```
jb1.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent ae) {
    if (jtf2.getText().isEmpty() ||jtf3.getText().isEmpty() ||jtf4.getText().isEmpty() ||jtf5.getText().isEmpty() ||jtf6.getText().isEmpty() ||jtf7.getText().isEmpty() ||jtf8.getT
        JOptionPane.showMessageDialog(null, "Fill all the box.");
    }
    else {
    try {
        //converting teacherID, YearsofExperience, and GradedScore into integer
        int teacherId = Integer.parseInt(jtf2.getText());
        String teacherName = jtf3.getText();
        String address = jtf4.getText();
        String workingType = jtf5.getText();
        String employmentStatus = jtf6.getText();
        int yearsOfExperience = Integer.parseInt(jtf9.getText());
        int gradedScore = Integer.parseInt(jtf8.getText());
        String department = jtf7.getText();

        Lecturer L1 = new Lecturer(teacherId, teacherName, address, workingType, employmentStatus, department, yearsOfExperience, gradedScore);
        A1.add(L1);

        //to verify the form is properly filled
        JOptionPane.showMessageDialog(null, "Lecturer added successfully!");
    }
    catch (NumberFormatException e) { //it handles the exception
        JOptionPane.showMessageDialog(null, "Please enter a valid Input!");
    }
    }
}
});
```

Figure 31: syntax error after correction

6.3 Runtime Error

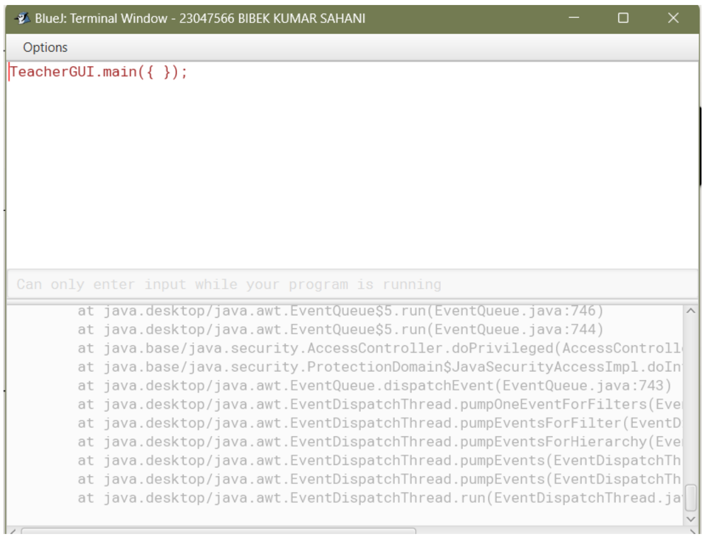
There was no and exception handling in jb2 “Grade assignment button”. While running the code by placing string value in text field format then there was error shown.

Before:



```
//to add function in grade the assignment button
jb2.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
        int yearsOfExperience = Integer.parseInt(jtf9.getText());
        int gradedScore = Integer.parseInt(jtf8.getText());
        String department = jtf7.getText();

        for (Teacher t : A1) {
            if (t instanceof Lecturer) {
                //downcasting
                Lecturer L1 = (Lecturer) t;
                L1.gradeAssignment(gradedScore, department, yearsOfExperience);
            }
        }

        //to verify the form is properly filled
        JOptionPane.showMessageDialog(null, "Grade Assigned successfully!");
    }
});
```

Figure 32: Code without exception handling



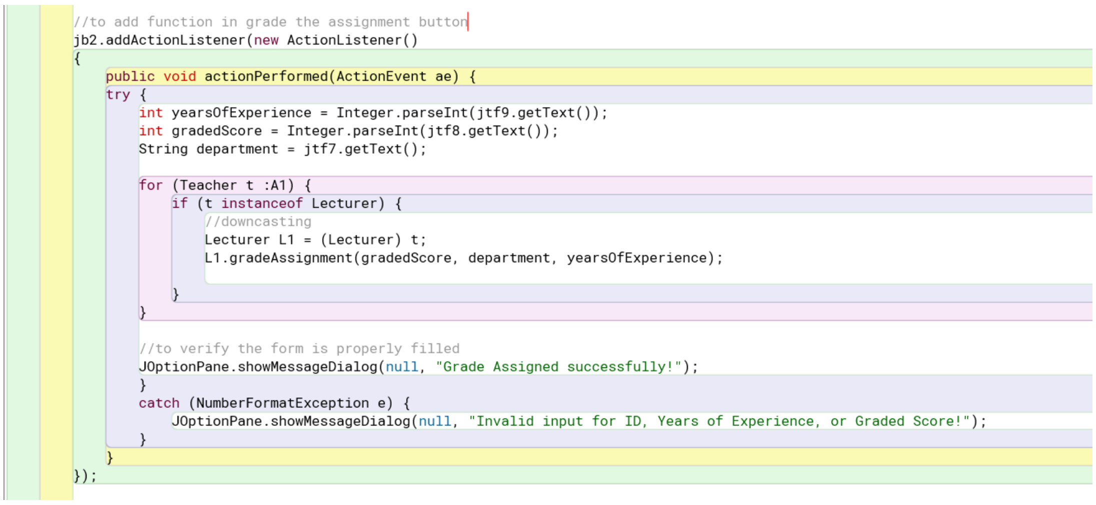
```
TeacherGUI.main({ });
```

```
Can only enter input while your program is running
```

```
        at java.desktop/java.awt.EventQueue$5.run(EventQueue.java:746)
        at java.desktop/java.awt.EventQueue$5.run(EventQueue.java:744)
        at java.base/java.security.AccessController.doPrivileged(AccessControll
        at java.base/java.security.ProtectionDomain$JavaSecurityAccessImpl.doIn
        at java.desktop/java.awt.EventQueue.dispatchEvent(EventQueue.java:743)
        at java.desktop/java.awt.EventDispatchThread.pumpOneEventForFilters(Eve
        at java.desktop/java.awt.EventDispatchThread.pumpEventsForFilter(EventD
        at java.desktop/java.awt.EventDispatchThread.pumpEventsForHierarchy(Eve
        at java.desktop/java.awt.EventDispatchThread.pumpEvents(EventDispatchTh
        at java.desktop/java.awt.EventDispatchThread.pumpEvents(EventDispatchTh
        at java.desktop/java.awt.EventDispatchThread.run(EventDispatchThread.ja
```

Figure 33: Error while running code without exception handling

After:



```
//to add function in grade the assignment button
jb2.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent ae) {
    try {
        int yearsOfExperience = Integer.parseInt(jtf9.getText());
        int gradedScore = Integer.parseInt(jtf8.getText());
        String department = jtf7.getText();

        for (Teacher t :A1) {
            if (t instanceof Lecturer) {
                //downcasting
                Lecturer L1 = (Lecturer) t;
                L1.gradeAssignment(gradedScore, department, yearsOfExperience);

            }
        }

        //to verify the form is properly filled
        JOptionPane.showMessageDialog(null, "Grade Assigned successfully!");
        }
        catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(null, "Invalid input for ID, Years of Experience, or Graded Score!");
        }
    }
});
```

Figure 34: Assigning Exception handling in code

Conclusion

The coursework was finally accomplished with lots of hard work and efforts. The different types of errors were occurred while doing the coursework. The main focus of the coursework was about the Graphical user interface (GUI) design which was also known as fundamental aspect of modern software development. I was able to learn and clearly understand the concepts Array List, Action Listener, components of Swing and AWT(Abstract Window Toolkit), and down casting Inheritance of classes. These all played a vital role in achieving the project objectives.

The obstacles and problems that were faced by me while during the coursework was adding the functions of buttons. The problems were solved by asking and visiting to the teacher daily and researching on various sites about down casting and up casting GUI designs in programming and also learning about the ideas and views which were kept on lecture and tutorial slides. It gave me a good knowledge and idea about programming to complete the course work.

References

byjus, 2024. *byjus.* [Online]
Available at: https://byjus.com/govt-exams/microsoft-word/
[Accessed 22 April 2024].

geeksforgeeks, 2022. *geeksforgeeks.* [Online]
Available at: https://www.geeksforgeeks.org/introduction-of-bluej/
[Accessed 22 April 2024].

intellipaat, 2024. *intellipat.* [Online]
Available at: https://intellipaat.com/blog/what-is-balsamiq/
[Accessed 22 April 2024].

Paraschiv, L., 2023. *Fotc.* [Online]
Available at: https://fotc.com/blog/draw-io-online-guide/#:~:text=What%20is%20Draw.io%3F,fully%20integrated%20with%20Google%20Drive.
[Accessed 22 April 2024].

Appendices

9.1 Code of Teacher class

```
public class Teacher

{

   private int teacherId;

   private String teacherName;  //camelcase

   private String address;

   private String workingType;

   private String employmentStatus;

   private int workingHours;


   //constructor method (Parameterized constructor)

   Teacher(int teacherId,String teacherName, String address, String workingType,
String employmentStatus)

   {

   //this.string = string;   (for reference)

   this.teacherId = teacherId;

   this.teacherName = teacherName;

   this.address = address;

   this.workingType = workingType;

   this.employmentStatus = employmentStatus;

   this.workingHours = 0;   //indicating not assigned

   }


   //accessor method (getter method)

   public int getTeacherId() {

      return teacherId;

   }


   public String getAddress() {
      return address;
   }

   public String getWorkingType() {
      return workingType;
   }

   public String getEmploymentStatus() {
      return employmentStatus;
   }

   public int getWorkingHours() {
      return workingHours;
   }

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

   //method to display teacher information
   public void displayTeacherInfo()
   {
      System.out.println("Teacher ID: "+ this.teacherId);
      System.out.println("Teacher Name: " + this.teacherName);
      System.out.println("Address: "+ this.address);
      System.out.println("Working Type: " + this.workingType);
      System.out.println("Employment Status: " + this.employmentStatus);

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

9.2 Code of Tutor class

```
public class Tutor extends Teacher {
    private double salary;
    private String specialization;
    private String academicQualifications;
    private int performanceIndex;
    private boolean isCertified;

    // Constructor
    public Tutor(int teacherId, String teacherName, String address, String
workingType, String employmentStatus,
                 int workingHours, double salary, String specialization, String
academicQualifications, int performanceIndex) {
        super(teacherId, teacherName, address, workingType, employmentStatus);
        super.setWorkingHours(workingHours); // Using the setter method from the
superclass
        this.salary = salary;
        this.specialization = specialization;
        this.academicQualifications = academicQualifications;
        this.performanceIndex = performanceIndex;
        this.isCertified = false;
    }

    // Accessor methods
    public double getSalary() {
        return salary;
    }

    public String getSpecialization() {
        return specialization;
    }

    public String getAcademicQualifications() {
        return academicQualifications;
    }

    public int getPerformanceIndex() {
        return performanceIndex;
    }

    public boolean isCertified() {
        return isCertified;
    }

    // Method to set salary and performanceIndex
    public void setSalaryAndPerformanceIndex(double newSalary, int
newPerformanceIndex) {
        if (!isCertified) {
            if (newPerformanceIndex > 5 && getWorkingHours() > 20) {
                double appraisalPercentage;
                if (newPerformanceIndex >= 5 && newPerformanceIndex <= 7) {
                    appraisalPercentage = 0.05;
                } else if (newPerformanceIndex >= 8 && newPerformanceIndex <= 9) {
                    appraisalPercentage = 0.10;
                } else {
```

```
                appraisalPercentage = 0.20;
            }

            double appraisalAmount = newSalary * appraisalPercentage;
            this.salary = newSalary + appraisalAmount;
            this.isCertified = true;
            this.performanceIndex = newPerformanceIndex;
        } else {
            System.out.println("Salary cannot be approved. Tutor has not been
certified yet or does not meet criteria.");
        }
    } else {
        System.out.println("Salary cannot be updated. Tutor has already been
certified.");
    }
}

// Method to remove the tutor
public void removeTutor() {
    this.salary = 0;
    this.specialization = "";
    this.academicQualifications = "";
    this.performanceIndex = 0;
    }


// Method to display tutor details
public void displayTutorInfo() {
    // Call the displayTeacherInfo method from the superclass
    super.displayTeacherInfo();
    if (isCertified) {
        System.out.println("Salary: " + salary);
        System.out.println("Specialization: " + specialization);
        System.out.println("Academic Qualifications: " + academicQualifications);
        System.out.println("Performance Index: " + performanceIndex);
    } else {
        System.out.println("Tutor has not been certified yet.");
    }
}
}
```

9.3 Code of Lecturer class

```
//Lecturer is subclass of Teacher
class Lecturer extends Teacher
{
    //attributes of lecturer class
    private String department;
    private int yearsOfExperience;   //camelcase
    private int gradedScore;
    private boolean hasGraded;
    private int workingHour;

    //constructor method (parameterized constructor)
    Lecturer(int teacherId, String teacherName, String address, String workingType,
String employmentStatus, String department, int yearsOfExperience, int
workingHours)
    {
        //constructor of super class
        super(teacherId, teacherName, address, workingType, employmentStatus);
        super.setWorkingHours(workingHours);
        this.department = department;
        this.yearsOfExperience = yearsOfExperience;
        this.hasGraded = false;   //initially, assignments have not been graded yet
    }

    //getter method
    public String getDepartment() {
        return department;
    }

     public int getYearsOfExperience() {
        return yearsOfExperience;
    }

     public boolean hasGraded() {
        return hasGraded;
    }

    //setter method
    public void setGradedScore(int gradedScore) {
        this.gradedScore = gradedScore;
    }

    //Additional method for grading assignments
    public void gradeAssignment(int gradedScore, String studentDepartment, int
studentYearsOfExperience)
    {
        if (!hasGraded && yearsOfExperience >= 5 &&
department.equals(studentDepartment))
        {
            if (gradedScore >= 70) {
                this.gradedScore = gradedScore;
                System.out.println("Grade: A");
            }
            else if (gradedScore >= 60) {
                this.gradedScore = gradedScore;
                System.out.println("Grade: B");
            }
            else if (gradedScore >= 50) {
                this.gradedScore = gradedScore;
                System.out.println("Grade: C");
            }
            else if (gradedScore >= 40) {
                this.gradedScore = gradedScore;
                System.out.println("Grade: D");
            }
            else {
                System.out.println("Grade: E");
            }
            //set graded to true
            this.hasGraded = true;
        }
        else {
            System.out.println("Not Graded(NG)");
        }
    }

    //method to display
    public void displayLecturerInfo()
    {
        super.displayTeacherInfo();  //display method of the superclass call
        System.out.println("Department: " + department);
        System.out.println("Years of Experience: " + yearsOfExperience);
        if (hasGraded) {
            System.out.println("Graded Score: " + gradedScore);
        }
        else {
            System.out.println("Not Graded(NG)");
        }
    }
}
```

9.4 Code of TecherGUI class

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

public class TeacherGUI{
    public static void main (String[]args) {
        ArrayList <Teacher> A1 = new ArrayList<Teacher>();
        JFrame jf1 = new JFrame("Teacher GUI");
        jf1.setSize(750, 470);
        jf1.setLayout(null);
        jf1.setResizable(false);

        JPanel jp1 = new JPanel();
        jp1.setBounds(0, 0, 750, 500);
        jp1.setLayout(null);
        jp1.setBackground(new Color(223,245,209));
        jf1.add(jp1);
```

```
JLabel l1 = new JLabel("Lecturer");

JLabel l2 = new JLabel("Teacher ID:");
JLabel l3 = new JLabel("Teacher Name:");
JLabel l4 = new JLabel("Address:");

JLabel l5 = new JLabel("Working Type:");
JLabel l6 = new JLabel("Employment Status:");
JLabel l10 = new JLabel("Working Hour:");

JLabel l7 = new JLabel("Department:");
JLabel l8 = new JLabel("Graded Score:");
JLabel l9 = new JLabel("Years of Experience:");


l1.setFont(new Font("Serif", Font.PLAIN, 25));


JTextField jtf2 = new JTextField();
JTextField jtf3 = new JTextField();
JTextField jtf4 = new JTextField();

JTextField jtf5 = new JTextField();
JTextField jtf6 = new JTextField();
JTextField jtf10 = new JTextField();

JTextField jtf7 = new JTextField();
JTextField jtf8 = new JTextField();
JTextField jtf9 = new JTextField();
```

```
JButton jb1 = new JButton("Add");
JButton jb2 = new JButton("Grade Assignment");
JButton jb3 = new JButton("Display");
JButton jb4 = new JButton("Clear");
JButton jb5 = new JButton("Tutor");


//for setting the bounds od labels
l1.setBounds(325, 10, 130, 35);

l2.setBounds(25, 80, 150, 35);
l3.setBounds(25, 115, 150, 35);
l4.setBounds(25, 150, 150, 35);

l5.setBounds(370, 80, 150, 35);
l6.setBounds(370, 115, 150, 35);
l10.setBounds(370, 150, 150, 35);

l7.setBounds(25, 220, 150, 35);
l8.setBounds(25, 255, 150, 35);
l9.setBounds(25, 290, 150, 35);


//for setting bounds of textfields
jtf2.setBounds(130, 83, 75, 25);
jtf3.setBounds(130, 118, 150, 25);
jtf4.setBounds(130, 153, 150, 25);
```

```
        jtf5.setBounds(490, 83, 150, 25);
        jtf6.setBounds(490, 118, 150, 25);
        jtf10.setBounds(490, 153, 150, 25);

        jtf7.setBounds(160, 223, 150, 25);
        jtf8.setBounds(160, 258, 150, 25);
        jtf9.setBounds(160, 293, 150, 25);



        jb1.setBounds(25, 390, 100, 25);
        jb2.setBounds(70, 335, 150, 25);
        jb3.setBounds(185, 390, 100, 25);
        jb4.setBounds(370, 390, 100, 25);
        jb5.setBounds(555, 390, 100, 25);

        //for adding labels
        jp1.add(l1);

        jp1.add(l2);
        jp1.add(l3);
        jp1.add(l4);

        jp1.add(l5);
        jp1.add(l6);
        jp1.add(l10);

        jp1.add(l7);
        jp1.add(l8);
        jp1.add(l9);
```

```
//for adding textfield
jp1.add(jtf2);
jp1.add(jtf3);
jp1.add(jtf4);

jp1.add(jtf5);
jp1.add(jtf6);
jp1.add(jtf10);

jp1.add(jtf7);
jp1.add(jtf8);
jp1.add(jtf9);

//for adding buttons
jp1.add(jb1);
jp1.add(jb2);
jp1.add(jb3);
jp1.add(jb4);
jp1.add(jb5);

//to add function of button for adding the lecturer
jb1.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent ae) {
    try {
        //converting teacherID, YearsofExperience, and GradedScore into integer
        int teacherId = Integer.parseInt(jtf2.getText());
        String teacherName = jtf3.getText();
        String address = jtf4.getText();
        String workingType = jtf5.getText();
```

```
            String employmentStatus = jtf6.getText();
            int yearsOfExperience = Integer.parseInt(jtf9.getText());
            //int gradedScore = Integer.parseInt(jtf8.getText());
            String department = jtf7.getText();
            int workingHour = Integer.parseInt(jtf10.getText());

            Lecturer L1 = new Lecturer(teacherId, teacherName, address,
workingType, employmentStatus, department, yearsOfExperience, workingHour);
            A1.add(L1);

            //to verify the form is properly filled
            JOptionPane.showMessageDialog(null, "Lecturer added successfully!");
        }
        catch (NumberFormatException e) { //it handles the exception
            JOptionPane.showMessageDialog(null, "Invalid Input!");
        }
        }
    });


    //to add function in grade the assignment button
    jb2.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent ae) {
        try {
            int yearsOfExperience = Integer.parseInt(jtf9.getText());
            int gradedScore = Integer.parseInt(jtf8.getText());
            String department = jtf7.getText();

            for (Teacher t :A1) {
                if (t instanceof Lecturer) {
                    //downcasting
                    Lecturer L1 = (Lecturer) t;
                    L1.gradeAssignment(gradedScore, department, yearsOfExperience);

                }
            }

            //to verify the form is properly filled
            JOptionPane.showMessageDialog(null, "Grade Assigned successfully!");
            }
            catch (NumberFormatException e) {
                JOptionPane.showMessageDialog(null, "Invalid input for ID, Years of
Experience, or Graded Score!");
            }
        }
    });

    //to add the function to display the lecturer
    jb3.addActionListener(new ActionListener()
    {
      public void actionPerformed(ActionEvent ae)
      {
          for (Teacher t : A1)
          {
              if (t instanceof Lecturer)
              {
                  //downcasting for displaying the lecturer
                  Lecturer L1 = (Lecturer) t;
                  L1.displayLecturerInfo();
              }
          }
    }
  });

  //to add the function clear the textfield of lecturer
  jb4.addActionListener(new ActionListener()
  {
     public void actionPerformed(ActionEvent ae) {

        jtf2.setText("");
        jtf3.setText("");
        jtf4.setText("");
        jtf5.setText("");
        jtf6.setText("");
        jtf7.setText("");
        jtf8.setText("");
        jtf9.setText("");
        jtf10.setText("");

        JOptionPane.showMessageDialog(null, "Textfields are cleared!");
     }
  });

  //to add the function of changing lecturer to tutor
  jb5.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent ae) {

  JPanel jp2 = new JPanel();
  jp2.setBounds(0, 0, 750, 500);
  jp2.setLayout(null);
  jp2.setBackground(new Color(223,245,209));
```

```
jf1.add(jp2);

//for creating label for tutor
JLabel l_1 = new JLabel("Tutor");

JLabel l_2 = new JLabel("Teacher ID:");
JLabel l_3 = new JLabel("Teacher Name:");
JLabel l_4 = new JLabel("Address:");

JLabel l_5 = new JLabel("Working Type:");
JLabel l_6 = new JLabel("Working Hours:");
JLabel l_7 = new JLabel("Specialization:");
JLabel l_8 = new JLabel("Employment Status:");
JLabel l_9 = new JLabel("Academic Qualification:");

JLabel l_10 = new JLabel("Salary:");
JLabel l_11 = new JLabel("Performance Index:");

l_1.setFont(new Font("Serif", Font.PLAIN, 25));

//for creating textfield for tutor
JTextField jtf_2 = new JTextField();
JTextField jtf_3 = new JTextField();
JTextField jtf_4 = new JTextField();

JTextField jtf_5 = new JTextField();
JTextField jtf_6 = new JTextField();
JTextField jtf_7 = new JTextField();
JTextField jtf_8 = new JTextField();
JTextField jtf_9 = new JTextField();
JTextField jtf_10 = new JTextField();
JTextField jtf_11 = new JTextField();

//for adding buttons for tutor
JButton jb_1 = new JButton("Add");
JButton jb_2 = new JButton("Set Salary");
JButton jb_3 = new JButton("Remove Tutor");
JButton jb_4 = new JButton("Display");
JButton jb_5 = new JButton("Clear");
JButton jb_6 = new JButton("Lecturer");

//for setting the bounds of labels
l_1.setBounds(325, 10, 130, 35);

l_2.setBounds(25, 80, 150, 35);
l_3.setBounds(25, 115, 150, 35);
l_4.setBounds(25, 150, 150, 35);

l_5.setBounds(25, 220, 150, 35);
l_6.setBounds(25, 255, 150, 35);
l_7.setBounds(25, 290, 150, 35);
l_8.setBounds(25, 325, 150, 35);
l_9.setBounds(400, 150, 150, 35);

l_10.setBounds(400, 80, 150, 35);
l_11.setBounds(400, 115, 150, 35);
```

```
//for setting the bounds of textfields
jtf_2.setBounds(130, 83, 75, 25);
jtf_3.setBounds(130, 118, 150, 25);
jtf_4.setBounds(130, 153, 150, 25);

jtf_5.setBounds(150, 223, 150, 25);
jtf_6.setBounds(150, 258, 150, 25);
jtf_7.setBounds(150, 293, 150, 25);
jtf_8.setBounds(150, 328, 150, 25);
jtf_9.setBounds(550, 158, 150, 25);

jtf_10.setBounds(550, 83, 150, 25);
jtf_11.setBounds(550, 118, 150, 25);


//for setting the bounds of buttons
jb_1.setBounds(25, 390, 100, 25);
jb_2.setBounds(470, 210, 150, 25);
jb_3.setBounds(455, 255, 180, 25);
jb_4.setBounds(185, 390, 100, 25);
jb_5.setBounds(370, 390, 100, 25);
jb_6.setBounds(555, 390, 100, 25);


//for adding labels
jp2.add(l_1);

jp2.add(l_2);
jp2.add(l_3);
jp2.add(l_4);
```

```
jp2.add(l_5);
jp2.add(l_6);
jp2.add(l_7);
jp2.add(l_8);
jp2.add(l_9);

jp2.add(l_10);
jp2.add(l_11);



//for adding textfield
jp2.add(jtf_2);
jp2.add(jtf_3);
jp2.add(jtf_4);

jp2.add(jtf_5);
jp2.add(jtf_6);
jp2.add(jtf_7);
jp2.add(jtf_8);
jp2.add(jtf_9);

jp2.add(jtf_10);
jp2.add(jtf_11);



//for adding buttons
jp2.add(jb_1);
jp2.add(jb_2);
jp2.add(jb_3);
```

```
        jp2.add(jb_4);
        jp2.add(jb_5);
        jp2.add(jb_6);

        jf1.getContentPane();
        jf1.remove(jp1);
        jf1.revalidate();
        jf1.repaint();



        //to add the function to add the tutor in add button
        jb_1.addActionListener(new ActionListener()
        {
           public void actionPerformed(ActionEvent ae) {
           try {
              //converting teacherID, YearsofExperience, and GradedScore into integer
              int teacherId = Integer.parseInt(jtf_2.getText());
              String teacherName = jtf_3.getText();
              String address = jtf_4.getText();
              String workingType = jtf_5.getText();
              String employmentStatus = jtf_8.getText();
              int workingHours = Integer.parseInt(jtf_6.getText());
              double salary = Double.parseDouble(jtf_10.getText());
              String specialization = jtf_7.getText();
              String academicQualification = jtf_9.getText();
              int performanceIndex = Integer.parseInt(jtf_11.getText());

              Tutor T1 = new Tutor(teacherId, teacherName, address, workingType,
employmentStatus, workingHours, salary, specialization, academicQualification,
performanceIndex);
              A1.add(T1);
        //to verify the form is properly filled
        JOptionPane.showMessageDialog(null, "Tutor added successfully!");
    }
    catch (NumberFormatException e) { //it handles the exception
        JOptionPane.showMessageDialog(null, "Invalid Input!");
    }
    }
});

//to add the function to set salary in salary button
jb_2.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent ae) {
    try{
        double salary = Double.parseDouble(jtf_10.getText());
        int performanceIndex = Integer.parseInt(jtf_11.getText());
        for (Teacher t : A1) {
            if (t instanceof Tutor) {
                Tutor T0 = (Tutor) t;

                T0.setSalaryAndPerformanceIndex(salary, performanceIndex);

                JOptionPane.showMessageDialog(null, "Salary and performance
index have been set succesfully!");
            }
        }
    }

    catch (NumberFormatException e) { //it handles the exception
```

```
                JOptionPane.showMessageDialog(null, "Invalid input for salary and
performance Index!");
            }
                }
        });

        //to add the function of remove tutor in remove button
        jb_3.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent ae) {
                for (Teacher t : A1) {
                    if (t instanceof Tutor && t.getTeacherId() ==Integer.parseInt(
jtf_2.getText())) {
                        Tutor T2 = (Tutor) t;
                        T2.removeTutor();

                        JOptionPane.showMessageDialog(null, "Tutor has been removed
succesfully!");
                        break;
                    }
                }
            }
        });

        //to add the function of display tutor info in display button
        jb_4.addActionListener(new ActionListener()
        {
           public void actionPerformed(ActionEvent ae)
           {
               for (Teacher t : A1)
               {
```

```
            if (t instanceof Tutor )
            {
                //downcasting for displaying the tutor info
                Tutor T1 = (Tutor) t;
                T1.displayTutorInfo();
            }
        }
    }
});

//for clearing all the fieldbox
jb_5.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent ae) {
        jtf_2.setText("");
        jtf_3.setText("");
        jtf_4.setText("");
        jtf_5.setText("");
        jtf_6.setText("");
        jtf_7.setText("");
        jtf_8.setText("");
        jtf_9.setText("");
        jtf_10.setText("");
        jtf_11.setText("");

        JOptionPane.showMessageDialog(null, "Textfields are cleared!");
    }
});

//for returning back to lecturer
```

```
        jb_6.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent ae) {
              // Removing the tutor panel (jp2)
              jf1.getContentPane();
              jf1.remove(jp2);

              // Adding the lecturer panel (jp1) back to the JFrame
              jf1.add(jp1);
              jf1.revalidate();
              jf1.repaint();

               }
            });
            }
        });

        //to set the frame visible
        jf1.setVisible(true);
    }
}
```