A logo for a university

Description automatically generated



**CS4001NI Programming**

**30% Individual Coursework**

**2023/24 Spring**

**Student Name:** Arpit Neupane

**London Met ID:** 23050197

**College ID:** NP01CP4A230331

**Assignment Due Date:** 10 May, 2024

**Assignment Submission Date:** 10 May, 2024

**Word Count:**

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

# Table of Content

Contents

[Table of Content 2](#_Toc166158813)

[Table of Figures 4](#_Toc166158814)

[Table of Tables 5](#_Toc166158815)

[INTRODUCTION 1](#_Toc166158816)

[Tools Used 1](#_Toc166158817)

[BlueJ 1](#_Toc166158818)

[MS-Word 2](#_Toc166158819)

[Class Diagram 3](#_Toc166158820)

[Pseudocode 5](#_Toc166158821)

[Method Description 35](#_Toc166158822)

[Testing 37](#_Toc166158823)

[Test 1: Checking if the program can be compiled and ran through the command prompt. 37](#_Toc166158824)

[Image proving the test result 38](#_Toc166158825)

[Test 2: To inspect the effects of Add lecturer, Add Tutor, Grade Assignment, Set Salary and Remove Tutor. 40](#_Toc166158826)

[Images proving test result 41](#_Toc166158827)

[Test 3: To insure that appropriate dialog boxes appear when unsuitable values are entered for Teacher ID. 45](#_Toc166158828)

[Images proving test result 46](#_Toc166158829)

[46](#_Toc166158830)

[Error, It’s Types and Detection 47](#_Toc166158831)

[Syntax Error 47](#_Toc166158832)

[Logical Error 49](#_Toc166158833)

[Runtime Error 50](#_Toc166158834)

[Changes to the old program 52](#_Toc166158835)

[Teacher 52](#_Toc166158836)

[52](#_Toc166158837)

[Tutor 52](#_Toc166158838)

[Lecturer 54](#_Toc166158839)

[Conclusion 56](#_Toc166158840)

[Bibliography 57](#_Toc166158841)

[Bibliography 57](#_Toc166158842)

[Appendix 58](#_Toc166158843)

[Teacher 58](#_Toc166158844)

[Lecturer 61](#_Toc166158845)

[Tutor 65](#_Toc166158846)

[TeacherGUI 69](#_Toc166158847)

# Table of Figures

[Figure 1: Image proving test 1.1 38](#_Toc166158848)

[Figure 2: Image proving test 1.2 39](#_Toc166158849)

[Figure 3: Image proving test 2.1 41](#_Toc166158850)

[Figure 4: Image proving test 2.2 42](#_Toc166158851)

[Figure 5: Image proving test 2.3 42](#_Toc166158852)

[Figure 6: Image proving test 2.4 42](#_Toc166158853)

[Figure 7: Image proving test 2.5 43](#_Toc166158854)

[Figure 8: Image proving test 2.6 43](#_Toc166158855)

[Figure 9: Image proving test 2.7 44](#_Toc166158856)

[Figure 10: Image proving test 2.8 44](#_Toc166158857)

[Figure 11: Image proving test 3.1 46](#_Toc166158858)

[Figure 12: Image proving test 3.2 46](#_Toc166158859)

[Figure 13: Syntax Error 1 48](#_Toc166158860)

[Figure 14: Syntax Error 2 48](#_Toc166158861)

[Figure 15: Logical Error 1 49](#_Toc166158862)

[Figure 16: Logical Error 2 49](#_Toc166158863)

[Figure 17: Runtime Error 1 50](#_Toc166158864)

[Figure 18: Runtime Error 2 51](#_Toc166158865)

[Figure 19: Runtime Error 3 51](#_Toc166158866)

[Figure 20: Change 1 52](file:///D:\Arpeet\College\Programming\java%20programming\Java%20Coursework%20improvada\Official%20Coursework.docx#_Toc166158867)

[Figure 21: Change 2 53](#_Toc166158868)

[Figure 22:Change 3 53](#_Toc166158869)

[Figure 23: Change 4 54](#_Toc166158870)

[Figure 24: Change 5 55](#_Toc166158871)

# Table of Tables

[Table 1: Parent Class 3](#_Toc166158872)

[Table 2: Child Class 1 3](#_Toc166158873)

[Table 3: Child Class 2 3](#_Toc166158874)

[Table 4: TeacherGUI class 4](#_Toc166158875)

[Table 5: Test 1 38](#_Toc166158876)

[Table 6: Test 2 40](#_Toc166158877)

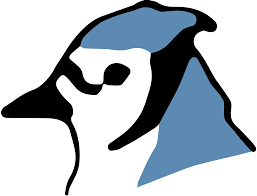
[Table 7: Test 3 45](#_Toc166158878)

# INTRODUCTION

Welcome to the documentation of the project, GUI building. This report/documentation serves as a simple guideline to those who may be new to the project or even people who might have worked towards this project. The project focuses on building a GUI(Graphical User Interface) for Teacher Management System of some kind, featuring lecturers and tutors. Functionalities such as grading the assignment of students, setting salary of tutors who have completed certain criteria, and removing tutors have been implemented as well.

## Tools Used

### BlueJ

BlueJ is a text editor or IDE(Integrated Development Environment) for Java which is a programming language. BlueJ is known all over for it’s simplicity, which has made it ideal for starting out. It was developed to tailor for teaching, and also comes with dedicated resources which includes a textbook for course. The interactive interface of BlueJ allows the users to interact with OOP specific concepts such as objects, which has made it powerful tool to learning Java. It works on several platforms and has been continuously updated to make it better for more than fifteen years now. (Oracle, 2007)

### MS-Word

The documentation was written and edited on MS-Word which is a fantastic software that allows one to create both simple and complex documents. It works on multiple platforms which is convenient in case of switching devices. It supports multiple features such as traditional options like copy, cut and paste, add headers and footers, page formatting and many more. (Rock, 2022)



# Class Diagram

Table 1: Parent Class

|  |
| --- |
| Teacher |
| - teacher\_id : Int  - teacher\_name : String  - address : String  - working\_type : String  - employment\_status : String  - working\_hours : Int |
| + <<constructor>> Teacher (teacher\_id : Int, teacher\_name : String, address : String, working\_type : String, employment\_status : String)  + getTeacher\_id() : Int  + getTeacher\_name() : String  + getAddress() : String  + getWorking\_type() : String  + getEmployment\_status() : String  + getWorking\_hours() : Int  + setWorking\_Hours(working\_hours : int) : Void  + display() : Void |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Lecturer | | - department : String  - years\_of\_experience : Int  - graded\_score : Int  - has\_graded : Boolean | | + <<constructor>> Lecturer (teacher\_id : Int, teacher\_name : String, address : String, working\_type : String, employment\_status : String, working\_hours : Int, department : String, years\_of\_experience : Int)  + getDepartment() : String  + getYears\_of\_experience() : Int  + getGraded\_score() : Int  + isHas\_graded() : Boolean  + setGraded\_Score(graded\_score : Int) : Void  + gradeAssignment(graded\_score : Int, department : String, years\_of\_experience : Int) : Void  + display() : Void |   Table 2: Child Class 1 | |  | | --- | | Tutor | | - salary : double  - specialization : String  - academic\_qualifications: String  - performance\_index : Int  - is\_certified : Boolean | | + <<constructor>> Tutor (teacher\_id : Int, teacher\_name : String, address : String, working\_type : String, employment\_status : String, working\_hours : Int, salary : Double, specialization : String, academic\_qualifications : String, performance\_index : Int)  + getSalary() : Double  + getSpecialization() : String  + getAcademic\_qualifications():  . String  + getPerformance\_index() : Int  + getIs\_certified() : Boolean  + setSalary(newSalary : double, new\_performance\_index : Int) : Void  + removeTutor() : Void  + display() : Void |   Table 3: Child Class 2 |

|  |
| --- |
| TeacherGUI |
| - listOfTeacher: ArrayList  - buttonsFrame, lecturerFrame, tutorFrame, gradeFrame, salaryFrame, removeFrame, lDisplayFrame, tDisplayFrame: JFrame  - tDisplayButton, tClearButton, tAddButton, tBackButton, lDisplayButton, lClearButton, lAddButton, lBackButton, gGradeButton, gClearButton, gBackButton, sSetSalaryButton, sClearButton, sBackButton, rRemoveButton, rBackButton, dlBackButton, dtBackButton: JButton  - mainPanel, lPanel, tPanel, gPanel, sPanel, rPanel, lDisplayPanel, tDisplayPanel: JPanel  - lHeader, lIdLabel, lNameLabel, lAddressLabel, lWorkingTypeLabel, lEmploymentStatusLabel, lDepartmentLabel, lExperienceLabel, lWorkingHoursLabel, tHeader, tIdLabel, tNameLabel, tAddressLabel, tWorkingTypeLabel, tEmploymentStatusLabel, tWorkingHoursLabel, tSalaryLabel, tSpecializationLabel, gHeader, gIdLabel, gDepartmentLabel, gGradedScoreLabel, gExperienceLabel, sHeader, sSalaryLabel, sPerformanceLabel, sIdLabel, rIdLabel, rHeader, dlHeader,, dtHeader: JLabel  - lId, lName, lAddress, lWorkingType, lEmploymentStatus, lDepartment, lExperience, lWorkingHours, tId, tName, tAddress, tWorkingType, tEmploymentStatus, tWorkingHours, tSalary, tSpecialization, tQualification, tPerformance, gId, gDepartment, gExperience, gGradedScore, sSalary, sPerformance, sId, rId: JTextField  - scrollingPanel: JScrollPane  - lIdAvailable, experienceL, workingHoursL, performanceT, workingHoursT, salaryT, tIdAvailable, idCheck, gradedScoreCheck, experienceCheck, salaryCheck, confirmation, idR: Int  - nameL, addressL, departmentL, workingTypeL, employmentStatusL, message, nameT, addressT, specializationT, workingTypeT, employmentStatusT, qualificationT, departmentCheck, resultingGrade, updatedSalary: String  - isRemoved, isAdded: Boolean |
| + <<constructor>> TeacherGUI()  + lecturersFrame(): Void  + displayLecturerList(): Void  + displayTutorList(): Void  + tutorsFrame(): Void  + gradesAssignment(): Void  + setsSalary(): Void  + removesTutor(): Void  + actionPerformed(): Void |

Table 4: TeacherGUI class

# Pseudocode

IMPORT required libraries such as: all from Swing and ActionEvent, ActionListener, Color, Font and Dimension from AWT and ArrayList.

CREATE a class TeacherGUI

DO

DECLARE ArrayList listOfTeacher

DECLARE JFrame buttonsFrame

DECLARE JPanel mainPanel

DECLARE JLabel welcomeLabel

DECLARE JButton addLecturerButton, addTutorButton, gradeButton,

setSalaryButton, removeTutorButton, exitButton

DECLARE JFrame lecturerFrame

DECLARE JPanel lPanel

DECLARE JLabel lHeader, lIdLabel, lNameLabel, lAddressLabel,

lWorkingTypeLabel, lEmploymentStatusLabel, lDepartmentLabel, lExperienceLabel,

lWorkingHoursLabel

DECLARE JTextField lId, lName, lAddress, lWorkingType, lEmploymentStatus,

lDepartment, lExperience, lWorkingHours

DECLARE JButton lDisplayButton, lClearButton, lAddButton, lBackButton

DECLARE JFrame tutorFrame

DECLARE JPanel tPanel

DECLARE JLabel tHeader, tIdLabel, tNameLabel, tAddressLabel,

tWorkingTypeLabel, tEmploymentStatusLabel, tWorkingHoursLabel, tSalaryLabel,

tSpecializationLabel

DECLARE JLabel tQualificationLabel, tPerformanceLabel

DECLARE JTextField tId, tName, tAddress, tWorkingType, tEmploymentStatus,

tWorkingHours, tSalary, tSpecialization, tQualification, tPerformance

DECLARE JButton tDisplayButton, tClearButton, tAddButton, tBackButton

DECLARE JFrame gradeFrame

DECLARE JPanel gPanel

DECLARE JLabel gHeader, gIdLabel, gDepartmentLabel, gGradedScoreLabel,

gExperienceLabel

DECLARE JTextField gId, gDepartment, gExperience, gGradedScore

DECLARE JButton gGradeButton, gClearButton, gBackButton

DECLARE JFrame salaryFrame

DECLARE JPanel sPanel

DECLARE JLabel sHeader, sSalaryLabel, sPerformanceLabel, sIdLabel

DECLARE JTextField sSalary, sPerformance, sId

DECLARE JButton sSetSalaryButton, sClearButton, sBackButton

DECLARE JFrame removeFrame

DECLARE JPanel rPanel

DECLARE JLabel rIdLabel, rHeader

DECLARE JTextField rId

DECLARE JButton rRemoveButton, rBackButton

DECLARE JFrame lDisplayFrame

DECLARE JPanel lDisplayPanel

DECLARE JLabel dlHeader

DECLARE JLabel lIdData, lNameData, lAddressData, lWorkingTypeData,

lEmploymentStatusData, lDepartmentData, lExperienceData, lWorkingHoursData,

lGradedScoreData

DECLARE JButton dlBackButton

DECLARE JFrame tDisplayFrame

DECLARE JPanel tDisplayPanel

DECLARE JLabel dtHeader

DECLARE JButton dtBackButton

DEFINE constructor TeacherGUI()

DO

INITIALIZE buttonsFrame

SET buttonsFrame close operation

SET buttonsFrame size

SET buttonsFrame location

INITIALIZE mainPanel to null

INITIALIZE and SET BOUNDS of welcomeLabel, addLecturerButton,

addTutorButton, gradeButton, setSalaryButton, removeTutorButton and

exitButton

ADD action listener to addLecturerButton

DO

CREATE lecturerFrame

SET lecturerFrame close operation

SET lecturerFrame size

SET lecturerFrame location

CALL lecturersFrame method

SET lecturerFrame visible

SET buttonsFrame invisible

END DO

ADD action listener to addTutorButton

DO

CREATE tutorFrame

SET tutorFrame close operation

SET tutorFrame size

SET tutorFrame location

CALL tutorsFrame method

SET tutorFrame visible

SET buttonsFrame invisible

END DO

ADD action listener to gradeButton

DO

CREATE gradeFrame

SET gradeFrame close operation

SET gradeFrame size

SET gradeFrame location

CALL gradesAssignment method

SET gradeFrame visible

SET buttonsFrame invisible

END DO

ADD action listener to setSalaryButton

DO

CREATE salaryFrame

SET salaryFrame close operation

SET salaryFrame size

SET salaryFrame location

CALL setsSalary method

SET salaryFrame visible

SET buttonsFrame invisible

END DO

ADD action listener to removeTutorButton

DO

CREATE removeFrame

SET removeFrame close operation

SET removeFrame size

SET removeFrame location

CALL removesTutor method

SET removeFrame visible

SET buttonsFrame invisible

END DO

SET welcomeLabel font size and color

SET buttons font size

SET welcomeLabel text color

SET mainPanel background color

ADD welcomeLabel to mainPanel

ADD buttons to mainPanel

ADD mainPanel TO buttonsFrame

SET buttonsFrame to visible

DEFINE lecturersFrame() with return type void

DO

DECLARE JPanel lPanel of null layout

SET lPanel background color to new Color(47, 79, 79)

DECLARE JLabel lHeader, lIdLabel, lNameLabel, lAddressLabel,

lWorkingTypeLabel, lDepartmentLabel, lExperienceLabel, lWorkingHoursLabel,

lEmploymentStatusLabel

INITIALIZE lHeader with text "Enter Lecturer Details" and set bounds

INITIALIZE lIdLabel with text "Id:" and set bounds

INITIALIZE lNameLabel with text "Name:" and bounds

INITIALIZE lAddressLabel with text "Address:" and bounds

INITIALIZE lWorkingTypeLabel with text "Working Type:" and bounds

INITIALIZE lDepartmentLabel with text "Department:" and bounds

INITIALIZE lExperienceLabel with text "Experience(Years):" and bounds

INITIALIZE lWorkingHoursLabel with text "Working Hours:" and bounds

INITIALIZE lEmploymentStatusLabel with text "Employment Status:" and bounds

DECLARE JTextField lId, lName, lAddress, lWorkingType, lDepartment,

lExperience, lWorkingHours, lEmploymentStatus

INITIALIZE lId, lName, lAddress, lWorkingType, lDepartment, lExperience,

lWorkingHours, lEmploymentStatus and set bounds

INITIALIZE lAddButton with text "Add to list, lDisplayButton with text "Display,

lClearButton with text "Clear, lBackButton with text "<-Back-“ and set bounds

ADD action listener to lAddButton

DO

IMPLEMENT try-catch block

IF any of the text fields are empty

DO

SHOW error message dialog "Empty Fields Found! Please fill all text

areas."

END DO

ELSE

DO

PARSE input to integers and strings respectively and store in variables

IF id and working hours are non-negative

DO

DECLARE boolean isAdded as true

FOR each teacher in listOfTeacher

DO

IF lecturer id matches and teacher is instance of Lecturer

DO

SET isAdded to false

END DO

END DO

IF isAdded is true

DO

CREATE new Lecturer object

ADD lecturer to listOfTeacher

SHOW message dialog "Successfully Added!"

CLEAR all text fields

END DO

ELSE

DO

SHOW error message dialog "Id is in use!! Try another lecturer id."

END DO

ELSE

DO

SHOW message dialog "Please enter positive values."

END DO

END DO

CATCH NumberFormatException

DO

SHOW error message dialog "Number Format Exception. Enter number only

in number-required fields."

END DO

END DO

ADD action listener to lClearButton

DO

IF any text field is not empty

DO

DISPLAY confirmation dialog "Do you really want to clear all fields?"

IF user chooses yes

DO

CLEAR all text fields

END DO

END DO

END DO

ADD action listener to lDisplayButton

DO

CREATE new display frame "Display Lecturer Details"

SET display frame close operation

SET display frame size

SET display frame location relative to buttonsFrame

CALL displayLecturerList method

END DO

ADD action listener to lBackButton

DO

CLEAR all text fields

HIDE lecturerFrame

SHOW buttonsFrame

END DO

DECLARE Font labelFont, headerFont, buttonFont

SET labelFont to lIdLabel font with size 16 and bold

SET headerFont to lHeader font with size increased by 80%

SET buttonFont to lClearButton font with size increased by 20%

SET lHeader font color to white

SET lIdLabel, lNameLabel, lAddressLabel, lWorkingTypeLabel, lDepartmentLabel,

lExperienceLabel, lWorkingHoursLabel, lEmploymentStatusLabel font color to

white

SET lDisplayButton, lClearButton, lAddButton, lBackButton background color to

new Color(211, 211, 211)

ADD lHeader, lIdLabel, lNameLabel, lAddressLabel, lWorkingTypeLabel,

lDepartmentLabel, lExperienceLabel, lWorkingHoursLabel,

lEmploymentStatusLabel to lPanel

ADD lId, lName, lAddress, lWorkingType, lDepartment, lExperience,

lWorkingHours, lEmploymentStatus to lPanel

ADD lAddButton, lDisplayButton, lClearButton, lBackButton to lPanel

ADD lPanel to lecturerFrame

END DO

DEFINE tutorsFrame() with return type void

DO

DECLARE JPanel tPanel with layout set to null

SET tPanel background color to new Color(47, 79, 79)

DECLARE JLabel tHeader, tIdLabel, tNameLabel, tAddressLabel,

tWorkingTypeLabel, tSpecializationLabel, tPerformanceLabel, tSalaryLabel,

tWorkingHoursLabel, tEmploymentStatusLabel, tQualificationLabel

INITIALIZE tHeader with text "Enter Tutor Details ", tIdLabel with text "Id: " ,

tNameLabel with text "Name: ", tAddressLabel with text "Address: ",

tWorkingTypeLabel with text "Working Type: ", tSpecializationLabel with text

"Specialization: ", tPerformanceLabel with text "Performance Index: ", tSalaryLabel

with text "Salary: ", tWorkingHoursLabel with text "Working Hours: ",

tEmploymentStatusLabel with text "Employment Status: ", tQualificationLabel with

text "Qualification: " and set proper bounds

DECLARE, INITIALZE and set bounds of JTextField tId, tName, tAddress,

tWorkingType, tSpecialization, tPerformance, tSalary, tWorkingHours,

tEmploymentStatus, tQualification

DECLARE JButton tAddButton, tDisplayButton, tClearButton, tBackButton

INITIALIZE tAddButton with text "Add to list", tDisplayButton with text "Display",

tClearButton with text "Clear", tBackButton with text "<-Back-" and set bounds

ADD action listener TO tAddButton:

DO

IMPLEMENT try-catch block

DO

IF any text field is empty

DO

SHOW error message dialog "Empty Fields Found! Please fill all text

areas. "

END DO

ELSE

DO

PARSE input to integers and strings

IF id, salary, performance, and working hours are non-negative

DO

DECLARE boolean isAdded as true

FOR each teacher in listOfTeacher

DO

IF tutor id matches and teacher is instance of Tutor

DO

SET isAdded to false

END DO

END DO

IF isAdded is true

DO

CREATE new Tutor object

ADD tutor to listOfTeacher

SHOW message dialog "Successfully Added!"

CLEAR all text fields

END DO

ELSE

DO

SHOW error message dialog "Id is already in use!! Try another tutor

id."

END DO

END DO

ELSE

DO

SHOW message dialog "Please enter positive values."

END DO

END DO

END DO

CATCH NumberFormatException

DO

SHOW error message dialog "Number Format Exception. Enter number only

in number-required fields."

END DO

END DO

ADD action listener to tClearButton

DO

IF any text field is not empty

DO

DISPLAY confirmation dialog "Do you really want to clear all fields?"

IF user chooses yes

DO

CLEAR all text fields

END DO

END DO

END DO

ADD action listener to tDisplayButton

DO

CREATE new display frame "Display Tutor Details"

SET display frame close operation

SET display frame size

SET display frame location relative to buttonsFrame

CALL displayTutorList method

END DO

ADD action listener to tBackButton

DO

CLEAR all text fields

HIDE tutorFrame

SHOW buttonsFrame

END DO

DECLARE Font labelFont, headerFont, buttonFont

SET labelFont to tIdLabel font with size 16 and bold

SET headerFont to tHeader font with size increased by 80%

SET buttonFont to tClearButton font with size increased by 20%

SET tHeader font color to white

SET tIdLabel, tNameLabel, tAddressLabel, tWorkingTypeLabel,

tSpecializationLabel, tPerformanceLabel, tSalaryLabel, tWorkingHoursLabel,

tEmploymentStatusLabel, tQualificationLabel font color to white

SET tDisplayButton, tClearButton, tAddButton, tBackButton background color to

new Color(211, 211, 211)

SET Color White to all labels and Color Grey to all buttons

ADD tHeader, tIdLabel, tNameLabel, tAddressLabel, tWorkingTypeLabel,

tSalaryLabel, tSpecializationLabel, tWorkingHoursLabel, tEmploymentStatusLabel,

tPerformanceLabel, tQualificationLabel to tPanel

ADD tId, tName, tAddress, tWorkingType, tSpecialization, tPerformance, tSalary,

tWorkingHours, tEmploymentStatus, tQualification to tPanel

ADD tDisplayButton, tClearButton, tAddButton, tBackButton to tPanel

ADD tPanel to tutorFrame

END DO

DEFINE gradesAssignment() with return type void

DO

DECLARE JPanel gPanel with layout set to null

SET gPanel background color to new Color(47, 79, 79)

DECLARE JLabel gHeader, gIdLabel, gDepartmentLabel, gGradedScoreLabel,

gExperienceLabel

INITIALIZE gHeader, gIdLabel, gDepartmentLabel, gGradedScoreLabel,

gExperienceLabel and set bounds

DECLARE JTextField gId, gDepartment, gGradedScore, gExperience

INITIALIZE gId, gDepartment, gGradedScore, gExperience and set bounds.

DECLARE JButton gGradeButton, gClearButton, gBackButton

INITIALIZE gGradeButton with text "Grade, gClearButton with text

"Clear" and gBackButton with text "<-Back-" and set bounds

ADD action listener to gGradeButton

DO

IMPLEMENT try-catch block

DO

IF any of the text fields are empty

DO

SHOW error message dialog "Empty Field Found! Please fill required area."

END DO

ELSE

DO

PARSE input to integers and strings and store in respective variables

IF id, department, experience, and graded score are non-negative

DO

FOR each teacher in listOfTeacher

DO

IF teacher's id matches idCheck and teacher is instance of Lecturer

DO

SET resultingGrade to result of gradeAssignment method of

Lecturer with parameters gradedScoreCheck, departmentCheck,

experienceCheck

END DO

END DO

IF resultingGrade is empty

DO

SHOW error message dialog "Lecturer id doesn't exist. Input correct id."

END DO

ELSE

DO

IF resultingGrade is "A" or "B" or "C" or "D" or "E"

DO

DECLARE gradedMessage as string "You have scored " +

resultingGrade + "!!"

SHOW message dialog gradedMessage

CLEAR all text fields

END DO

ELSE

DO

SHOW message dialog resultingGrade

END DO

END DO

END DO

ELSE

DO

SHOW message dialog "Please enter positive values."

END DO

END DO

END DO

CATCH NumberFormatException

DO

SHOW error message dialog "Number Format Exception. Enter number only in

number-required fields."

END DO

ADD action listener to gClearButton

DO

IF any text field is not empty

DO

DISPLAY confirmation dialog "Do you really want to clear all fields?"

IF user chooses yes

DO

CLEAR all text fields

END DO

END DO

END DO

ADD action listener to gBackButton

DO

CLEAR all text fields

HIDE gradeFrame

SHOW buttonsFrame

END DO

// Change font size and color for labels and buttons

DECLARE Font labelFont, headerFont, buttonFont

SET labelFont to gIdLabel font with size 16 and bold

SET headerFont to gHeader font with size increased by 80%

SET buttonFont to gClearButton font with size increased by 20%

SET gHeader font color to white

SET gIdLabel, gDepartmentLabel, gGradedScoreLabel, gExperienceLabel font

color to white

SET gGradeButton, gClearButton, gBackButton background color to new

Color(211, 211, 211)

ADD gHeader, gIdLabel, gDepartmentLabel, gGradedScoreLabel,

gExperienceLabel to gPanel

ADD gId, gDepartment, gGradedScore, gExperience to gPanel

ADD gGradeButton, gClearButton, gBackButton to gPanel

ADD gPanel to gradeFrame

END DO

DEFINE setsSalary() with return type void

DO

DECLARE JPanel sPanel with layout set to null

SET sPanel background color to new Color(47, 79, 79)

DECLARE JLabel sHeader, sIdLabel, sPerformanceLabel, sSalaryLabel

INITIALIZE sHeader with text "Enter Details", sIdLabel with text "Id: ",

sPerformanceLabel with text "Performance: “, sSalaryLabel and set bounds

DECLARE JTextField sId, sPerformance, sSalary

INITIALIZE sId, sPerformance, sSalary and set bounds

DECLARE JButton sSetSalaryButton, sClearButton, sBackButton

INITIALIZE sSetSalaryButton with text "Set", sClearButton with text "Clear”,

sBackButton with text "<-Back-" and set bounds

ADD action listener to sSetSalaryButton

DO

IMPLEMENT try-catch block

DO

IF any of the text fields are empty

DO

SHOW error message dialog "Empty Field Found! Please fill required

area."

END DO

ELSE

DO

PARSE input to integers and doubles

IF id, performance, and salary are non-negative

DO

PARSE input to integers and strings and store in respective

variables

FOR each teacher in listOfTeacher

DO

IF teacher's id matches idCheck and teacher is instance of Tutor

DO

SET updatedSalary to result of setSalary method of Tutor with

parameters salaryCheck and performanceCheck

END DO

END DO

IF updatedSalary is empty

DO

SHOW error message dialog "Tutor id doesn't exist. Input correct id."

END DO

ELSE

DO

IF performanceCheck > 5

DO

IF performanceCheck > 10

DO

SHOW message dialog updatedSalary

END DO

ELSE

DO

DECLARE salaryMessage as string "The tutor's updated salary is " + updatedSalary

SHOW message dialog salaryMessage

CLEAR all text fields

END DO

END DO

ELSE IF performanceCheck < 5

DO

SHOW message dialog updatedSalary

ELSE

DO

SHOW message dialog "Unexpected Error"

END DO

END DO

END DO

ELSE

DO

SHOW message dialog "Please enter positive values."

END DO

END DO

END DO

CATCH NumberFormatException

DO

SHOW error message dialog "Number Format Exception. Enter number

only in number-required fields."

END DO

ADD action listener to sClearButton

DO

IF any text field is not empty

DO

DISPLAY confirmation dialog "Do you really want to clear all fields?"

IF user chooses yes

DO

CLEAR all text fields

END DO

END DO

END DO

ADD action listener to sBackButton

DO

CLEAR all text fields

HIDE salaryFrame

SHOW buttonsFrame

END DO

DECLARE Font labelFont, headerFont, buttonFont

SET labelFont to sIdLabel font with size 16 and bold

SET headerFont to sHeader font with size increased by 80%

SET buttonFont to sClearButton font with size increased by 20%

SET sHeader font color to white

SET sIdLabel, sPerformanceLabel, sSalaryLabel font color to white

SET sSetSalaryButton, sClearButton, sBackButton background color to new

Color(211, 211, 211)

ADD sHeader, sIdLabel, sPerformanceLabel, sSalaryLabel to sPanel

ADD sId, sPerformance, sSalary to sPanel

ADD sSetSalaryButton, sClearButton, sBackButton to sPanel

ADD sPanel to salaryFrame

END DO

DEFINE removesTutor() with return type void

DO

DECLARE JPanel rPanel with layout set to null

SET rPanel background color to new Color(47, 79, 79)

DECLARE JLabel rHeader, rIdLabel

DECLARE JTextField rId, rHeader, rIdLabel, rId and set

DECLARE JButton rRemoveButton, rBackButton

INITIALIZE rRemoveButton with text "Remove, rBackButton with text "<-Back-"

and set bounds

DECLARE Font labelFont, headerFont, buttonFont

SET labelFont to rIdLabel font with size 16 and bold

SET headerFont to rHeader font with size increased by 80%

SET buttonFont to rRemoveButton font with size increased by 20%

SET rHeader font color to white

SET rIdLabel font color to white

SET rRemoveButton, rBackButton background color to new Color(211, 211,

211)

ADD action listener TO rRemoveButton

DO

IMPLEMENT try-catch block

DO

IF rId text field is empty

DO

SHOW error message dialog "Empty Field Found! Please fill required

area."

END DO

ELSE

DO

PARSE input to integer and store in repective variables

IF idR is non-negative

DO

DECLARE isRemoved as Boolean, set to false

FOR each teacher in listOfTeacher

DO

IF teacher's id matches idR and teacher is instance of Tutor

DO

REMOVE teacher from listOfTeacher

SET isRemoved to true

END DO

END DO

IF isRemoved is true

DO

SHOW message dialog "The tutor has been removed successfully"

CLEAR rId text field

END DO

ELSE

DO

SHOW error message dialog "Enter valid Tutor Id."

END DO

END DO

ELSE

DO

SHOW message dialog "Please enter positive values."

END DO

END DO

END DO

CATCH NumberFormatException  
 DO

SHOW error message dialog "Number Format Exception. Enter number only in number-required fields."

END DO

END DO

ADD action listener TO rBackButton

DO

CLEAR rId text field

HIDE removeFrame

SHOW buttonsFrame

END DO

ADD rHeader, rIdLabel, rId, rRemoveButton, rBackButton to rPanel

ADD rPanel to removeFrame

END DO

DEFINE main method

DO

INITIALIZE TeacherGUI instance using SwingUtilities.invokeLater

END DO

END DO

# Method Description

1. **TeacherGUI()** : It is a constructor for class TeacherGUI. A constructor is a special type of method which has the same name as the class and is used to ensure that the programmer doesn’t forget to call method to pass data, dodging compile time error. It has no parameters and return type is void. Almost all the code has been written inside the constructor or other functions have been called inside it.
2. **lecturersFrame()** : is a method where the implementation of frame where lecturer’s data will be entered. It has no parameters and is of void return type; doesn’t return anything. When this method is called, a new JPanel will be created, asking the user for input of multiple datas of lecturer. This panel also consists of buttons for adding lecturer, displaying list of added lecturer, clearing fields and back button to return to main frame.
3. **tutorsFrame()** : is a method where the implementation of frame where tutor’s data will be entered. It has no parameters and is of void return type; doesn’t return anything. When this method is called, a new JPanel will be created, asking the user for input of multiple datas of tutor. This panel also consists of buttons for adding tutor, displaying list of added tutor, clearing fields and back button to return to main frame.
4. **displayLecturerList()** : It is a method where the implementation of frame where lecturer’s data will be displayed. It has no parameters and is of void return type; doesn’t return anything. When this method is called, a new JPanel is created which consists of a JTextArea with JScrollingPane; which makes it scrollable. It displays the list of added lecturer if data was entered.
5. **displayTutorList()** : It is a method where the implementation of frame where tutor’s data will be displayed. It has no parameters and is of void return type; doesn’t return anything. When this method is called, a new JPanel is created which consists of a JTextArea with JScrollingPane; which makes it scrollable. It displays the list of added tutorif data was entered.
6. **gradesAssignment()** : It is a method where the implementation of frame where lecturer will grade students and the data will be displayed. It has no parameters and is of void return type; doesn’t return anything. When this method is called, a new JPanel is created which consists of JLabels and JTextFields asking user for input and if conditions are met, the graded score is displayed.
7. **setsSalary()** : It is a method where the implementation of frame where tutor’s salary will be set and the data will be displayed. It has no parameters and is of void return type; doesn’t return anything. When this method is called, a new JPanel is created which consists of JLabels and JTextFields asking user for input and if conditions are met; adequate performance index and working hours; the tutor’s salary will be updated and shown.
8. **removesTutor():** : It is a method where the implementation of frame where a specified tutor will be removed. It has no parameters and is of void return type; doesn’t return anything. When this method is called, a new JPanel is created which consists of a JLabel and JTextField asking user for id of tutor to remove and if conditions are met; valid tutor id; the tutor will be removed from the Arraylist.
9. **Main method:** Main method is one of the most significant methods in Java. It’s where the process of object creation is done. An object of type TeacherGUI is created inside an EDT runnable in this method which should always be done when handling heavy GUIs events such as mouse clicks, keyboard inputs and many more so that there is proper GUI responsiveness and multiple tasks can be handled simultaneously.
10. **actionPerformed() :** It is a method mostly used in GUIs to handle user interaction. When a specific button is clicked, the associated Listener’s actionPerformed method runs and the code inside is executed.
    1. **Add lecturer :** When this button is clicked, the getText() method extracts user input and adds it to the arraylist listOfTeacher.
    2. **Add tutor : :** When this button is clicked, the getText() method extracts user input and adds it to the arraylist listOfTeacher.
    3. **Clear :** The clear button listener is also in each frame of lecturer, tutor, grade assignment and set salary which when clicked removes the data entered by user from the text-field. This is a quality of life button which makes it easier to remove all entered data at once, saving manual labor.

# Testing

## Test 1: Checking if the program can be compiled and ran through the command prompt.

|  |  |
| --- | --- |
| Objective | To check If program compiles and runs through command prompt. |
| Action | Open the command prompt from the location of files, and inputting commands, javac TeacherGUI.java to compile and java TeacherGUI to run. |
| Expected Result | The TeacherGUI class should be successfully compiled and run. |
| Actual Result | The TeacherGUI class was successfully compiled and run. |
| Conclusion | The test was successful. |

Table 5: Test 1

### Image proving the test result

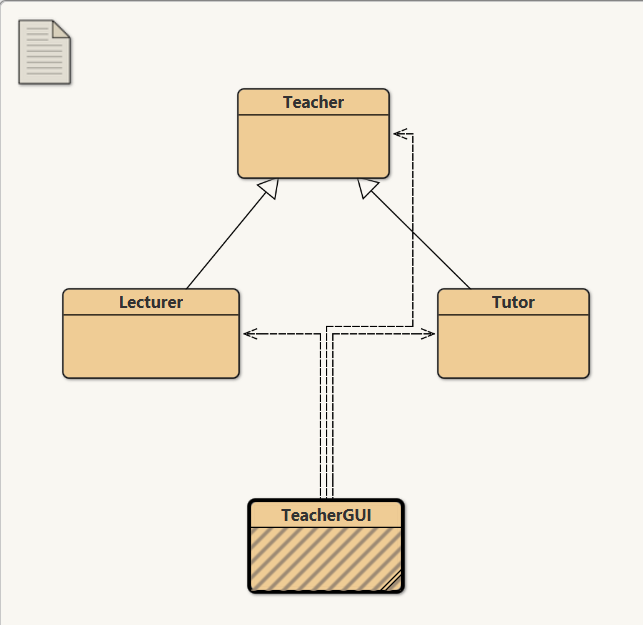


Figure 1: Image proving test 1.1

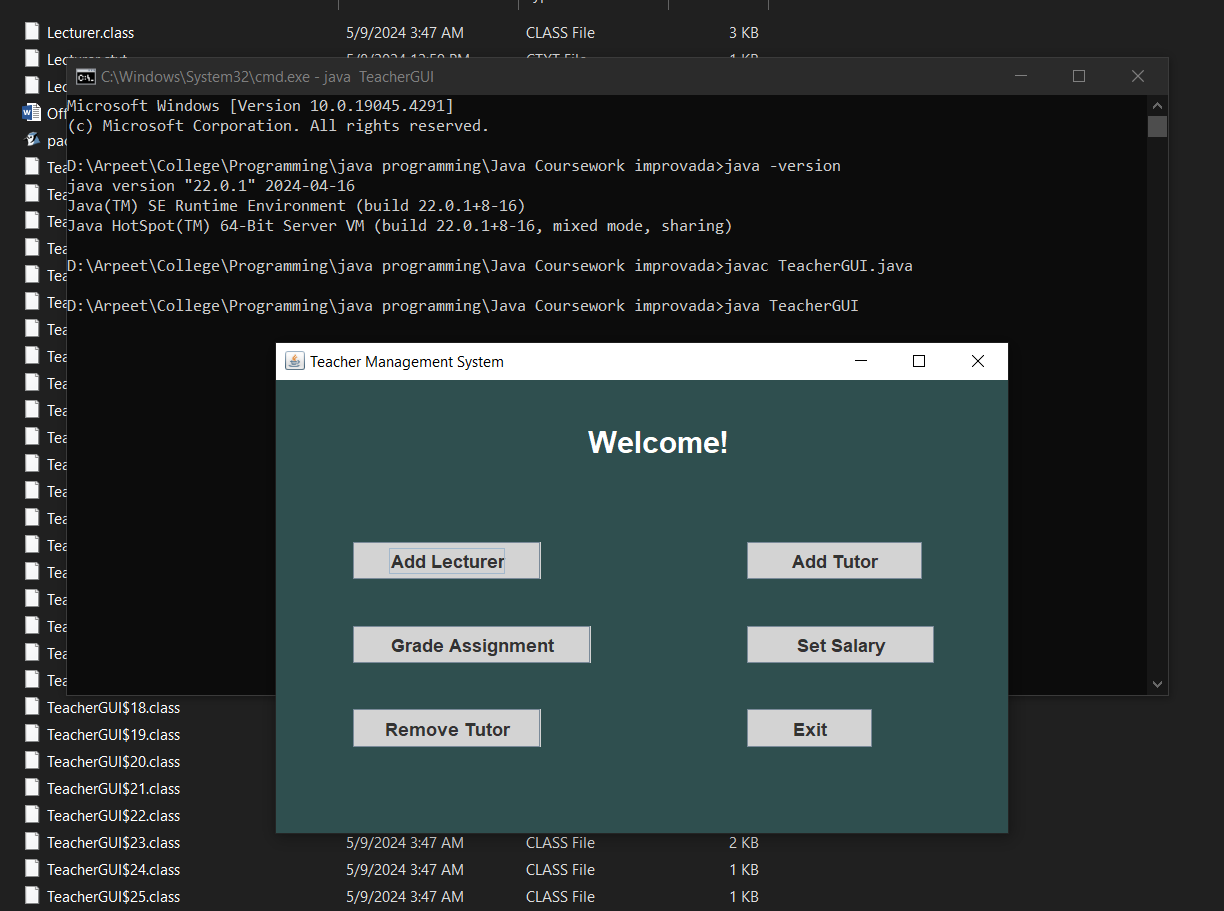


Figure 2: Image proving test 1.2

## Test 2: To inspect the effects of Add lecturer, Add Tutor, Grade Assignment, Set Salary and Remove Tutor.

|  |  |
| --- | --- |
| Objective | To inspect the effects of Add lecturer, Add Tutor, Grade Assignment, Set Salary and Remove Tutor. |
| Action | Respective required values are inserted into each frame and added / graded / salary set / removed. |
| Expected Result | The lecturer and tutor should be added, assignment should be graded, salary should be set and tutor must be removed. |
| Actual Result | The lecturer and tutor was added, assignment was graded, salary was set and tutor was removed. |
| Conclusion | The test was successful. |

Table 6: Test 2

### Images proving test result

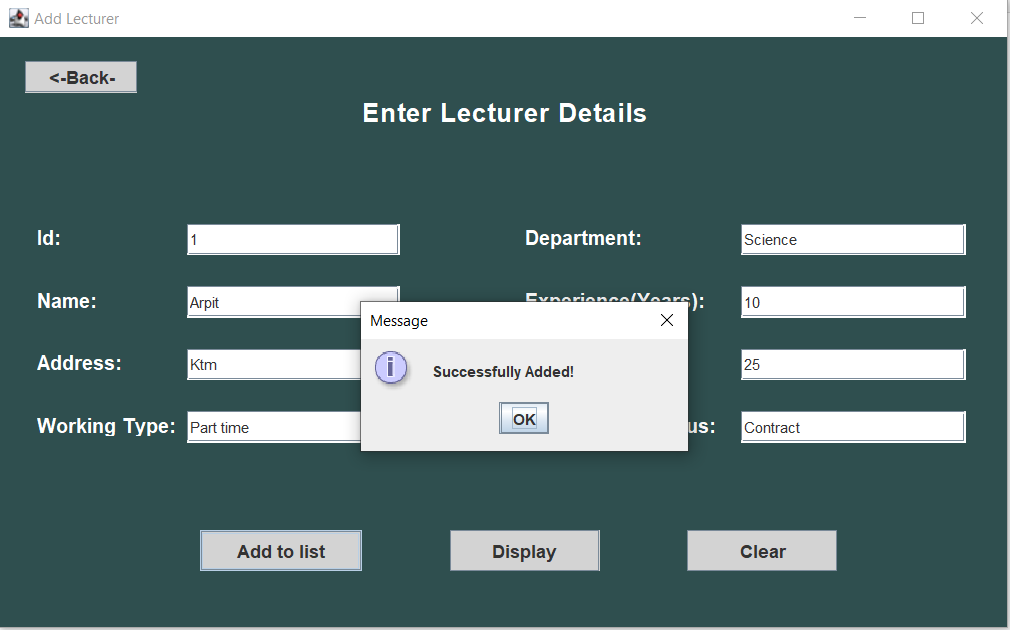


Figure 3: Image proving test 2.1

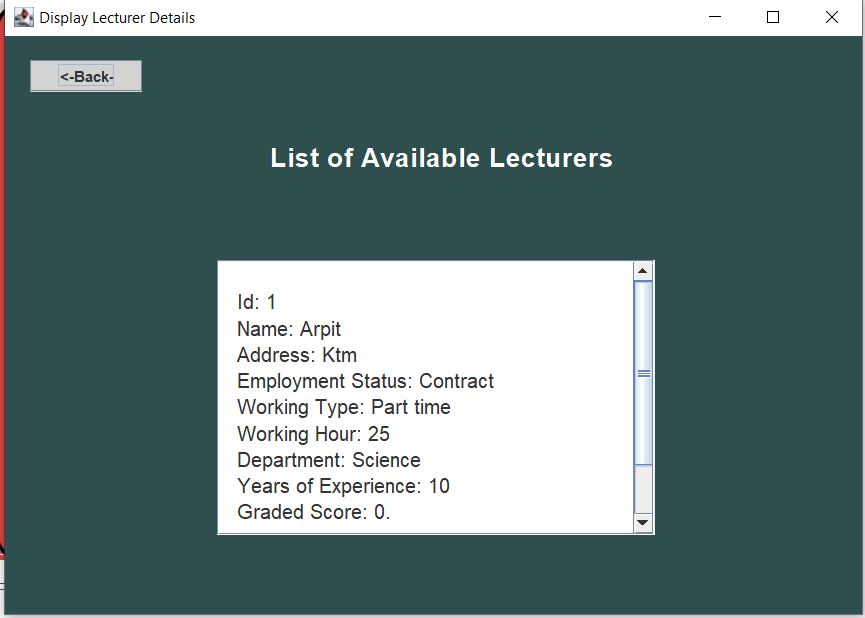


Figure 4: Image proving test 2.2

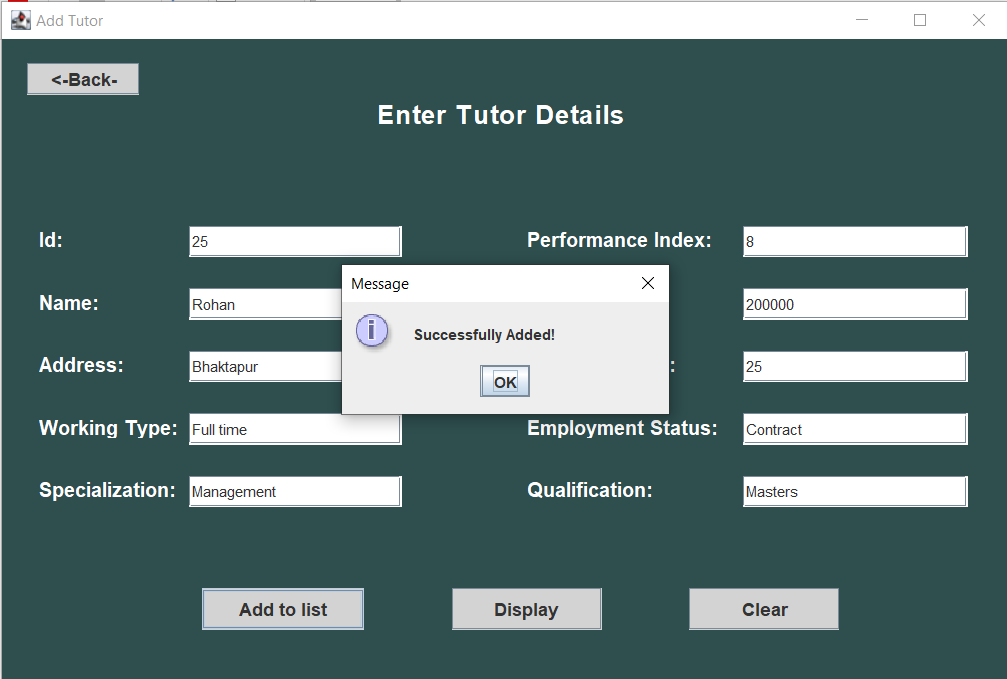


Figure 5: Image proving test 2.3

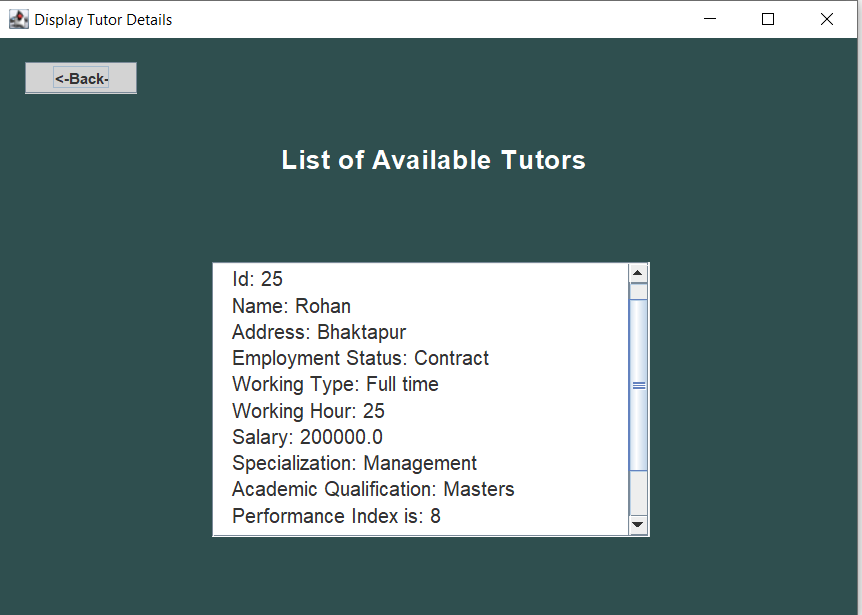


Figure 6: Image proving test 2.4

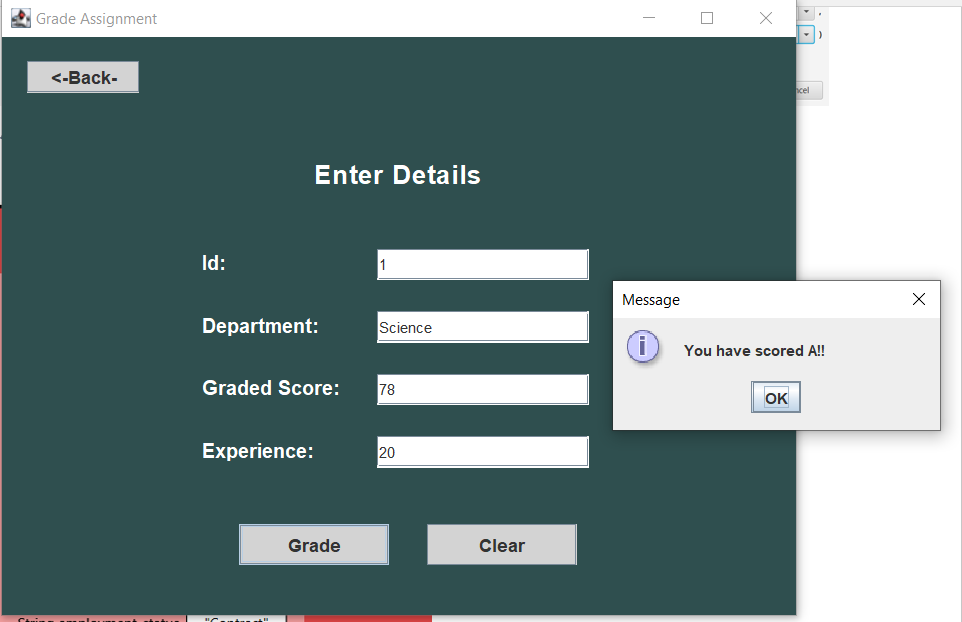


Figure 7: Image proving test 2.5

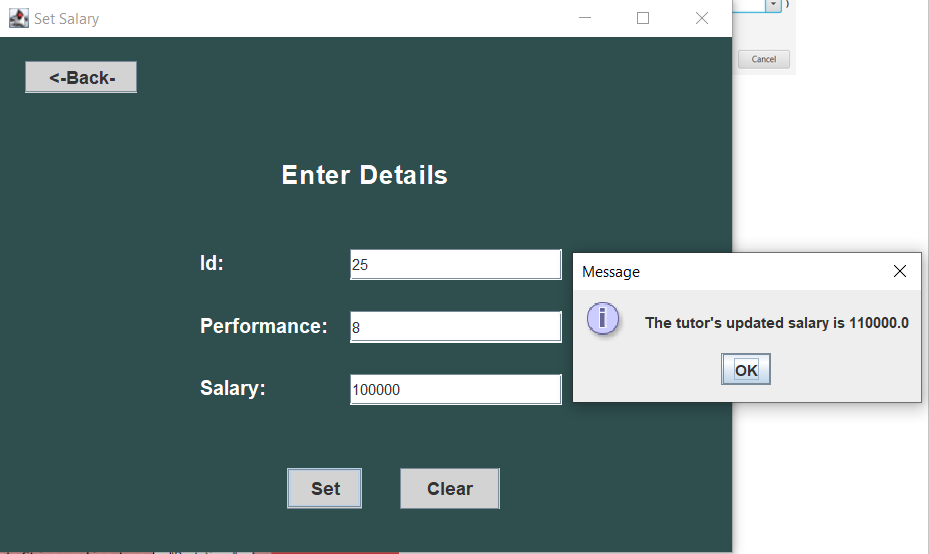


Figure 8: Image proving test 2.6

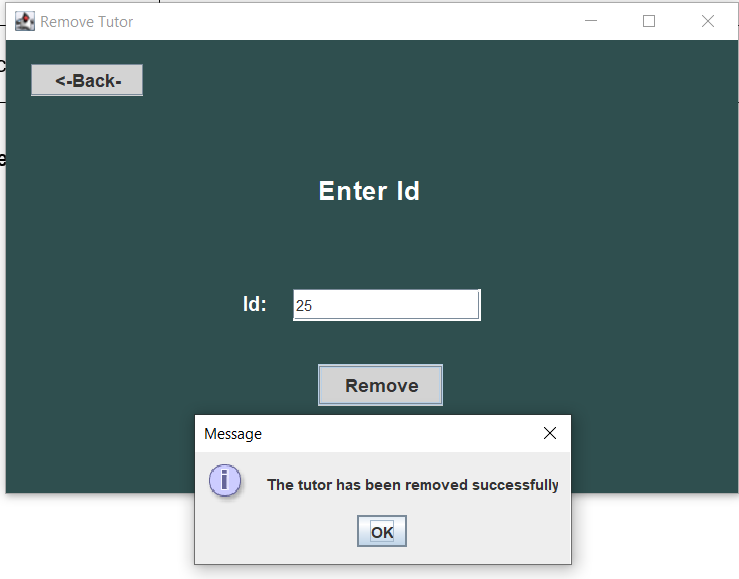


Figure 9: Image proving test 2.7

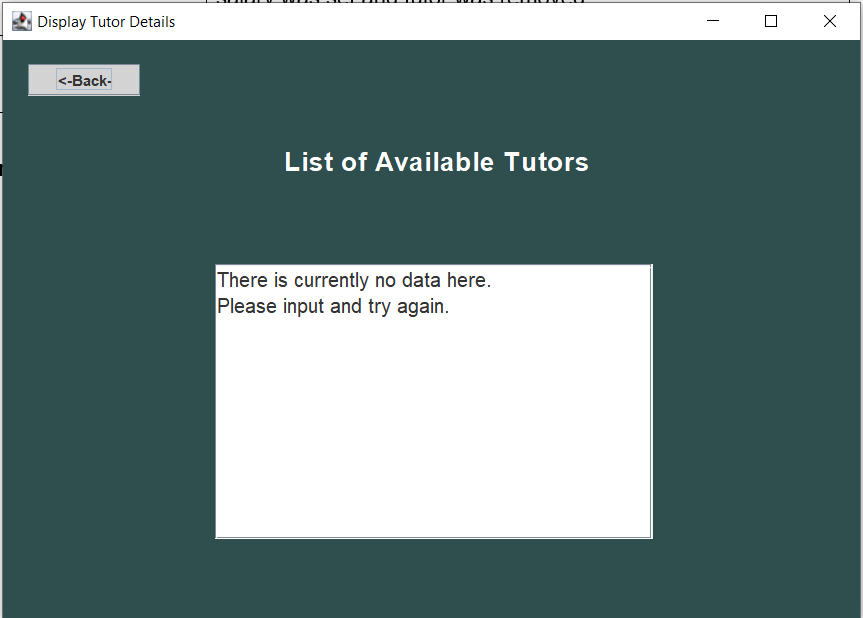


Figure 10: Image proving test 2.8

## Test 3: To insure that appropriate dialog boxes appear when unsuitable values are entered for Teacher ID.

|  |  |
| --- | --- |
| Objective | To insure that appropriate dialog boxes appear when unsuitable values are entered for Teacher ID. |
| Action | Unsuitable values are entered for Teacher Id. |
| Expected Result | Appropriate dialog boxes should appear and give suitable message to the user. |
| Actual Result | Appropriate dialog boxes appeared and gave suitable message to the user. |
| Conclusion | The test was successful. |

Table 7: Test 3

### Images proving test result

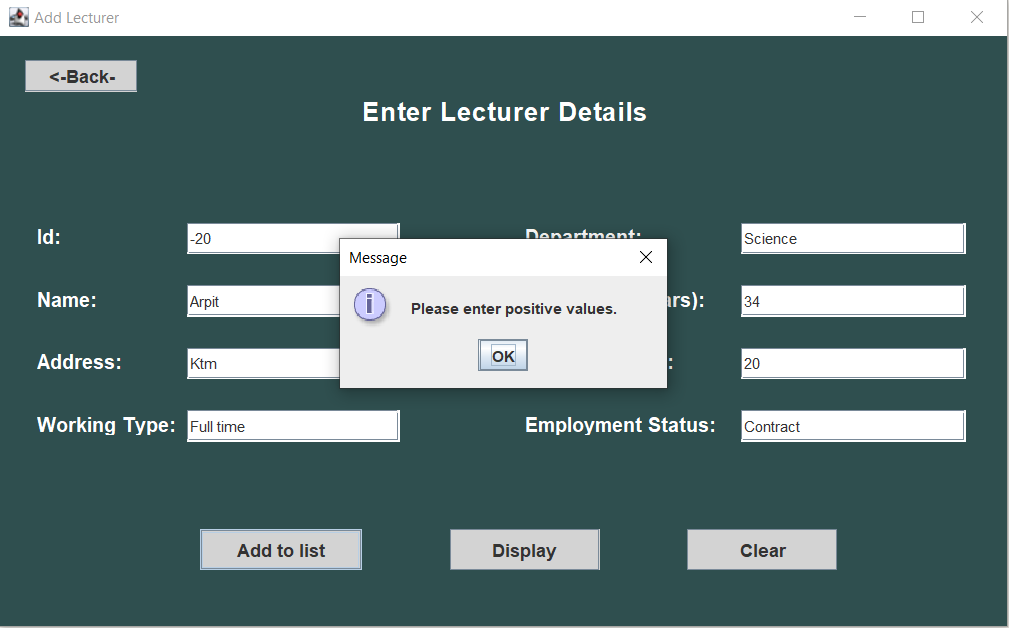


Figure 11: Image proving test 3.1

### 

Figure 12: Image proving test 3.2

# Error, It’s Types and Detection

Errors are blunders or mistakes that happens when you are coding. It is not a bad thing as making errors is how we grow and learn. There are exactly 3 types of errors and they are categorized based on their cause. They are: Syntax Error, Semantic Error and Logical Error.

## Syntax Error

An error which occurs when the source code doesn’t follow grammatical rules of programming (called syntax) and hence, contains mistakes. (Neupane, 2023); self-reference

Some of the examples of syntax errors are:

1. Misspelling keywords – Eg: publc instead of public.
2. Opening a parenthesis or a block (eg: if block) and forgetting to close it.
3. Missing the use of semi colon (;).

A **syntax error** I came across in my program was:

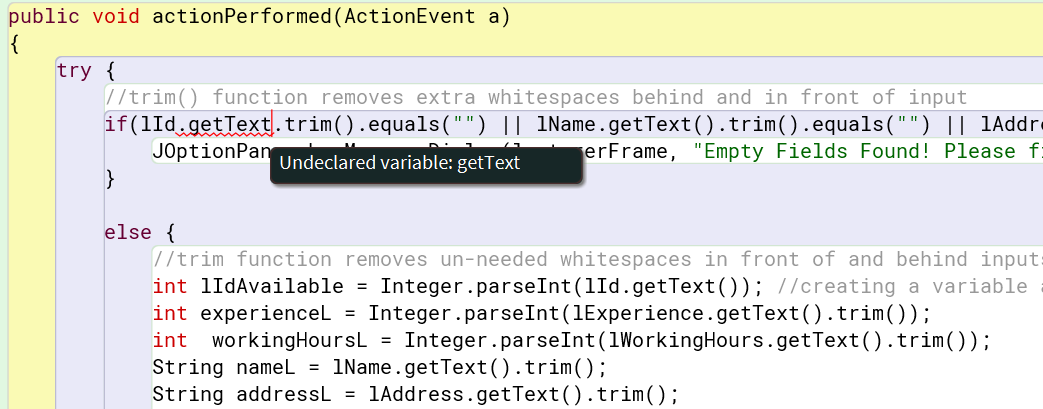


Figure 13: Syntax Error 1

Here, getText isn’t being recognized as a variable not because it was undeclared, but because I was missing the parenthesis a function is supposed to have.

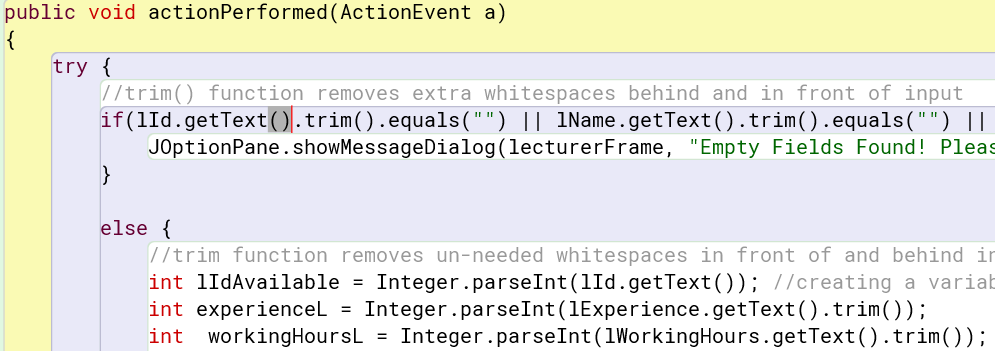


Figure 14: Syntax Error 2

The error is solved after putting parenthesis () after getText function.

## Logical Error

A logical error occurs when there is flaw in the applied logic when solving the problem.

Some examples of logical error are:

1. Incorrectly applying logic for a loop, example: when iterating over an array, one should start from index 0 but can be forgotten sometimes.
2. Applying incorrect conditions on if - else if - else ladder.
3. Missing certain conditions when solving a problem, example: when calculating factorial, one may forget to handle case where the user may input zero.

A **logical error** I came across in my program was:

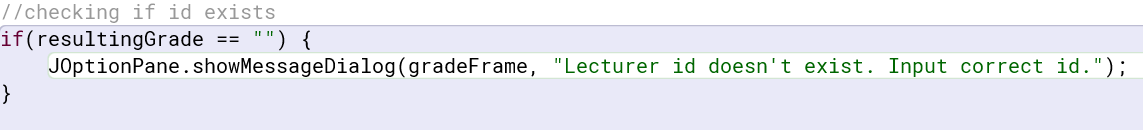


Figure 15: Logical Error 1

Here, I was trying to compare resultingGrade, which is a string, with an empty string using == operator which is a logical error.

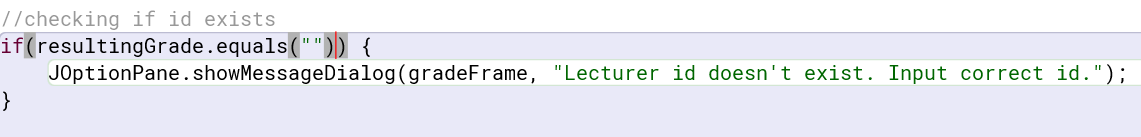


Figure 16: Logical Error 2

This error was solved by using the function equals(“”) to check if resultingGrade was empty.

## Runtime Error

Runtime errors are also known as **exceptions**. Sometimes when a program is written and compiled, there seems to be no syntax error. But while the program is running, errors are found which abruptly stops the program.

Some examples of runtime errors are:

1. NullPointerException: Occurs when a program tries to access a variable o object that is not referencing anything or has null reference.
2. NumberFormatException: Occurs when incorrect datatype is entered during input.

A **runtime error** I came across in my program was:

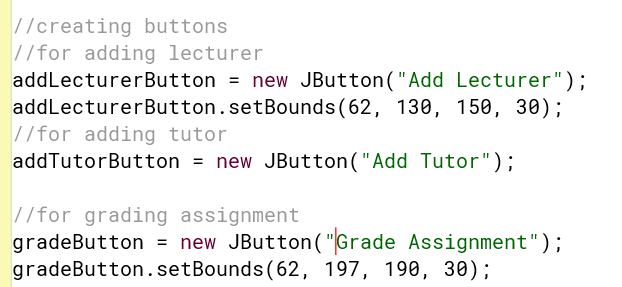


Figure 17: Runtime Error 1

Here, I had forgotten to setBounds to addTutorButton which caused the button to not appear on the GUI screen as shown below.

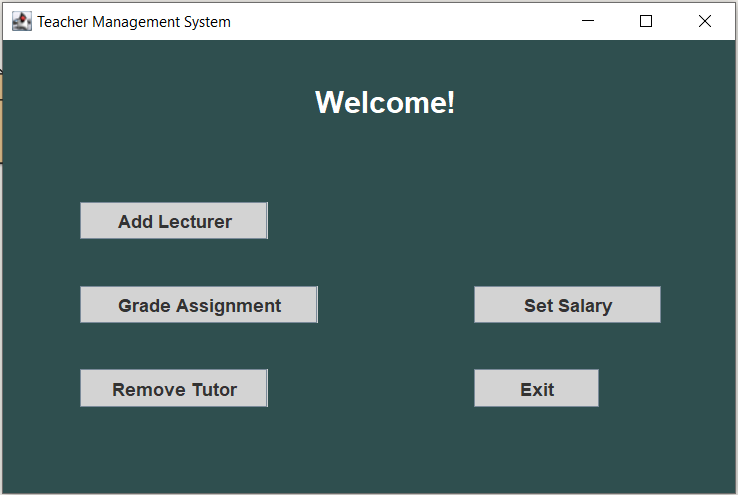


Figure 18: Runtime Error 2

The error was solved after the setting bounds of the button as shown below:

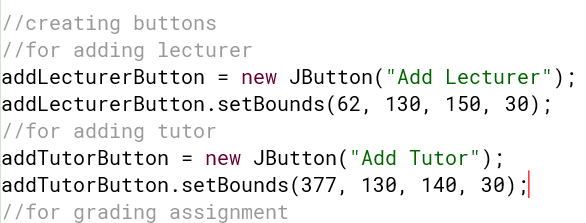


Figure 19: Runtime Error 3

# Changes to the old program

## Teacher

## 

Figure 20: Change 1

Before, there were print statements there to display the output. This time, I have created an empty string, concatenated all the result to the string and returned it to use it in the display section of TeacherGUI.

## Tutor

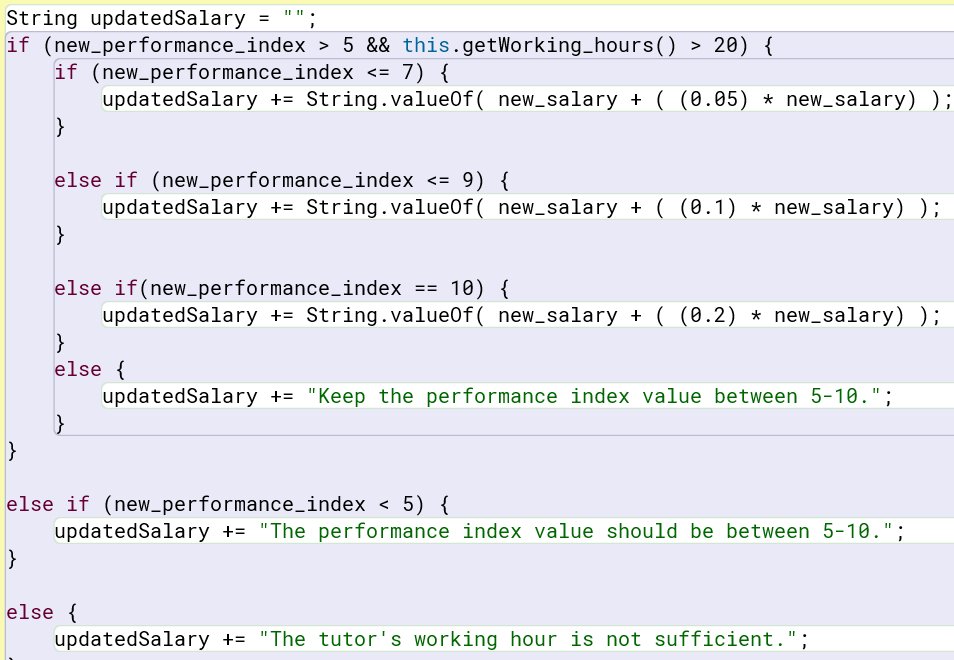


Figure 21: Change 2

Here, before there were print statements and redefining of new\_salary. This time, I have created an empty string, concatenated result to the string and returned it to use it in the set salary section of TeacherGUI.

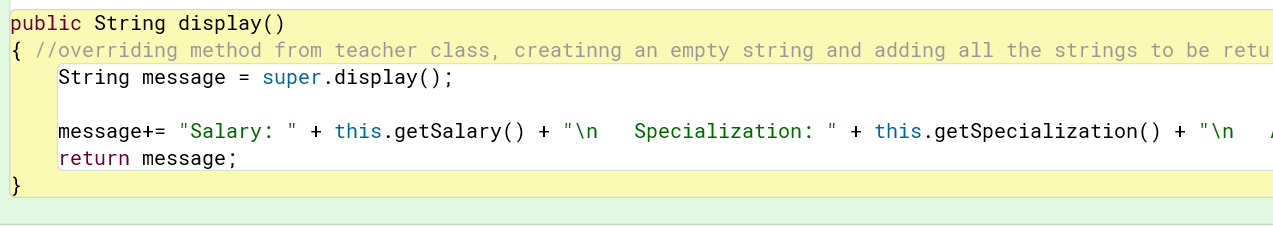


Figure 22:Change 3

Here as well, the print statements have been replaced by creation of a string which inherits the display function of Teacher class, concatenatation all the result and returned it to use it in the display section of TeacherGUI.

## Lecturer

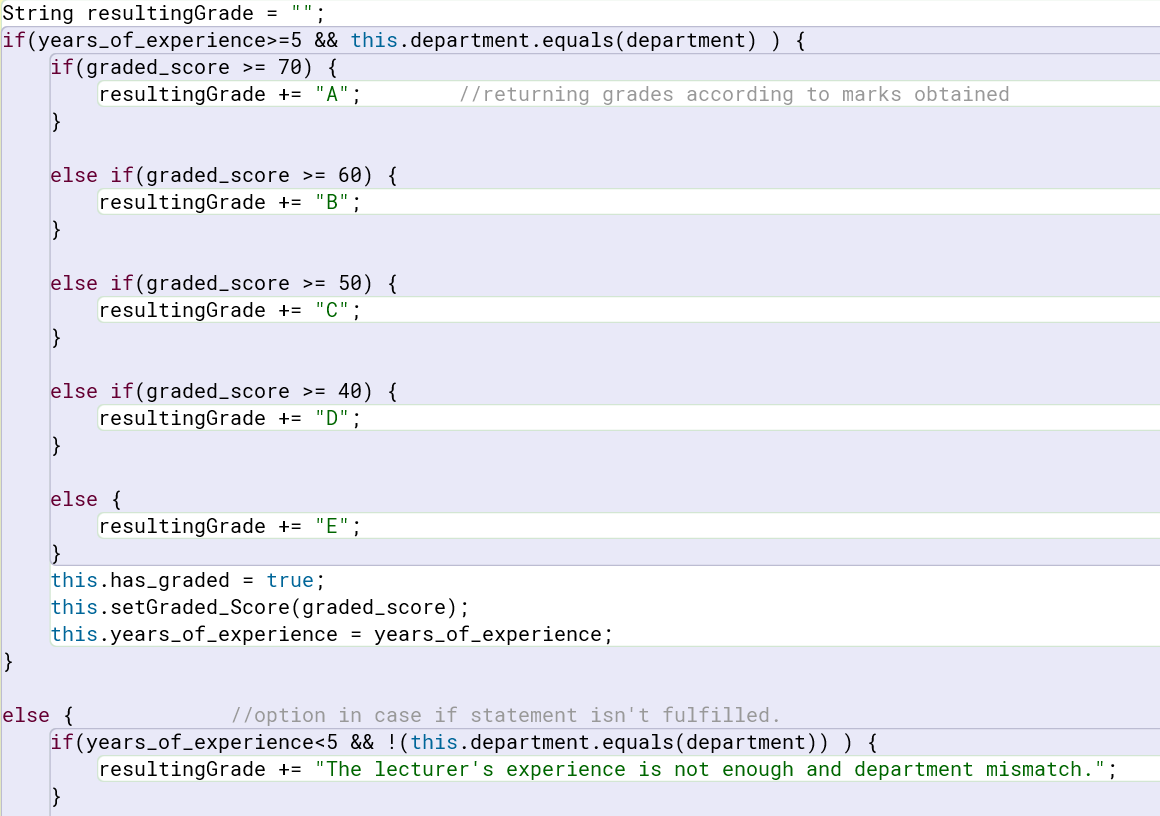


Figure 23: Change 4

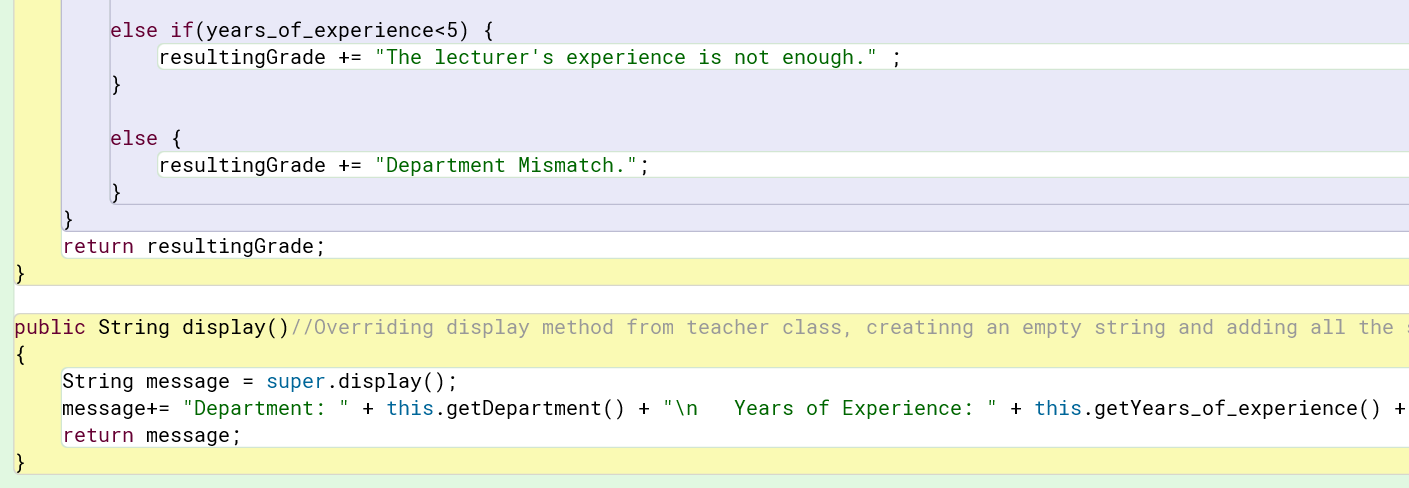


Figure 24: Change 5

Here, before there were print statements showing grade according to graded\_score. This time, I have created an empty string, concatenated result or appropriate message in case of condition mismatch, to the string and returned it to use it in the grade assignment section of TeacherGUI.

Before, there were print statements there to display the output. This time, I have created a string which inherits the display function of Teacher class, concatenated all the result to the string and returned it to use it in the display section of TeacherGUI.

# Conclusion

Through learning is how we gain new experiences, skills, and knowledge. Sometimes, It happens naturally, through experiences; something akin to child learning to speak or a bird learning to fly. We mostly copy our close ones the most and imitate their behaviour, which is why it’s said that a child will turn out to be a good person if the parents are decent. (Neupane, 2023); self-reference

Through this coursework, I really learned a lot of ways to not only properly create GUI based applications but it has also cleared up a lot of doubts I had when I first started leaning about this topic. Even those topics which I had knowledge of but no practical knowledge is now crystal clear to me. Through the coursework, I’ve also expanded my research skills and through self-research and self-study, it feels as though I’ve learned a lot more than I would have through my teachers.

There were lots of challenges throughout the development of this project and through research skills and self-study, those problems were solved. There was a massive satisfaction whenever I solved an error that was wasting lots of time.

I would like to express my gratitude to the teachers and staff who came up with this coursework as well as my friends and colleagues who cleared me up on some concepts.

This coursework has equipped me with essential skills and ideas which I hope goes past just completing assignments and actually building enterprise level projects in the near future.

# Bibliography

# Bibliography

Neupane, A., 2023. *Documentation on Project About Inheritance,* Bhaktapur: Arpit Neupane.

Oracle, 2007. *About BlueJ.* [Online]   
Available at: https://www.bluej.org/about.html  
[Accessed 9 May 2024].

Rock, U. L., 2022. *Microsoft Word.* [Online]   
Available at: https://ualr.edu/itservices/applications/v/microsoft-word/#:~:text=Microsoft%20Word%20is%20a%20word,both%20simple%20and%20complex%20documents.  
[Accessed 9 May 2024].

Rogers, K., 2023. *Computer Data Types.* [Online]   
Available at: https://study.com/learn/lesson/computer-data-types-sources.html

# Appendix

## Teacher

public class Teacher

{

private int teacher\_id;

private String teacher\_name;

private String address; //instance variables

private String working\_type;

private String employment\_status;

private int working\_hours;

public Teacher(int teacher\_id, String teacher\_name, String address, String working\_type, String employment\_status) //Constructor

{

this.teacher\_id = teacher\_id;

this.teacher\_name = teacher\_name;

this.address = address;

this.working\_type = working\_type;

this.employment\_status = employment\_status;

}

public int getTeacher\_id() //accessor method for teacher\_id

{

return this.teacher\_id;

}

public String getTeacher\_name() //accessor method for teacher\_name

{

return this.teacher\_name;

}

public String getAddress() //accessor method for address

{

return this.address;

}

public String getWorking\_type() //accessor method for working\_type

{

return this.working\_type;

}

public String getEmployment\_status() //accessor method for employment\_status

{

return this.employment\_status;

}

public int getWorking\_hours() //accessor method for working hours

{

return this.working\_hours;

}

public void setWorking\_Hours(int working\_hours) //mutator method for working\_hours

{

this.working\_hours = working\_hours;

}

public String display() //creatinng an empty string and adding all the strings to be returned to it and then returning it

{

String message = "";

message+= "\n Id: " + this.getTeacher\_id() + "\n Name: " + this.getTeacher\_name() + "\n ";

message+= "Address: " + this.getAddress() + "\n Employment Status: " + this.getEmployment\_status() + "\n ";

message+= "Working Type: " + this.getWorking\_type() + "\n Working Hour: " + this.getWorking\_hours() + "\n ";

return message;

}

}

## Lecturer

public class Lecturer extends Teacher //inheritance

{

private String department;

private int years\_of\_experience;

private int graded\_score; //instance variables.

private boolean has\_graded;

public Lecturer(int teacher\_id, String teacher\_name, String address, String working\_type, int working\_hours, String employment\_status, String department, int years\_of\_experience)

{ //constructor^

super(teacher\_id, teacher\_name, address, working\_type, employment\_status);

this.setWorking\_Hours(working\_hours);

this.department = department;

this.years\_of\_experience = years\_of\_experience;

this.graded\_score = 0;

this.has\_graded = false;

}

public String getDepartment() //accessor method for department

{

return this.department;

}

public int getYears\_of\_experience() //accessor method for years of experience

{

return this.years\_of\_experience;

}

public int getGraded\_score() //accessor method for graded\_score

{

return this.graded\_score;

}

public boolean isHas\_graded() //accessor method for has\_graded

{

return this.has\_graded;

}

public void setGraded\_Score(int graded\_score) //mutator method for graded\_score

{

this.graded\_score = graded\_score;

}

public String gradeAssignment(int graded\_score, String department, int years\_of\_experience) //method gradeAssignment

{

String resultingGrade = "";

if(years\_of\_experience>=5 && this.department.equals(department) ) {

if(graded\_score >= 70) {

resultingGrade += "A"; //returning grades according to marks obtained

}

else if(graded\_score >= 60) {

resultingGrade += "B";

}

else if(graded\_score >= 50) {

resultingGrade += "C";

}

else if(graded\_score >= 40) {

resultingGrade += "D";

}

else {

resultingGrade += "E";

}

this.has\_graded = true;

this.setGraded\_Score(graded\_score);

this.years\_of\_experience = years\_of\_experience;

}

else { //option in case if statement isn't fulfilled.

if(years\_of\_experience<5 && !(this.department.equals(department)) ) {

resultingGrade += "The lecturer's experience is not enough and department mismatch.";

}

else if(years\_of\_experience<5) {

resultingGrade += "The lecturer's experience is not enough." ;

}

else {

resultingGrade += "Department Mismatch.";

}

}

return resultingGrade;

}

public String display()//Overriding display method from teacher class, creatinng an empty string and adding all the strings to be returned to it and then returning it

{

String message = super.display();

message+= "Department: " + this.getDepartment() + "\n Years of Experience: " + this.getYears\_of\_experience() + "\n Graded Score: " + this.getGraded\_score() + ".\n\n\n";

return message;

}

}

## Tutor

public class Tutor extends Teacher

{

private double salary;

private String specialization;

private String academic\_qualifications; //instance variables

private int performance\_index;

private boolean is\_certified;

public Tutor(int teacher\_id, String teacher\_name, String address, String working\_type, String employment\_status, int working\_hours, double salary, String specialization, String academic\_qualifications, int performance\_index)

{ //^Constructor

super(teacher\_id, teacher\_name, address, working\_type, employment\_status);

this.setWorking\_Hours(working\_hours);

this.salary = salary;

this.specialization = specialization;

this.academic\_qualifications = academic\_qualifications;

this.performance\_index = performance\_index;

this.is\_certified = false;

}

public double getSalary() //accessor method for salary

{

return this.salary;

}

public String getSpecialization() //accessor method for specialization

{

return this.specialization;

}

public String getAcademic\_qualifications() //accessor method for academic\_qualifications

{

return this.academic\_qualifications;

}

public int getPerformance\_index() //accessor method for performance\_index

{

return this.performance\_index;

}

public boolean getIs\_certified() //accessor method for is\_certified....get is used instead of typical naming convention as "is"

{ //is already in name

return this.is\_certified;

}

public String setSalary(double new\_salary, int new\_performance\_index) //mutator method for salary

{

String updatedSalary = "";

if (new\_performance\_index > 5 && this.getWorking\_hours() > 20) {

if (new\_performance\_index <= 7) {

updatedSalary += String.valueOf( new\_salary + ( (0.05) \* new\_salary) ); //we have a string type storing variable so converting salary to string type

}

else if (new\_performance\_index <= 9) {

updatedSalary += String.valueOf( new\_salary + ( (0.1) \* new\_salary) );

}

else if(new\_performance\_index == 10) {

updatedSalary += String.valueOf( new\_salary + ( (0.2) \* new\_salary) );

}

else {

updatedSalary += "Keep the performance index value between 5-10.";

}

}

else if (new\_performance\_index < 5) {

updatedSalary += "The performance index value should be between 5-10.";

}

else {

updatedSalary += "The tutor's working hour is not sufficient.";

}

this.salary = new\_salary;

this.performance\_index = new\_performance\_index;

return updatedSalary;

}

public String display()

{ //overriding method from teacher class, creatinng an empty string and adding all the strings to be returned to it and then returning it

String message = super.display();

message+= "Salary: " + this.getSalary() + "\n Specialization: " + this.getSpecialization() + "\n Academic Qualification: " + this.getAcademic\_qualifications() + "\n Performance Index is: " + this.getPerformance\_index() + "\n\n\n";

return message;

}

}

## TeacherGUI

import javax.swing.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.Color;

import java.awt.Font;

import java.util.ArrayList;

import java.awt.Dimension;

public class TeacherGUI {

//creating an arraylist of type Teacher

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

//components of main frame

private JFrame buttonsFrame;

private JPanel mainPanel;

private JLabel welcomeLabel;

private JButton addLecturerButton, addTutorButton, gradeButton, setSalaryButton, removeTutorButton, exitButton;

//components of Lecturer frame

private JFrame lecturerFrame;

private JPanel lPanel;

private JLabel lHeader, lIdLabel, lNameLabel, lAddressLabel, lWorkingTypeLabel, lEmploymentStatusLabel, lDepartmentLabel, lExperienceLabel, lWorkingHoursLabel;

private JTextField lId, lName, lAddress, lWorkingType, lEmploymentStatus, lDepartment, lExperience, lWorkingHours;

private JButton lDisplayButton, lClearButton, lAddButton, lBackButton;

//components of Tutor frame

private JFrame tutorFrame;

private JPanel tPanel;

private JLabel tHeader, tIdLabel, tNameLabel, tAddressLabel, tWorkingTypeLabel, tEmploymentStatusLabel, tWorkingHoursLabel, tSalaryLabel, tSpecializationLabel;

private JLabel tQualificationLabel, tPerformanceLabel;

private JTextField tId, tName, tAddress, tWorkingType, tEmploymentStatus, tWorkingHours, tSalary, tSpecialization, tQualification, tPerformance;

private JButton tDisplayButton, tClearButton, tAddButton, tBackButton;

//components of Grade Assignment frame

private JFrame gradeFrame;

private JPanel gPanel;

private JLabel gHeader, gIdLabel, gDepartmentLabel, gGradedScoreLabel, gExperienceLabel;

private JTextField gId, gDepartment, gExperience, gGradedScore;

private JButton gGradeButton, gClearButton, gBackButton;

//components of Set Salary frame

private JFrame salaryFrame;

private JPanel sPanel;

private JLabel sHeader, sSalaryLabel, sPerformanceLabel, sIdLabel;

private JTextField sSalary, sPerformance, sId;

private JButton sSetSalaryButton, sClearButton, sBackButton;

//components of Remove Tutor frame

private JFrame removeFrame;

private JPanel rPanel;

private JLabel rIdLabel, rHeader;

private JTextField rId;

private JButton rRemoveButton, rBackButton;

//components of lDisplay Frame

private JFrame lDisplayFrame;

private JPanel lDisplayPanel;

private JLabel dlHeader;

private JLabel lIdData, lNameData, lAddressData, lWorkingTypeData, lEmploymentStatusData, lDepartmentData, lExperienceData, lWorkingHoursData, lGradedScoreData;

private JButton dlBackButton;

//components of tDisplay Frame

private JFrame tDisplayFrame;

private JPanel tDisplayPanel;

private JLabel dtHeader;

private JButton dtBackButton;

public TeacherGUI() {

//creating frame

buttonsFrame = new JFrame("Teacher Management System");

buttonsFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); //terminates JVM when close button is clicked

buttonsFrame.setSize(600, 400); //setting size of frame

buttonsFrame.setLocationRelativeTo(null); //centers the frame when program runs

//creating panels

mainPanel = new JPanel(null);

//creating labels

welcomeLabel = new JLabel("Welcome!");

welcomeLabel.setBounds(250, 20, 200, 60); //setting bounds (x-axis, y-axis, width, height)

//creating buttons

//for adding lecturer

addLecturerButton = new JButton("Add Lecturer");

addLecturerButton.setBounds(62, 130, 150, 30);

//for adding tutor

addTutorButton = new JButton("Add Tutor");

addTutorButton.setBounds(377, 130, 140, 30);

//for grading assignment

gradeButton = new JButton("Grade Assignment");

gradeButton.setBounds(62, 197, 190, 30);

//for setting salary

setSalaryButton = new JButton("Set Salary");

setSalaryButton.setBounds(377, 197, 150, 30);

//for removing tutor

removeTutorButton = new JButton("Remove Tutor");

removeTutorButton.setBounds(62, 264, 150, 30);

//exiting frame

exitButton = new JButton("Exit");

exitButton.setBounds(377, 264, 100, 30);

//adding actionListeners

addLecturerButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

//creating another frame when add Lecturer button is clicked

lecturerFrame = new JFrame("Add Lecturer");

lecturerFrame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE); //different from exit on close; runs in the background

lecturerFrame.setSize(820, 510);

lecturerFrame.setLocationRelativeTo(buttonsFrame); //lays the frame relative to buttonsFrame

lecturersFrame(); //lecturersFrame is a method implemented outside of constructor and is being called here

lecturerFrame.setVisible(true); //makes lecturerFrame visible

buttonsFrame.setVisible(false); //and buttonsFrame invisible

}

});

addTutorButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

//creating another frame when add Tutor button is clicked

tutorFrame = new JFrame("Add Tutor");

tutorFrame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE); //different from exit on close; runs in the background

tutorFrame.setSize(820, 550);

tutorFrame.setLocationRelativeTo(buttonsFrame); //lays the frame relative to buttonsFrame

tutorsFrame();

tutorFrame.setVisible(true);

buttonsFrame.setVisible(false);

}

});

gradeButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

//creating another frame when grade assignment button is clicked

gradeFrame = new JFrame("Grade Assignment");

gradeFrame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE); //different from exit on close; runs in the background

gradeFrame.setSize(650, 500);

gradeFrame.setLocationRelativeTo(buttonsFrame); //lays the frame relative to buttonsFrame

gradesAssignment();

gradeFrame.setVisible(true);

buttonsFrame.setVisible(false);

}

});

setSalaryButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

//creating another frame when setSalary is clicked

salaryFrame = new JFrame("Set Salary");

salaryFrame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE); //different from exit on close; runs in the background

salaryFrame.setSize(600, 450);

salaryFrame.setLocationRelativeTo(buttonsFrame); //lays the frame relative to buttonsFrame

setsSalary();

salaryFrame.setVisible(true);

buttonsFrame.setVisible(false);

}

});

removeTutorButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

//creating another frame when removeTutor is clicked

removeFrame = new JFrame("Remove Tutor");

removeFrame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE); //different from exit on close; runs in the background

removeFrame.setSize(600, 400);

removeFrame.setLocationRelativeTo(buttonsFrame); //lays the frame relative to buttonsFrame

removesTutor();

removeFrame.setVisible(true);

buttonsFrame.setVisible(false);

}

});

exitButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

System.exit(0);

}

});

//changing the font size

//of welcomeLabel

Font welcomeFont = welcomeLabel.getFont();

welcomeFont = welcomeFont.deriveFont(welcomeFont.getSize() \* 2f); //increase font size by 200%

welcomeLabel.setFont(welcomeFont);

//increasing font size for buttons

Font buttonsFont = addLecturerButton.getFont();

buttonsFont = buttonsFont.deriveFont(addLecturerButton.getFont().getSize() \* 1.2f);

addLecturerButton.setFont(buttonsFont);

addTutorButton.setFont(buttonsFont);

gradeButton.setFont(buttonsFont);

setSalaryButton.setFont(buttonsFont);

removeTutorButton.setFont(buttonsFont);

exitButton.setFont(buttonsFont);

//setting text color

welcomeLabel.setForeground(Color.WHITE); //setting text color to white

//setting background color

mainPanel.setBackground(new Color(47, 79, 79)); //setting background to darkish green of panel

addLecturerButton.setBackground(new Color(211, 211, 211));

addTutorButton.setBackground(new Color(211, 211, 211));

gradeButton.setBackground(new Color(211, 211, 211)); //setting grey bgcolor to buttons

setSalaryButton.setBackground(new Color(211, 211, 211));

removeTutorButton.setBackground(new Color(211, 211, 211));

exitButton.setBackground(new Color(211, 211, 211));

//adding to panel

//adding labels to panel

mainPanel.add(welcomeLabel);

//adding buttons to panel

mainPanel.add(addLecturerButton);

mainPanel.add(gradeButton);

mainPanel.add(removeTutorButton);

mainPanel.add(addTutorButton);

mainPanel.add(setSalaryButton);

mainPanel.add(exitButton);

//adding the main panel to the frame's content pane

buttonsFrame.add(mainPanel);

buttonsFrame.setVisible(true); //finally, setting visiblity to true

}

//creating a method where all components of frame (where lecturer is added) is implemented

public void lecturersFrame() {

lPanel = new JPanel(null);

lPanel.setBackground(new Color(47, 79, 79)); //giving bgcolor

//creating labels and setting bounds

lHeader = new JLabel("Enter Lecturer Details ");

lHeader.setBounds(290, 30, 290, 60);

lIdLabel = new JLabel("Id: ");

lIdLabel.setBounds(30, 150, 50, 20);

lNameLabel = new JLabel("Name: ");

lNameLabel.setBounds(30, 200, 70, 20);

lAddressLabel = new JLabel("Address: ");

lAddressLabel.setBounds(30, 250, 100, 20);

lWorkingTypeLabel = new JLabel("Working Type: ");

lWorkingTypeLabel.setBounds(30, 300, 150, 20);

lDepartmentLabel = new JLabel("Department: ");

lDepartmentLabel.setBounds(420, 150, 100, 20);

lExperienceLabel = new JLabel("Experience(Years): ");

lExperienceLabel.setBounds(420, 200, 150, 20);

lWorkingHoursLabel = new JLabel("Working Hours: ");

lWorkingHoursLabel.setBounds(420, 250, 190, 20);

lEmploymentStatusLabel = new JLabel("Employment Status: ");

lEmploymentStatusLabel.setBounds(420, 300, 170, 20);

//creating text-fields

lId = new JTextField();

lId.setBounds(150, 150, 170, 25);

lName = new JTextField();

lName.setBounds(150, 200, 170, 25);

lAddress = new JTextField();

lAddress.setBounds(150, 250, 170, 25);

lWorkingType = new JTextField();

lWorkingType.setBounds(150, 300, 170, 25);

lDepartment = new JTextField();

lDepartment.setBounds(593, 150, 180, 25);

lExperience = new JTextField();

lExperience.setBounds(593, 200, 180, 25);

lWorkingHours = new JTextField();

lWorkingHours.setBounds(593, 250, 180, 25);

lEmploymentStatus = new JTextField();

lEmploymentStatus.setBounds(593, 300, 180, 25);

//creating buttons

lAddButton = new JButton("Add to list");

lAddButton.setBounds(160, 395, 130, 33);

lDisplayButton = new JButton("Display");

lDisplayButton.setBounds(360, 395, 120, 33);

lClearButton = new JButton("Clear");

lClearButton.setBounds(550, 395, 120, 33);

lBackButton = new JButton("<-Back-");

lBackButton.setBounds(20, 20, 90, 25);

//number format listener try-catch block

lAddButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent a)

{

try {

//trim() function removes extra whitespaces behind and in front of input

if(lId.getText().trim().equals("") || lName.getText().trim().equals("") || lAddress.getText().trim().equals("") || lWorkingType.getText().trim().equals("") || lExperience.getText().trim().equals("") || lDepartment.getText().trim().equals("") || lWorkingHours.getText().trim().equals("") || lEmploymentStatus.getText().trim().equals("") ) {

JOptionPane.showMessageDialog(lecturerFrame, "Empty Fields Found! Please fill all text areas. ", "Error", JOptionPane.ERROR\_MESSAGE);

}

else {

//trim function removes un-needed whitespaces in front of and behind inputs

int lIdAvailable = Integer.parseInt(lId.getText()); //creating a variable and putting lecturer id on it

int experienceL = Integer.parseInt(lExperience.getText().trim());

int workingHoursL = Integer.parseInt(lWorkingHours.getText().trim());

String nameL = lName.getText().trim();

String addressL = lAddress.getText().trim();

String departmentL = lDepartment.getText().trim();

String workingTypeL = lWorkingType.getText().trim();

String employmentStatusL = lEmploymentStatus.getText().trim();

if (lIdAvailable >=0 && workingHoursL >=0) {

//checking if id is available or not

boolean isAdded = true;

for(int i=0; i<listOfTeacher.size(); i++)

{ //iterating throughout the array

if ( (lIdAvailable == listOfTeacher.get(i).getTeacher\_id()) && (listOfTeacher.get(i) instanceof Lecturer) ) {

isAdded = false;

}

}

//adding only when lecturer id is available

if (isAdded == true) {

//adding to arrayList, upcasting Lecturer values to Teacher type

Lecturer lecturer = new Lecturer(lIdAvailable, nameL, addressL, workingTypeL, workingHoursL, employmentStatusL, departmentL, experienceL);

listOfTeacher.add(lecturer);

//showing msg that addition of lecturer was successful

JOptionPane.showMessageDialog(lecturerFrame, "Successfully Added!");

//clearing all fields after adding successfully

lId.setText("");

lName.setText("");

lAddress.setText("");

lWorkingType.setText("");

lDepartment.setText("");

lExperience.setText("");

lWorkingHours.setText("");

lEmploymentStatus.setText("");

}

else {

JOptionPane.showMessageDialog(lecturerFrame, "Id is already in use!! Try another lecturer id.", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

else {

JOptionPane.showMessageDialog(salaryFrame, "Please enter positive values.");

}

}

}catch(NumberFormatException e) {

JOptionPane.showMessageDialog(lecturerFrame, "Number Format Exception. Enter number only in number-required fields.");

}

}

});

//clearing all fields, action listener

lClearButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent b)

{

if(!lId.getText().equals("") || !lName.getText().equals("") || !lAddress.getText().equals("") || !lWorkingType.getText().equals("") || !lDepartment.getText().equals("") || !lExperience.getText().equals("") || !lWorkingHours.getText().equals("") || !lEmploymentStatus.getText().equals("") ){

int confirmation = JOptionPane.showConfirmDialog(lecturerFrame, "Do you really want to clear all fields?", "Clear", JOptionPane.WARNING\_MESSAGE);

if (confirmation == JOptionPane.YES\_OPTION) {

lId.setText("");

lName.setText("");

lAddress.setText("");

lWorkingType.setText("");

lDepartment.setText("");

lExperience.setText("");

lWorkingHours.setText("");

lEmploymentStatus.setText("");

}

}

}

});

//opens new Frame to display entered tutors

lDisplayButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent c)

{

//creating display frame for when display button is clicked

lDisplayFrame = new JFrame("Display Lecturer Details");

lDisplayFrame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE); //different from exit on close; runs in the background

lDisplayFrame.setSize(700, 500);

lDisplayFrame.setLocationRelativeTo(buttonsFrame); //lays the frame relative to buttonsFrame

displayLecturerList();

}

});

//goes back to main frame(buttonsFrame)

lBackButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent d)

{

//clears all of its fields after clicking the back button to avoid issues with exception handeling

lId.setText("");

lName.setText("");

lAddress.setText("");

lWorkingType.setText("");

lDepartment.setText("");

lExperience.setText("");

lWorkingHours.setText("");

lEmploymentStatus.setText("");

lecturerFrame.setVisible(false);

buttonsFrame.setVisible(true); //makes main frame visible

}

});

//changing font size for all labels except lHeader

Font labelFont = lIdLabel.getFont();

labelFont = labelFont.deriveFont(Font.BOLD, 16);

lIdLabel.setFont(labelFont);

lNameLabel.setFont(labelFont);

lAddressLabel.setFont(labelFont);

lWorkingTypeLabel.setFont(labelFont);

lDepartmentLabel.setFont(labelFont);

lExperienceLabel.setFont(labelFont);

lWorkingHoursLabel.setFont(labelFont);

lEmploymentStatusLabel.setFont(labelFont);

//increasing font size for lHeader

Font headerFont = lHeader.getFont();

headerFont = headerFont.deriveFont(lHeader.getFont().getSize() \* 1.8f); //increasing by 80%

lHeader.setFont(headerFont);

//increasing font size for buttons

Font buttonFont = lClearButton.getFont();

buttonFont = buttonFont.deriveFont(lClearButton.getFont().getSize() \* 1.2f); //increasing button's font size by 20%

lAddButton.setFont(buttonFont);

lClearButton.setFont(buttonFont);

lDisplayButton.setFont(buttonFont);

lBackButton.setFont(buttonFont);

lHeader.setForeground(Color.WHITE);

lIdLabel.setForeground(Color.WHITE);

lNameLabel.setForeground(Color.WHITE);

lAddressLabel.setForeground(Color.WHITE);

lWorkingTypeLabel.setForeground(Color.WHITE); //setting white text color to labels

lDepartmentLabel.setForeground(Color.WHITE);

lExperienceLabel.setForeground(Color.WHITE);

lWorkingHoursLabel.setForeground(Color.WHITE);

lEmploymentStatusLabel.setForeground(Color.WHITE);

lDisplayButton.setBackground(new Color(211, 211, 211));

lClearButton.setBackground(new Color(211, 211, 211)); //setting grey bgcolor to buttons

lAddButton.setBackground(new Color(211, 211, 211));

lBackButton.setBackground(new Color(211, 211, 211));

//adding to panel

lPanel.add(lHeader);

lPanel.add(lIdLabel);

lPanel.add(lNameLabel);

lPanel.add(lAddressLabel);

lPanel.add(lWorkingTypeLabel);

lPanel.add(lDepartmentLabel);

lPanel.add(lExperienceLabel);

lPanel.add(lWorkingHoursLabel);

lPanel.add(lEmploymentStatusLabel);

lPanel.add(lId);

lPanel.add(lName);

lPanel.add(lAddress);

lPanel.add(lWorkingType);

lPanel.add(lDepartment);

lPanel.add(lExperience);

lPanel.add(lWorkingHours);

lPanel.add(lEmploymentStatus);

lPanel.add(lDisplayButton);

lPanel.add(lClearButton);

lPanel.add(lAddButton);

lPanel.add(lBackButton);

//adding to frame

lecturerFrame.add(lPanel);

}

public void displayLecturerList() {

lDisplayPanel = new JPanel(null);

lDisplayPanel.setBackground(new Color(47, 79, 79));//creating labels and text-fields, setting bounds

//creating labels

dlHeader = new JLabel("List of Available Lecturers");

dlHeader.setBounds(212, 80, 300, 35);

//creating textarea

JTextArea displayArea = new JTextArea(10, 10);

String message = "";

for(int i=0; i<listOfTeacher.size(); i++)

{ //iterating throughout the arraylist

if ( (listOfTeacher.get(i) instanceof Lecturer) ) {

message += listOfTeacher.get(i).display() ;

}

}

//checking if user has inputted any data and displaying according to result

if (message.equals("")) {

displayArea.setText("There is currently no data here.\n Please input and try again.");

displayArea.setFont(displayArea.getFont().deriveFont(16.0f)); //changing font of displayed text

}

else {

displayArea.setText(message);

displayArea.setFont(displayArea.getFont().deriveFont(16.0f)); //changing font of displayed text

}

//making it so that when display button is clicked, text area view starts from beginning

displayArea.setCaretPosition(0);

//creating scrollPane

JScrollPane scrollingPanel = new JScrollPane(displayArea);

scrollingPanel.setBounds(170, 180, 350, 220);

//creating buttons

dlBackButton = new JButton("<-Back-");

dlBackButton.setBounds(20, 20, 90, 25);

//increasing font size for dHeader

Font headerFont = dlHeader.getFont();

headerFont = headerFont.deriveFont(dlHeader.getFont().getSize() \* 1.8f); //increase by 80%

dlHeader.setFont(headerFont);

//giving white color to labels

dlHeader.setForeground(Color.WHITE);

//changing button bg color

dlBackButton.setBackground(new Color(211, 211, 211));

//action listener

//goes back to Lecturer's frame

dlBackButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent b)

{

lDisplayFrame.setVisible(false);

lecturerFrame.setVisible(true); //when back button is clicked, it goes back to previous frame

}

});

//adding to panel

lDisplayPanel.add(dlHeader);

lDisplayPanel.add(dlBackButton);

lDisplayPanel.add(scrollingPanel);

//adding to frame

lDisplayFrame.add(lDisplayPanel);

lecturerFrame.setVisible(false);

lDisplayFrame.setVisible(true); //makes only the displaying frame visible

}

public void displayTutorList() {

tDisplayPanel = new JPanel(null);

tDisplayPanel.setBackground(new Color(47, 79, 79));//creating labels and text-fields, setting bounds

//creating labels

dtHeader = new JLabel("List of Available Tutors");

dtHeader.setBounds(225, 80, 300, 35);

//creating textarea

JTextArea displayArea = new JTextArea(10, 10);

String message = "";

for(int i=0; i<listOfTeacher.size(); i++)

{ //iterating throughout the arraylist

if ( (listOfTeacher.get(i) instanceof Tutor) ) {

message += listOfTeacher.get(i).display() ;

}

}

//checking if user has inputted any data and displaying according to result

if (message.equals("")) {

displayArea.setText("There is currently no data here. \nPlease input and try again.");

displayArea.setFont(displayArea.getFont().deriveFont(16.0f)); //changing font of displayed text

}

else {

displayArea.setText(message);

displayArea.setFont(displayArea.getFont().deriveFont(16.0f)); //changing font of displayed text

}

//making it so that when display button is clicked, text area view starts from beginning

displayArea.setCaretPosition(0);

//creating scrollPane

JScrollPane scrollingPanel = new JScrollPane(displayArea);

scrollingPanel.setBounds(170, 180, 350, 220);

//creating buttons

dtBackButton = new JButton("<-Back-");

dtBackButton.setBounds(20, 20, 90, 25);

//increasing font size for dHeader

Font headerFont = dtHeader.getFont();

headerFont = headerFont.deriveFont(dtHeader.getFont().getSize() \* 1.8f); //increase by 80%

dtHeader.setFont(headerFont);

//giving white color to labels

dtHeader.setForeground(Color.WHITE);

//changing button bg color

dtBackButton.setBackground(new Color(211, 211, 211));

//action listener

//goes back to Tutor's frame

dtBackButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent b)

{

tDisplayFrame.setVisible(false);

tutorFrame.setVisible(true); //when back button is clicked, it goes back to previous frame

}

});

//adding to panel

tDisplayPanel.add(dtHeader);

tDisplayPanel.add(dtBackButton);

tDisplayPanel.add(scrollingPanel);

//adding to frame

tDisplayFrame.add(tDisplayPanel);

tutorFrame.setVisible(false);

tDisplayFrame.setVisible(true); //makes only the displaying frame visible

}

public void tutorsFrame() {

tPanel = new JPanel(null);

tPanel.setBackground(new Color(47, 79, 79)); //giving bgcolor

//creating labels and setting bounds

tHeader = new JLabel("Enter Tutor Details ");

tHeader.setBounds(300, 30, 260, 60);

tIdLabel = new JLabel("Id: ");

tIdLabel.setBounds(30, 150, 50, 20);

tNameLabel = new JLabel("Name: ");

tNameLabel.setBounds(30, 200, 70, 20);

tAddressLabel = new JLabel("Address: ");

tAddressLabel.setBounds(30, 250, 100, 20);

tWorkingTypeLabel = new JLabel("Working Type: ");

tWorkingTypeLabel.setBounds(30, 300, 150, 20);

tSpecializationLabel = new JLabel("Specialization: ");

tSpecializationLabel.setBounds(30, 350, 150, 20);

tPerformanceLabel = new JLabel("Performance Index: ");

tPerformanceLabel.setBounds(420, 150, 180, 20);

tSalaryLabel = new JLabel("Salary: ");

tSalaryLabel.setBounds(420, 200, 100, 20);

tWorkingHoursLabel = new JLabel("Working Hours: ");

tWorkingHoursLabel.setBounds(420, 250, 190, 20);

tEmploymentStatusLabel = new JLabel("Employment Status: ");

tEmploymentStatusLabel.setBounds(420, 300, 170, 20);

tQualificationLabel = new JLabel("Qualification: ");

tQualificationLabel.setBounds(420, 350, 170, 20);

//creating text-fields

tId = new JTextField();

tId.setBounds(150, 150, 170, 25);

tName = new JTextField();

tName.setBounds(150, 200, 170, 25);

tAddress = new JTextField();

tAddress.setBounds(150, 250, 170, 25);

tWorkingType = new JTextField();

tWorkingType.setBounds(150, 300, 170, 25);

tSpecialization = new JTextField();

tSpecialization.setBounds(150, 350, 170, 25);

tPerformance = new JTextField();

tPerformance.setBounds(593, 150, 180, 25);

tSalary = new JTextField();

tSalary.setBounds(593, 200, 180, 25);

tWorkingHours = new JTextField();

tWorkingHours.setBounds(593, 250, 180, 25);

tEmploymentStatus = new JTextField();

tEmploymentStatus.setBounds(593, 300, 180, 25);

tQualification = new JTextField();

tQualification.setBounds(593, 350, 180, 25);

//creating buttons

tAddButton = new JButton("Add to list");

tAddButton.setBounds(160, 440, 130, 33);

tDisplayButton = new JButton("Display");

tDisplayButton.setBounds(360, 440, 120, 33);

tClearButton = new JButton("Clear");

tClearButton.setBounds(550, 440, 120, 33);

tBackButton = new JButton("<-Back-");

tBackButton.setBounds(20, 20, 90, 25);

//number format listener try-catch block

tAddButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent a)

{

try {

if( (tId.getText().trim().equals("")) || (tName.getText().trim().equals("")) || (tAddress.getText().trim().equals("")) || (tWorkingType.getText().trim().equals("")) || (tSalary.getText().trim().equals("")) || (tQualification.getText().trim().equals("")) || (tWorkingHours.getText().trim().equals("")) || (tEmploymentStatus.getText().trim().equals("")) || (tSpecialization.getText().trim().equals("")) || (tPerformance.getText().trim().equals(""))) {

JOptionPane.showMessageDialog(tutorFrame, "Empty Fields Found! Please fill all text areas. ", "Error", JOptionPane.ERROR\_MESSAGE);

}

else {//trim function removes un-needed whitespaces in front of and behind inputs

int performanceT = Integer.parseInt(tPerformance.getText().trim());

int salaryT = Integer.parseInt(tSalary.getText().trim());

int workingHoursT = Integer.parseInt(tWorkingHours.getText().trim());

String nameT = tName.getText().trim();

String addressT = tAddress.getText().trim();

String specializationT = tSpecialization.getText().trim();

String workingTypeT = tWorkingType.getText().trim();

String employmentStatusT = tEmploymentStatus.getText().trim();

String qualificationT = tQualification.getText().trim();

int tIdAvailable = Integer.parseInt(tId.getText());

if (tIdAvailable >=0 && salaryT >=0 && performanceT >=0 && workingHoursT >=0) {

boolean isAdded = true;

for(int i=0; i<listOfTeacher.size(); i++) {

if ( (tIdAvailable == listOfTeacher.get(i).getTeacher\_id()) && (listOfTeacher.get(i) instanceof Tutor) ) {

isAdded = false;

}

}

//adding only when tutor id is available

if (isAdded == true) {

//adding to arrayList, upcasting Tutor values to Teacher type

Tutor tutor = new Tutor(tIdAvailable, nameT, addressT, workingTypeT, employmentStatusT, workingHoursT, salaryT, specializationT, qualificationT, performanceT);

listOfTeacher.add(tutor);

//showing msg that addition of tutor was successful

JOptionPane.showMessageDialog(tutorFrame, "Successfully Added!");

//clearing all fields after adding successfully

tId.setText("");

tName.setText("");

tAddress.setText("");

tWorkingType.setText("");

tSalary.setText("");

tSpecialization.setText("");

tWorkingHours.setText("");

tEmploymentStatus.setText("");

tPerformance.setText("");

tQualification.setText("");

}

else {

JOptionPane.showMessageDialog(tutorFrame, "Id is already in use!! Try another tutor id.", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

else {

JOptionPane.showMessageDialog(salaryFrame, "Please enter positive values.");

}

}

}catch(NumberFormatException e) {

JOptionPane.showMessageDialog(tutorFrame, "Number Format Exception. Enter number only in number-required fields.");

}

}

});

//clearing all fields, action listener

tClearButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent b)

{

if((!tId.getText().equals("")) || (!tName.getText().equals("")) || (!tAddress.getText().equals("")) || (!tWorkingType.getText().equals("")) || (!tSalary.getText().equals("")) || (!tQualification.getText().equals("")) || (!tWorkingHours.getText().equals("")) || (!tEmploymentStatus.getText().equals("")) || (!tSpecialization.getText().equals("")) || (!tPerformance.getText().equals(""))) {

int confirmation = JOptionPane.showConfirmDialog(tutorFrame, "Do you really want to clear all fields?", "Clear", JOptionPane.WARNING\_MESSAGE);

if (confirmation == JOptionPane.YES\_OPTION) {

tId.setText("");

tName.setText("");

tAddress.setText("");

tWorkingType.setText("");

tSpecialization.setText("");

tSalary.setText("");

tWorkingHours.setText("");

tEmploymentStatus.setText("");

tQualification.setText("");

tPerformance.setText("");

}

}

}

});

//opens new Frame to display entered tutors

tDisplayButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent c)

{

//creating display frame for when display button is clicked

tDisplayFrame = new JFrame("Display Tutor Details");

tDisplayFrame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE); //different from exit on close; runs in the background

tDisplayFrame.setSize(700, 500);

tDisplayFrame.setLocationRelativeTo(buttonsFrame); //lays the frame relative to buttonsFrame

displayTutorList();

}

});

//goes back to main frame(buttonsFrame)

tBackButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent d)

{

//clears all of its fields after clicking the back button to avoid issues with exception handeling

tId.setText("");

tName.setText("");

tAddress.setText("");

tWorkingType.setText("");

tSpecialization.setText("");

tSalary.setText("");

tWorkingHours.setText("");

tEmploymentStatus.setText("");

tQualification.setText("");

tPerformance.setText("");

tutorFrame.setVisible(false);

buttonsFrame.setVisible(true);

}

});

//changing font size for all labels except tHeader

Font labelFont = tIdLabel.getFont();

labelFont = labelFont.deriveFont(Font.BOLD, 16);

tIdLabel.setFont(labelFont);

tNameLabel.setFont(labelFont);

tAddressLabel.setFont(labelFont);

tWorkingTypeLabel.setFont(labelFont);

tSpecializationLabel.setFont(labelFont);

tPerformanceLabel.setFont(labelFont);

tWorkingHoursLabel.setFont(labelFont);

tEmploymentStatusLabel.setFont(labelFont);

tSalaryLabel.setFont(labelFont);

tQualificationLabel.setFont(labelFont);

//increasing font size for tHeader

Font headerFont = tHeader.getFont();

headerFont = headerFont.deriveFont(tHeader.getFont().getSize() \* 1.8f); //increasing by 80%

tHeader.setFont(headerFont);

//increasing font size for buttons

Font buttonFont = tClearButton.getFont();

buttonFont = buttonFont.deriveFont(tClearButton.getFont().getSize() \* 1.2f);

tAddButton.setFont(buttonFont);

tClearButton.setFont(buttonFont);

tDisplayButton.setFont(buttonFont);

tBackButton.setFont(buttonFont);

tHeader.setForeground(Color.WHITE);

tIdLabel.setForeground(Color.WHITE);

tNameLabel.setForeground(Color.WHITE);

tAddressLabel.setForeground(Color.WHITE);

tWorkingTypeLabel.setForeground(Color.WHITE); //setting white text color to labels

tSpecializationLabel.setForeground(Color.WHITE);

tSalaryLabel.setForeground(Color.WHITE);

tWorkingHoursLabel.setForeground(Color.WHITE);

tEmploymentStatusLabel.setForeground(Color.WHITE);

tPerformanceLabel.setForeground(Color.WHITE);

tQualificationLabel.setForeground(Color.WHITE);

tDisplayButton.setBackground(new Color(211, 211, 211));

tClearButton.setBackground(new Color(211, 211, 211)); //setting grey bgcolor to buttons

tAddButton.setBackground(new Color(211, 211, 211));

tBackButton.setBackground(new Color(211, 211, 211));

//adding to panel

tPanel.add(tHeader);

tPanel.add(tIdLabel);

tPanel.add(tNameLabel);

tPanel.add(tAddressLabel);

tPanel.add(tWorkingTypeLabel);

tPanel.add(tSalaryLabel);

tPanel.add(tSpecializationLabel);

tPanel.add(tWorkingHoursLabel);

tPanel.add(tEmploymentStatusLabel);

tPanel.add(tPerformanceLabel);

tPanel.add(tQualificationLabel);

tPanel.add(tId);

tPanel.add(tName);

tPanel.add(tAddress);

tPanel.add(tWorkingType);

tPanel.add(tSpecialization);

tPanel.add(tSalary);

tPanel.add(tWorkingHours);

tPanel.add(tEmploymentStatus);

tPanel.add(tPerformance);

tPanel.add(tQualification);

tPanel.add(tDisplayButton);

tPanel.add(tClearButton);

tPanel.add(tAddButton);

tPanel.add(tBackButton);

//adding to frame

tutorFrame.add(tPanel);

}

public void gradesAssignment() {

gPanel = new JPanel(null);

gPanel.setBackground(new Color(47, 79, 79)); //giving bgcolor

//creating labels and setting bounds

gHeader = new JLabel("Enter Details ");

gHeader.setBounds(250, 80, 260, 60);

gIdLabel = new JLabel("Id: ");

gIdLabel.setBounds(160, 170, 50, 20);

gDepartmentLabel = new JLabel("Department: ");

gDepartmentLabel.setBounds(160, 220, 120, 20);

gGradedScoreLabel = new JLabel("Graded Score: ");

gGradedScoreLabel.setBounds(160, 270, 130, 20);

gExperienceLabel = new JLabel("Experience: ");

gExperienceLabel.setBounds(160, 320, 120, 20);

//creating text-fields and setting bounds

gId = new JTextField();

gId.setBounds(300, 170, 170, 25);

gDepartment = new JTextField();

gDepartment.setBounds(300, 220, 170, 25);

gGradedScore = new JTextField();

gGradedScore.setBounds(300, 270, 170, 25);

gExperience = new JTextField();

gExperience.setBounds(300, 320, 170, 25);

//creating buttons

gGradeButton = new JButton("Grade");

gGradeButton.setBounds(190, 390, 120, 33);

gClearButton = new JButton("Clear");

gClearButton.setBounds(340, 390, 120, 33);

gBackButton = new JButton("<-Back-");

gBackButton.setBounds(20, 20, 90, 25);

//adding action listeners to buttons

//grades according to score

gGradeButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent a)

{

try

{

if( (gId.getText().trim().equals("")) || (gDepartment.getText().trim().equals("")) || (gGradedScore.getText().trim().equals("")) || (gExperience.getText().trim().equals("")) ) {

JOptionPane.showMessageDialog(gradeFrame, "Empty Field Found! Please fill required area. ", "Error", JOptionPane.ERROR\_MESSAGE);

}

else {

//grading code

String departmentCheck = gDepartment.getText().trim(); //trim function removes un-needed whitespaces in front of and behind inputs

int experienceCheck = Integer.parseInt(gExperience.getText().trim());

int gradedScoreCheck = Integer.parseInt(gGradedScore.getText().trim());

int idCheck = Integer.parseInt(gId.getText().trim());

String resultingGrade = "";

if (idCheck >=0 && experienceCheck >=0 && gradedScoreCheck >= 0) {

//iterating through the arraylist and checking if each is instance of lecturer

for(int i=0; i<listOfTeacher.size(); i++) {

if( (listOfTeacher.get(i).getTeacher\_id() == idCheck) && (listOfTeacher.get(i) instanceof Lecturer) ) {

resultingGrade = ((Lecturer)listOfTeacher.get(i)).gradeAssignment(gradedScoreCheck, departmentCheck, experienceCheck );

}

}

//checking if id exists

if(resultingGrade.equals("")) {

JOptionPane.showMessageDialog(gradeFrame, "Lecturer id doesn't exist. Input correct id.");

}

else {

//showing result and clearing only after grading successfully

if (resultingGrade.equals("A") || resultingGrade.equals("B") || resultingGrade.equals("C") || resultingGrade.equals("D") || resultingGrade.equals("E")) {

String gradedMessage = "You have scored " + resultingGrade + "!!";

JOptionPane.showMessageDialog(gradeFrame, gradedMessage);

gId.setText("");

gDepartment.setText("");

gExperience.setText("");

gGradedScore.setText("");

}

else {

//if resultingGrade string doesn't have store grade then showing appropriate message, i.e, department or experience mismatch

JOptionPane.showMessageDialog(gradeFrame, resultingGrade);

}

}

}

else {

JOptionPane.showMessageDialog(salaryFrame, "Please enter positive values.");

}

}

}

catch(NumberFormatException e) {

JOptionPane.showMessageDialog(gradeFrame, "Number Format Exception. Enter number only in number-required fields.");

}

}

});

//clears all fields

gClearButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent b)

{

if((!gId.getText().equals("")) || (!gDepartment.getText().equals("")) || (!gGradedScore.getText().equals("")) || (!gExperience.getText().equals(""))) {

int confirmation = JOptionPane.showConfirmDialog(gradeFrame, "Do you really want to clear all fields?", "Clear", JOptionPane.WARNING\_MESSAGE);

if (confirmation == JOptionPane.YES\_OPTION) {

gId.setText("");

gDepartment.setText("");

gGradedScore.setText("");

gExperience.setText("");

}

}

}

});

//goes back to main frame(buttonsFrame)

gBackButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent c)

{

//clears all of its fields after clicking the back button to avoid issues with exception handeling

gId.setText("");

gDepartment.setText("");

gGradedScore.setText("");

gExperience.setText("");

gradeFrame.setVisible(false);

buttonsFrame.setVisible(true);

}

});

//changing font size for all labels except gHeader

Font labelFont = gIdLabel.getFont();

labelFont = labelFont.deriveFont(Font.BOLD, 16);

gIdLabel.setFont(labelFont);

gDepartmentLabel.setFont(labelFont);

gGradedScoreLabel.setFont(labelFont);

gExperienceLabel.setFont(labelFont);

//increasing font size for gHeader

Font headerFont = gHeader.getFont();

headerFont = headerFont.deriveFont(gHeader.getFont().getSize() \* 1.8f); //increasing by 80%

gHeader.setFont(headerFont);

//increasing font size for buttons

Font buttonFont = gClearButton.getFont();

buttonFont = buttonFont.deriveFont(gClearButton.getFont().getSize() \* 1.2f); //increasing by 20%

gGradeButton.setFont(buttonFont);

gClearButton.setFont(buttonFont);

gBackButton.setFont(buttonFont);

gHeader.setForeground(Color.WHITE);

gIdLabel.setForeground(Color.WHITE);

gDepartmentLabel.setForeground(Color.WHITE);

gGradedScoreLabel.setForeground(Color.WHITE);

gExperienceLabel.setForeground(Color.WHITE); //setting white text color to labels

gGradeButton.setBackground(new Color(211, 211, 211));

gClearButton.setBackground(new Color(211, 211, 211)); //setting grey bgcolor to buttons

gBackButton.setBackground(new Color(211, 211, 211));

//adding to panel

gPanel.add(gHeader);

gPanel.add(gIdLabel);

gPanel.add(gDepartmentLabel);

gPanel.add(gGradedScoreLabel);

gPanel.add(gExperienceLabel);

gPanel.add(gId);

gPanel.add(gDepartment);

gPanel.add(gGradedScore);

gPanel.add(gExperience);

gPanel.add(gGradeButton);

gPanel.add(gClearButton);

gPanel.add(gBackButton);

//adding to frame

gradeFrame.add(gPanel);

}

public void setsSalary() {

sPanel = new JPanel(null);

sPanel.setBackground(new Color(47, 79, 79)); //giving bgcolor

//creating labels and setting bounds

sHeader = new JLabel("Enter Details ");

sHeader.setBounds(225, 80, 260, 60);

sIdLabel = new JLabel("Id: ");

sIdLabel.setBounds(160, 170, 50, 20);

sPerformanceLabel = new JLabel("Performance: ");

sPerformanceLabel.setBounds(160, 220, 140, 20);

sSalaryLabel = new JLabel("Salary: ");

sSalaryLabel.setBounds(160, 270, 130, 20);

//creating text-fields and setting bounds

sId = new JTextField();

sId.setBounds(280, 170, 170, 25);

sPerformance = new JTextField();

sPerformance.setBounds(280, 220, 170, 25);

sSalary = new JTextField();

sSalary.setBounds(280, 270, 170, 25);

//creating buttons

sSetSalaryButton = new JButton("Set");

sSetSalaryButton.setBounds(230, 345, 60, 33);

sClearButton = new JButton("Clear");

sClearButton.setBounds(320, 345, 80, 33);

sBackButton = new JButton("<-Back-");

sBackButton.setBounds(20, 20, 90, 25);

//adding action listeners to buttons

//grades according to score

sSetSalaryButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent a)

{

try {

if( sId.getText().trim().equals("") || sPerformance.getText().trim().equals("") || sSalary.getText().trim().equals("") ) {

JOptionPane.showMessageDialog(salaryFrame, "Empty Field Found! Please fill required area. ", "Error", JOptionPane.ERROR\_MESSAGE);

}

else {

//setting salary code

int idCheck = Integer.parseInt(sId.getText().trim());

double salaryCheck = Integer.parseInt(sSalary.getText().trim());

int performanceCheck = Integer.parseInt(sPerformance.getText().trim());

String updatedSalary = "";

if (idCheck >=0 && salaryCheck >=0 && performanceCheck >=0) {

//iterating through the arraylist and checking if each is instance of lecturer

for(int i=0; i<listOfTeacher.size(); i++) {

if( (listOfTeacher.get(i).getTeacher\_id() == idCheck) && (listOfTeacher.get(i) instanceof Tutor) ) {

updatedSalary = ((Tutor)listOfTeacher.get(i)).setSalary(salaryCheck, performanceCheck);

}

}

if(updatedSalary.equals("")) {

JOptionPane.showMessageDialog(salaryFrame, "Tutor id doesn't exist. Input correct id.");

}

else {

if (performanceCheck > 5) {

if (performanceCheck > 10) {

JOptionPane.showMessageDialog(salaryFrame, updatedSalary);

}

else {

String salaryMessage = "The tutor's updated salary is " + updatedSalary;

JOptionPane.showMessageDialog(salaryFrame, salaryMessage);

//clearing field after setting salary successfully

sId.setText("");

sPerformance.setText("");

sSalary.setText("");

}

}

else if (performanceCheck < 5){

JOptionPane.showMessageDialog(salaryFrame, updatedSalary);

}

else {

JOptionPane.showMessageDialog(salaryFrame, "Unexpected Error");

}

}

}

else {

JOptionPane.showMessageDialog(salaryFrame, "Please enter positive values.");

}

}

}catch(NumberFormatException e) {

JOptionPane.showMessageDialog(salaryFrame, "Number Format Exception. Enter number only in number-required fields.");

}

}

});

//clears all fields

sClearButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent b)

{

if( !sId.getText().equals("") || !sPerformance.getText().equals("") || !sSalary.getText().equals("") ) {

int confirmation = JOptionPane.showConfirmDialog(salaryFrame, "Do you really want to clear all fields?", "Clear", JOptionPane.WARNING\_MESSAGE);

if (confirmation == JOptionPane.YES\_OPTION) {

sId.setText("");

sPerformance.setText("");

sSalary.setText("");

}

}

}

});

//goes back to main frame(buttonsFrame)

sBackButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent c)

{

//clears all of its fields after clicking the back button to avoid issues with exception handeling

sId.setText("");

sPerformance.setText("");

sSalary.setText("");

salaryFrame.setVisible(false);

buttonsFrame.setVisible(true);

}

});

//changing font size for all labels except sHeader

Font labelFont = sIdLabel.getFont();

labelFont = labelFont.deriveFont(Font.BOLD, 16);

sIdLabel.setFont(labelFont);

sPerformanceLabel.setFont(labelFont);

sSalaryLabel.setFont(labelFont);

//increasing font size for sHeader

Font headerFont = sHeader.getFont();

headerFont = headerFont.deriveFont(sHeader.getFont().getSize() \* 1.8f); //increasing by 80%

sHeader.setFont(headerFont);

//increasing font size for buttons

Font buttonFont = sClearButton.getFont();

buttonFont = buttonFont.deriveFont(sClearButton.getFont().getSize() \* 1.2f); //increasing by 20%

sSetSalaryButton.setFont(buttonFont);

sClearButton.setFont(buttonFont);

sBackButton.setFont(buttonFont);

sHeader.setForeground(Color.WHITE);

sIdLabel.setForeground(Color.WHITE);

sPerformanceLabel.setForeground(Color.WHITE);

sSalaryLabel.setForeground(Color.WHITE); //setting white text color to labels

sSetSalaryButton.setBackground(new Color(211, 211, 211));

sClearButton.setBackground(new Color(211, 211, 211)); //setting grey bgcolor to buttons

sBackButton.setBackground(new Color(211, 211, 211));

//adding to panel

sPanel.add(sHeader);

sPanel.add(sIdLabel);

sPanel.add(sPerformanceLabel);

sPanel.add(sSalaryLabel);

sPanel.add(sId);

sPanel.add(sPerformance);

sPanel.add(sSalary);

sPanel.add(sSetSalaryButton);

sPanel.add(sClearButton);

sPanel.add(sBackButton);

//adding to frame

salaryFrame.add(sPanel);

}

public void removesTutor() {

rPanel = new JPanel(null);

rPanel.setBackground(new Color(47, 79, 79));

//creating labels and text-fields, setting bounds

rHeader = new JLabel("Enter Id");

rHeader.setBounds(250, 90, 460, 60);

rIdLabel = new JLabel("Id: ");

rIdLabel.setBounds(190, 200, 50, 20);

rId = new JTextField();

rId.setBounds(230, 200, 150, 25);

//creating buttons

rRemoveButton = new JButton("Remove");

rRemoveButton.setBounds(250, 260, 100, 33);

rBackButton = new JButton("<-Back-");

rBackButton.setBounds(20, 20, 90, 25);

//changing font size for id label

Font labelFont = rIdLabel.getFont();

labelFont = labelFont.deriveFont(Font.BOLD, 16);

rIdLabel.setFont(labelFont);

//increasing font size for rHeader

Font headerFont = rHeader.getFont();

headerFont = headerFont.deriveFont(rHeader.getFont().getSize() \* 1.8f); //increase by 80%

rHeader.setFont(headerFont);

//increasing font size for buttons

Font buttonFont = rRemoveButton.getFont();

buttonFont = buttonFont.deriveFont(rRemoveButton.getFont().getSize() \* 1.2f);

rRemoveButton.setFont(buttonFont);

rBackButton.setFont(buttonFont);

//giving white color to labels

rHeader.setForeground(Color.WHITE);

rIdLabel.setForeground(Color.WHITE);

//changing button bg color

rRemoveButton.setBackground(new Color(211, 211, 211));

rBackButton.setBackground(new Color(211, 211, 211));

//action listener

//number format exception

rRemoveButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent a)

{

try {

if( rId.getText().trim().equals("") ) {

JOptionPane.showMessageDialog(removeFrame, "Empty Field Found! Please fill required area. ", "Error", JOptionPane.ERROR\_MESSAGE);

}

else {

int idR = Integer.parseInt(rId.getText().trim());

if (idR >=0) {

//removing code

boolean isRemoved = false;

for(int i=0; i<listOfTeacher.size(); i++) {

if ( idR == listOfTeacher.get(i).getTeacher\_id() && listOfTeacher.get(i) instanceof Tutor ) {

listOfTeacher.remove(i);

isRemoved = true;

}

}

if(isRemoved == true) {

//showing message to user that the tutor has been removed successfully

JOptionPane.showMessageDialog(removeFrame, "The tutor has been removed successfully");

//clearing field after removing successfully

rId.setText("");

}

else {

//showing message to user that the tutor has not been removed; failed to remove

JOptionPane.showMessageDialog(removeFrame, "Enter valid Tutor Id.", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

else {

JOptionPane.showMessageDialog(salaryFrame, "Please enter positive values.");

}

}

}catch(NumberFormatException e) {

JOptionPane.showMessageDialog(removeFrame, "Number Format Exception. Enter number only in number-required fields.");

}

}

});

//goes back to main frame(buttonsFrame)

rBackButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent b)

{

//clears its fields after clicking the back button to avoid issues with exception handeling

rId.setText("");

removeFrame.setVisible(false);

buttonsFrame.setVisible(true);

}

});

//adding to panel

rPanel.add(rHeader);

rPanel.add(rIdLabel);

rPanel.add(rId);

rPanel.add(rRemoveButton);

rPanel.add(rBackButton);

//adding to frame

removeFrame.add(rPanel);

}

public static void main(String[] args)

{

SwingUtilities.invokeLater(new Runnable() {

public void run() {

new TeacherGUI();

}

});

}

}