



**Module Code & Module Title CS4001 Programming**

**Assessment weightage & Type 30% Individual Coursework**

**Year and Semester 2024 Spring**

**Student’s Name: Janak Devkota**

**Group: C6**

**College ID: NP04CP4S230019**

**London Met ID: 23048806 Assessment Due Date: March 29, 2024**

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

Table of Contents

[1. Introduction 1](#_Toc164728077)

[2. Class Diagram 2](#_Toc164728078)

[I. Class Diagram Of Teacher Class 2](#_Toc164728079)

[II. Class Diagram Of Lecturer Class 2](#_Toc164728080)

[III. Class Diagram Of Tutor Class 2](#_Toc164728081)

[IV. Class Diagram Of TeacherGui Class 2](#_Toc164728082)

[3. Pseudocode 2](#_Toc164728083)

[4. Method Description 2](#_Toc164728084)

[5. Testing 2](#_Toc164728085)

[I. Test 1 2](#_Toc164728086)

[II. Test 2 2](#_Toc164728087)

[- Add Lecturer 2](#_Toc164728088)

[- Add Tutor 2](#_Toc164728089)

[- Grade Assignment 2](#_Toc164728090)

[- Set Salary 2](#_Toc164728091)

[- Remove Tutor 2](#_Toc164728092)

[- Display 2](#_Toc164728093)

[III. Test 3 2](#_Toc164728094)

[6. Error Detection and Correction 2](#_Toc164728095)

[I. Syntax Error 2](#_Toc164728096)

[II. Runtime Error 2](#_Toc164728097)

[III. Logical Error 2](#_Toc164728098)

[7. Conclusion 2](#_Toc164728099)

[8. Bibliography 2](#_Toc164728100)

[9. Appendix 2](#_Toc164728101)

**Table of Figure**

**List of Tables**

# Introduction

This coursework focuses on making user-friendly interfaces for programs, called graphical user interfaces (GUIs). We've already worked on some classes for this project, like teacher, lecturer, and tutor. Instead of just typing commands into a black screen, GUIs use pictures or symbols to help us communicate with computers. In Java, there are two main ways to make these GUIs: AWT and Swing. We picked Swing because it's newer and has more tools to make fancier interfaces, while still being easy on the computer's resources.

Our main class for this program is TeacherGui. We're storing information in ArrayLists. Last time, we used BlueJ to write our code, but now we're using IntelliJ IDEA.

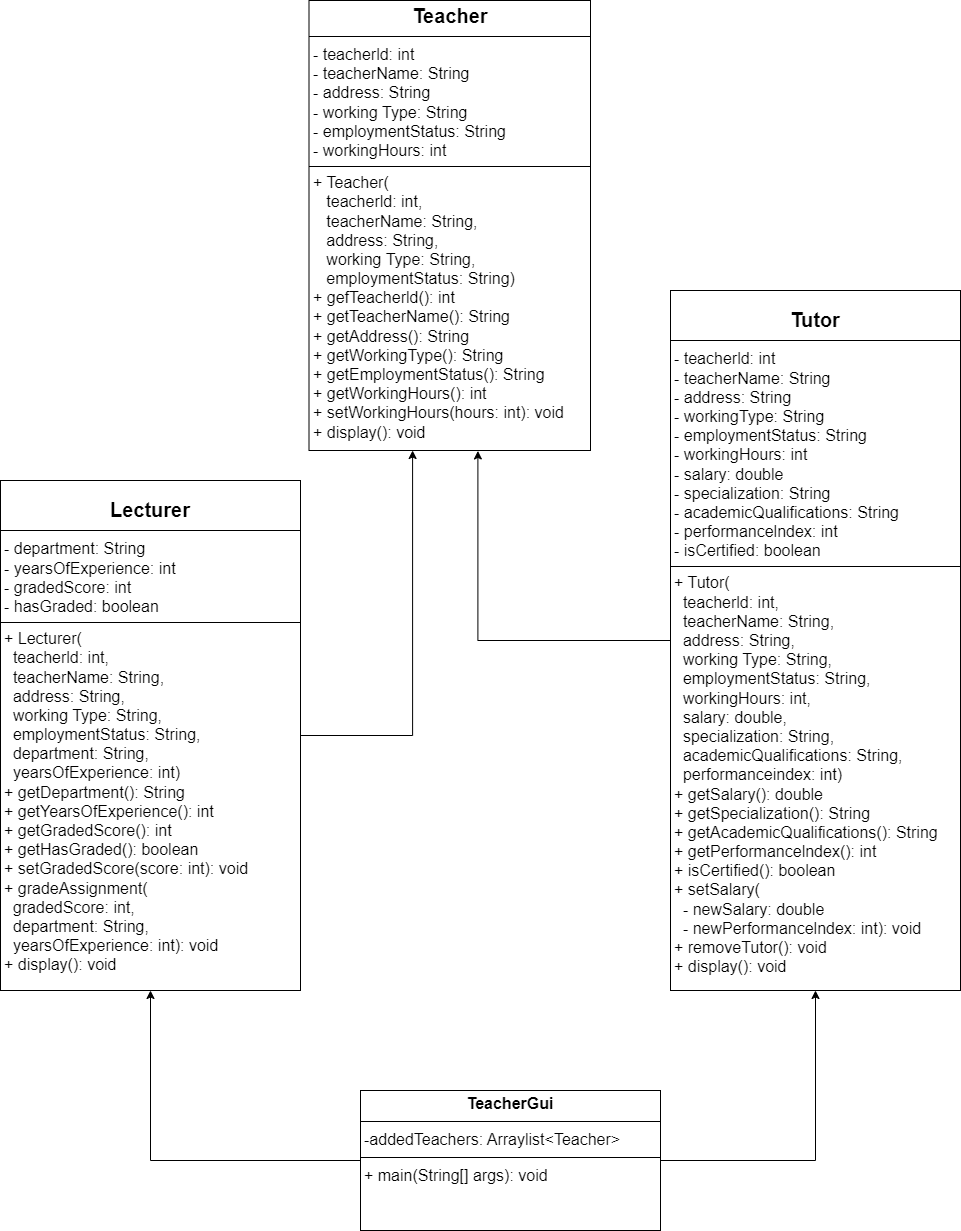
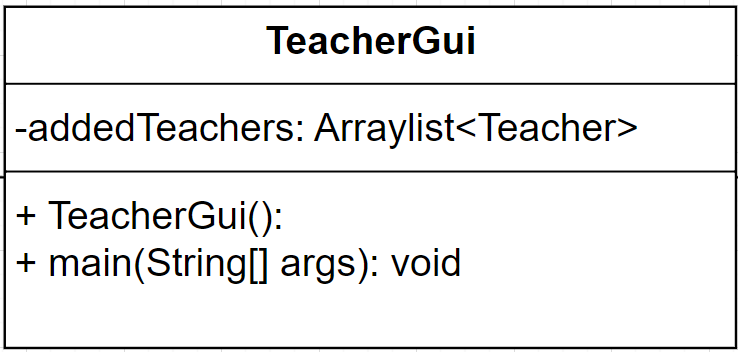
This course covers both writing code and explaining what it does. By practicing, we'll get good at making GUIs, using Java Swing, and working with ArrayLists.

The program itself has a constructor and a main method. Inside the constructor, we set up the frame and sections. Each section has its own job, like creating labels, text fields, and buttons. When we run the program, the main method calls the constructor to show the GUI.

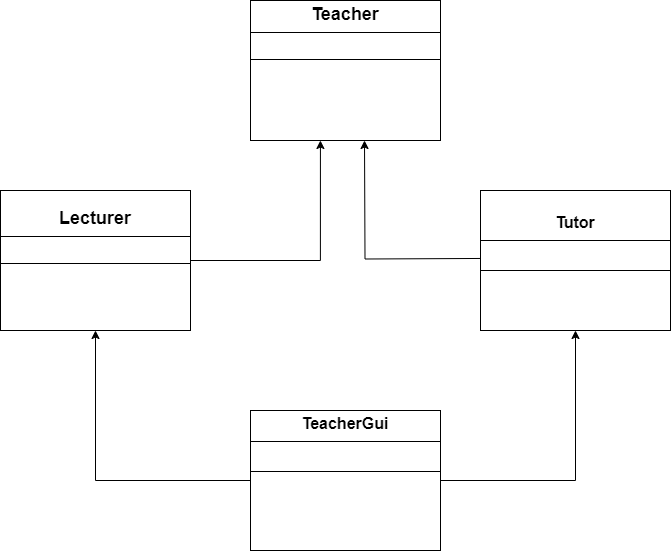
So, in short, we're learning to make programs easier to use with GUIs. We're using Java Swing because it's powerful but not too heavy on the computer. Our TeacherGui class is the heart of the program, and we're storing data in ArrayLists. We're learning both coding and explaining what we've coded, focusing on GUIs, Swing, and ArrayLists. The constructor sets up the interface, and the main method starts everything up.

As I prepare for this coursework, it's crucial to understand not just how to code but also why specific tools and techniques are chosen. By grasping GUI development principles like user experience and the efficiency of Swing over AWT, I become a more versatile programmer. Mastering data structures like ArrayLists enhances my ability to manage information effectively. This coursework not only improves my technical skills but also deepens my understanding of software design principles, laying a solid foundation for future programming challenges.

# Class Diagram

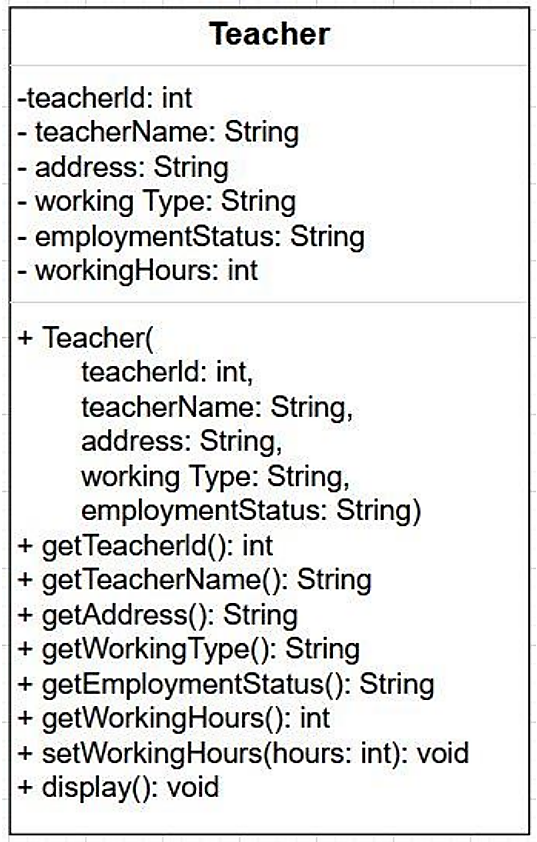


* Empty Class Diagram:



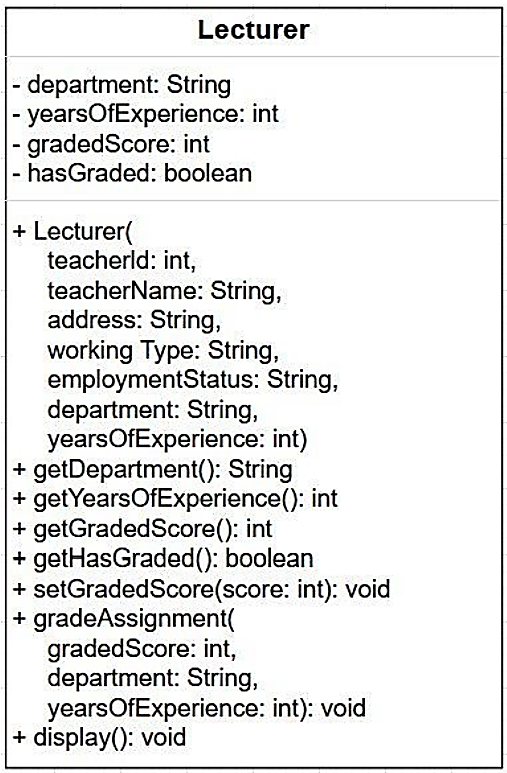
This program features a hierarchical class structure comprising three classes: Teacher, Lecturer, and Tutor. In this design, Teacher acts as the superclass, while Lecturer and Tutor serve as its subclasses. This class diagram encapsulates the hierarchical inheritance structure, showcasing the relationships and attributes of Teacher, Lecturer, and Tutor classes. The strategic use of getter and setter methods underlines the program's commitment to encapsulation, promoting a well-organized and efficient approach to handling private instance variables.

## Class Diagram Of Teacher Class



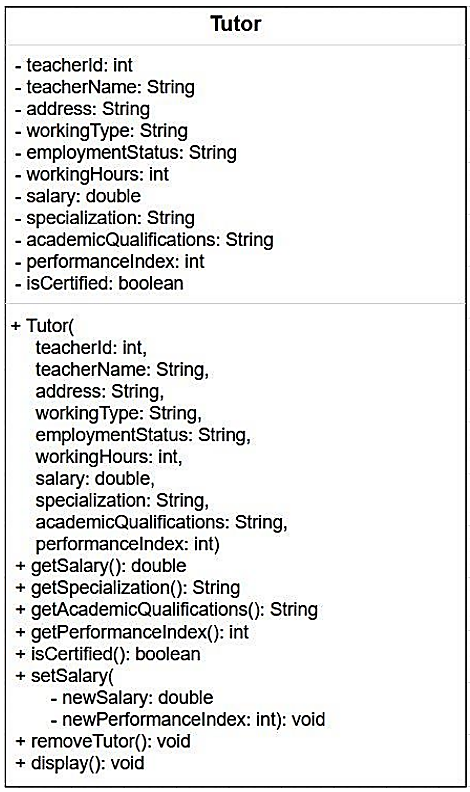
The class diagram for the "Teacher" class defines attributes like teacherId, teacherName, address, workingType, employmentStatus, and workingHours. It includes methods (getters and setters) for accessing and modifying these attributes, facilitating object-oriented management and manipulation of teacher data in a program.

## Class Diagram Of Lecturer Class



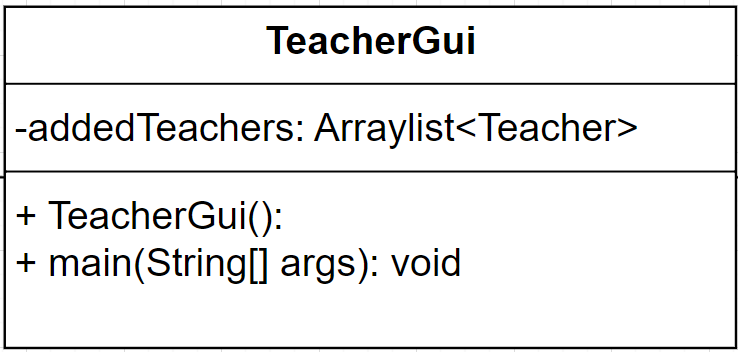
The extended "Lecturer" class introduces additional attributes: department, yearsOfExperience, gradedScore, and hasGraded. It includes methods like getDepartment, getYearsOfExperience, getGradedScore, getHasGraded, setGradedScore, gradeAssignment, and display, tailored for lecturers' unique responsibilities. This extension enhances the object-oriented framework, addressing specific lecturer requirements and functionalities.

## Class Diagram Of Tutor Class



The "Tutor" class expands upon "Teacher" and "Lecturer," featuring attributes such as salary, specialization, academic qualifications, performance index, and isCertified. Accompanying methods include getSalary, getSpecialization, getAcademicQualifications, getPerformanceIndex, isCertified, setSalary, removeTutor, and display. This comprehensive template ensures effective representation and interaction with tutor instances in software systems. Feel free to inquire about specific implementation details or further clarification.

## IV. Class Diagram Of TeacherGui Class



# 3. Pseudocode

# 4. Method Description

**I. .Constructor TeacherGui()**

**II. Main method()**

# 5. Testing

## I. Test 1

## II. Test 2

### - Add Lecturer

### - Add Tutor

### - Grade Assignment

### - Set Salary

### - Remove Tutor

### - Display

## III. Test 3

# 6. Error Detection and Correction

## I. Syntax Error

## II. Runtime Error

## III. Logical Error

# 7. Conclusion

# 8. Bibliography

# 9. Appendix

}

//Main method that call constructor public static void main(String[] args) {

new StudentGui();

}

}