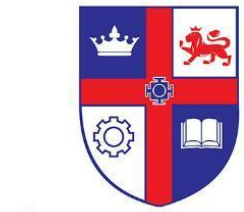
LONDON METROPOLITAN UNIVERSITY



islington college
(इस्लिङ्टन कलेज)

CS4001NI Programming

30% Individual Coursework

2023-24 Autumn

Student Name: Aman Adhikari

London Met ID: 23047306

College ID: NP01NT4A230003

Group: N1

Assignment Due Date: Friday, January 26, 2024

Assignment Submission Date: Friday, January 26, 2024

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

Contents

Table of Figure:

Table of Table:

1. Introduction

Java programming is done with object orientation and classes. The goal of the language's design is to minimize implementation dependencies. Developers won't have to worry about writing code for every platform by using this language. Sometimes this language is associated with "write once and run everywhere," also known as "WORA." It also suggests that byte code (.class files), which are generated during the compilation of Java code, may be executed on any platform that supports Java without the need for further compilation. The Java language was first developed in 1995. Its main application is in the development of desktop, mobile, and web apps. The Java language is widely recognized for its security, ease of use, and robustness. Its goal is to have as few implementation dependencies as possible. (Anon., 2023)

1.1 About the coursework

Mr. Ujwal Subedi is our lecturer teacher, and this is the first assignment we have to do for top programming. The workshop and tutor are overseen by Mrs. Astha Sharma. They do admirably in this module. They know a great deal about this module. This course was given at the end of the semester. This assignment makes up 30% of the module's grade. The purpose of this project is to use Java's object-oriented idea to construct a real-world issue scenario. To do this, we will need to create a class that represents a teacher and two subclasses that represent a lecturer and a tutor, respectively. Some of the information is entirely new to us because this is the first course we were given on the subject at hand. We had many buddy discussions in order to do this task on time.

1.2 Tools used

In this coursework, I have used different tools to complete the whole thing. And a brief description of all the software and websites is given below:

1.2.1 Blue j

Figure 1:Blue J

The development of BlueJ was aimed at facilitating object-oriented programming instruction, and its interface allows users to view classes and coded objects visually. The tool is designed to make programming languages such as Java more accessible by utilizing visual representations of Java code. The underlying concept behind object-oriented programming is to apply a multi-view approach to display both visible objects and the source code underlying them. The BlueJ interface operates on a similar concept, allowing users to add classes from external files and create new projects with ease. While this type of application is not unique, other interface scales also display source code using icons. (Anon., 2023)

1.2.2 MS Word

Figure 2 Ms Word

Microsoft Word is a robust word processing tool with an array of capabilities for writing reports, papers, resumes, and letters at the professional level. Word has features that include photo support, text and font formatting, HTML compatibility, advanced page layout, spell and grammar checking, and more than basic text editors. As part of the Microsoft Office Suite, Word is one of the powerful applications created to provide businesses of all sizes access to a wide range of tools for almost any kind of task. (Anon., 2022)

1.2.3 Draw.io

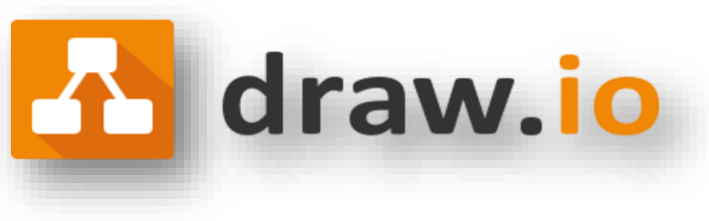


Figure 3 Draw . IO

A program called Draw.io, formerly known as Diagrams.net, is designed specifically for making charts and diagrams. It offers layout possibilities that are both automatically generated and user-customizable. With the tool's extensive array of forms and graphic elements, users may design intricate and one-of-a-kind diagrams. Draw.io retains an easy-to-use interface that is both aesthetically beautiful and straightforward, even with its complex capabilities. It is a good substitute for well-known applications like yEd or Microsoft Visio, but it also offers special characteristics that make it stand out. Users may also create their own image libraries and import diagrams from other applications, such as Visio and Gliffy, using this software. (Anon., 2023)

2. Class Diagram

A fixed perspective of an application is shown in the class diagram. It shows the types of things that are present in the system and their connections to each other. In addition to having objects of their own, classes can inherit from each other. In addition to creating executable software code, a class diagram is used to illustrate, explain, and record a variety of system features. The name of the class is included in the upper portion. A class represents an object having comparable connections, characteristics, methods, and meanings. These characteristics, which characterize the standard of the class, make up the middle section. Operations and steps are given in the last part. A list with a single line for each method is used to represent the methods.

2.1 Class Diagram of Teacher Class

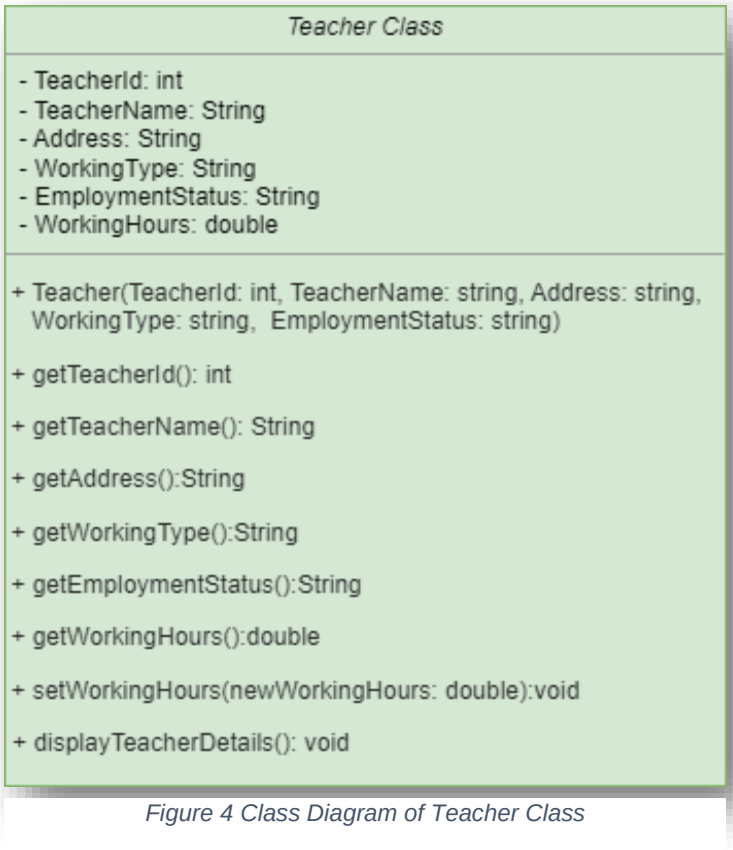


Figure 4 Class Diagram of Teacher Class

Explanation

- The class diagram represent the class with attributes and method.
- Attributes are listed with their respected types.
- The constructor is shown with their parameter.
- Used accessor method (public method) to retrieve attribute values.
- The mutator method is used to set the value of the attributes.
- And last is display method that helps to display the information about Teacher Class.

2.2 Class Diagram of Lecturer Class

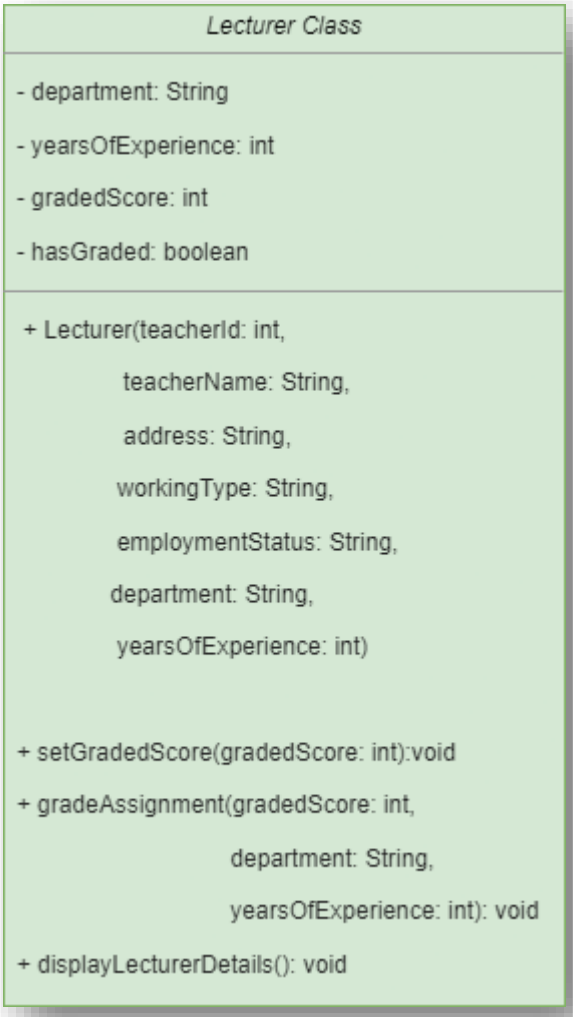


Figure 5 Class Daigram Of Lecturer Class

Explanation

- The Lecture class is sub class of Teacher class.
- The specified attributes of Lecture class has been added.
- The mutator method is assigned for ‘GradeScore’ attribute.
- The method gradeAssignment updates the hasGraded property by assigning grades based on specified criteria for assignments.
- The display method is used for display the Teacher, Lecturer, department, YearOfExperince.

2.3 Class Diagram of Tutor Class

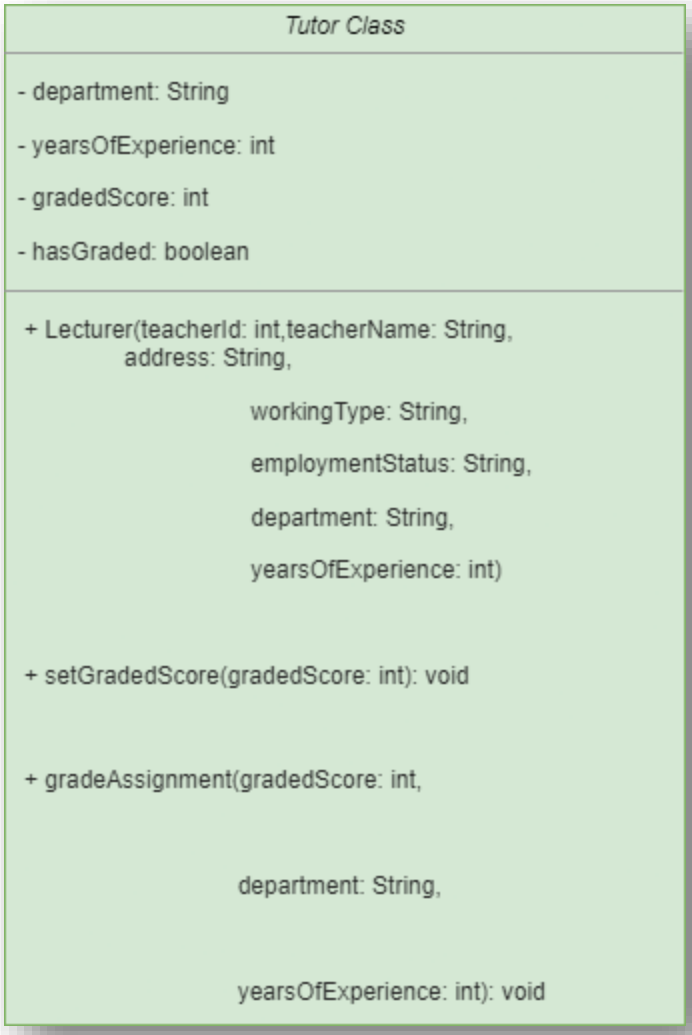


Figure 6 Class Daigram Of Tutor Class

Explanation

- The Tutor class is a subclass of the Teacher class.
- The specified attributes of Tutor class has been added.
- There is a method (setSalaryAndCertify) to certify the tutor and set the salary reliant on the performance index. The isCertified attribute is also updated.
- If the tutor is not certified, another procedure (removeTutor) resets relevant attributes to zero and sets isCertified to false.

- The display method is used for display the Tutor is it was certified otherwise it also display TeacherDetail.

2.4 Combined inheritance Diagram

Figure 7 Combined inheritance Diagram

3. Pseudocode

Pseudocode is a set of simple instructions that can be quickly converted into any programming language for a specific platform. Programmers can ensure that their steps for writing code are comprehensive, efficient, and free from mistakes or omissions. They can establish a framework for documentation, making sure that anyone, including clients and managers, can easily understand the program's logic and processes. This is done by outlining the steps in writing before attempting to write the code. By following this process, programmers can quickly implement presentations, updates, and enhancements. Pseudocode is used to outline the program's logic in a language that is easy to understand. The pseudocode can then be converted into any programming language for any platform, allowing for elegant and efficient code production. Additionally, the original pseudocode can be preserved for documentation and notation purposes, making it easy for future programmers to update and modify the method.

3.1 Pseudocode of Teacher Class

START

CREATE a parent class Teacher

DO

DECLARE an instance variable

DECLARE an instance variable

DECLARE an instance variable

DECLARE an instance variable

DECLARE an instance variable

DECLARE an instance variable

CREATE class constructor which accepts every attribute mentioned above

DO

SET this.TeacherID = TeacherID

SET this.TeacherName = TeacherName

SET this.Address = Address

SET this.WorkingType = WorkingType

SET this.EmploymentStatus = EmploymentStatus

END DO

CREATE getter method getTeacherID() that return int value

DO

RETURN TeacherID

END DO

CREATE getter method getTeacherName() that return string value

DO

RETURN TeacherName

END DO

CREATE getter method getAddress() that return string value

DO

RETURN Address

END DO

CREATE getter method getWorkingType() that return string value

DO

RETURN WorkingType

END DO

CREATE getter method getEmploymentStatus() that return string value

DO

RETURN EmploymentStatus

END DO

CREATE getter method getWorkingHours() that return int value

```
        DO
        RETURN WorkingHours
        END DO

CREATE setter method setWorkingHours(newWorkingHours) that return int value
        DO
        RETURN newWorkingHours
        END DO

CREATE method name display()
        DO
         PRINT "Teacher ID:" + TeacherID
         PRINT "Teacher Name:" + TeacherName
         PRINT "Address:" + Address
         PRINT "Working Type:" + WorkingType
         PRINT "Employment Status:" + EmploymentStatus
        IF (WorkingHours > 0:)
        DO
        PRINT "Working Hours:" + WorkingHours
        END DO
ELSE:
        DO
        PRINT "Working hours not assigned."
```

```
        END IF
        END DO
END DO
```

3.2 Pseudocode of Lecturer Class

```
CLASS Lecturer extends Teacher:
    PRIVATE String Department
    PRIVATE int YearsOfExperience
   PRIVATE int GradeScore
   PRIVATE boolean HasGraded

   METHOD Lecturer(int TeacherID, String TeacherName, String Address, String EmploymentStatus,
        String Department, int YearsOfExperience, String WorkingType):
        SUPER(TeacherID, TeacherName, Address, WorkingType, EmploymentStatus)
        THIS.Department = Department
        THIS YearsOfExperience = YearsOfExperience
        THIS.GradeScore = 0
        THIS.HasGraded = fa
    METHOD getDepartment():
    METHOD this.Department

    METHOD getYearsOfExperience():
        RETURN this.YearsOfExperience
```

```
METHOD getGradeScore():
    RETURN this.GradeScore


METHOD hasGraded():
    RETURN this.HasGraded
METHOD setGradeScore(int GradeScore):
    THIS.GradeScore = GradeScore
METHOD grade(int GradeScore, String Department, int YearsOfExperience):
    IF (this.YearsOfExperience >= 5 and this.Department.equals(Department)):
    IF (GradeScore >= 70):
        THIS.GradeScore = GradeScore
      ELSE IF (GradeScore >= 60):
        THIS.GradeScore = GradeScore
      ELSE IF (GradeScore >= 50):
        THIS.GradeScore = GradeScore
      ELSE IF (GradeScore >= 40):
        THIS.GradeScore = GradeScore
     ELSE
        THIS.GradeScore = 0
      THIS.HasGraded = true
    ELSE:
      PRINT("Lecturer has not graded yet. Please meet the eligibility criteria.")
METHOD display()
```

```
    SUPER.display()
    PRINT("Department: " + Department)
    PRINT("YearsOfExperience: " + YearsOfExperience)
    IF (HasGraded):
        PRINT("HasGraded: " + HasGraded)
    ELSE:
        PRINT("GradeScore: " + GradeScore)
```

3.3 Pseudocode of Tutor Class

CLASS Tutor extends Teacher:

PRIVATE double salary

PRIVATE String specialization

PRIVATE String academicQualification

PRIVATE int performanceIndex

PRIVATE boolean isCertified

METHOD Tutor(int TeacherID, String TeacherName, String Address, String WorkingType, String EmploymentStatus,

int WorkingHours, double salary, String specialization, String academicQualification, int performanceIndex):

SUPER(TeacherID, TeacherName, Address, WorkingType, EmploymentStatus)

SUIPER.setWorkingHours(WorkingHours)

THIS.salary = salary

THIS.specialization = specialization

THIS.academicQualification = academicQualification

THIS.performanceIndex = performanceIndex

THIS.isCertified = false

METHOD getSalary():

RETURN this.salary

METHOD getSpecialization():

RETURN this.specialization

```
METHOD getAcademicQualification():
    RETURN this.academicQualification

METHOD getPerformanceIndex():
    RETURN this.performanceIndex

METHOD getIsCertified():
    RETURN this.isCertified
METHOD setSalary(double newSalary, int newPerformanceIndex):
    IF (newPerformanceIndex > 5 and super.getWorkingHours() > 20):
        DOUBLE appraisal = 0
        IF (newPerformanceIndex >= 5 and newPerformanceIndex <= 7):
            APPRAISAL = 0.5
        ELSE IF (newPerformanceIndex >= 8 and newPerformanceIndex <= 9):
            APPRAISAL = 0.1
        ELSE IF (newPerformanceIndex == 10):
            APPRAISAL = 0.2
        THIS.salary += appraisal * this.salary
        THIS.isCertified = true
    ELSE:
        PRINT("Tutor cannot be certified due to insufficient performanceIndex or
WorkingHours.")
```

```
METHOD removeTutor():
    IF (isCertified):
        THIS.salary = 0
        THIS.specialization = null
        THIS.academicQualification = null
        THIS.performanceIndex = 0
        THIS.isCertified = false
        PRINT("Tutor removed successfully.")
    ELSE:
        PRINT("Cannot remove a certified Tutor.")
METHOD displayDetails():
    SUPER.display()

    IF (isCertified):
        PRINT("Salary: $" + salary)
        PRINT("Specialization: " + specialization)
        PRINT("Academic Qualification: " + academicQualification)
        PRINT("Performance Index: " + performanceIndex)
    IF END
    ELSE:
        PRINT("Tutor is not certified.")
```

4. Description of methods

4.1 Method description of Teacher class

Method	Description
+Teacher()	Constructor that uses supplied data for TeacherID, TeacherName, Address, WorkingType, and EmploymentStatus to initialize the Teacher object. There is no working hour established.
+geTeacherID()	This method helps to return the TeacherId of int type.
+getTeacherName()	This method helps to return the TeacherName of String type.
+getAddress()	This method helps to return the Address of String type.
+getWorkingType()	This method helps to return the WorkingType of String type.
+getEmployemtStatus()	This method helps to return the EmploymentStatus of String type.
+getWorkinHours()	This method helps to return the WorkingHours of String type.
+setWorkingHours()	This helps to set the value of newWorkingHours as int type.
+display()	This method is used to display the detail of teacher if assigned, if not assigned it will show assigned message.

Table 1 Method Description Of Teacher Class

4.2 Method description of Lecturer class

Method	Description
+getLecturer()	Constructor that calls the superclass constructor after initializing the Lecturer object with the supplied data. Sets HasGraded to false and GradeScore to 0.
+getDepartment()	This method helps to return the department of String type.
+getYearsOfExperince()	This method helps to return the YearOfExperince of int type.
+getGradeScore()	This method helps to return the GradeScre of int type.
+gethasGraded()	This method helps to return the hasGraded of boolean type.
+setGradeScore()	This helps to set the value of GradeSocre as int type.
+GradeAssignment	Method to grade assignments based on specified criteria. Updates GradeScore and sets HasGraded to true if the lecturer meets the eligibility conditions. Displays a message if the lecturer has not graded yet or does not meet eligibility criteria.

Table 2 Method description of Lecturer class

4.3 Method description of Tutor class

Method	Description
+Tutor()	
+getSalary()	This method helps to return the salary of int type.
+getSpecialization()	This method helps to return the Specialization of String type.
+getAcademicQualification()	This method helps to return the AcademicQulification of String type.
+getPerformanceIndex()	This method helps to return the PerformanceIndex of int type.
+getIsCertified()	This method helps to return the IsCertified of int type.
+setSalary()	This method for determining salary based on performance index and new salary.
+removeTutor()	Method for using the instructor. If the tutor is not certified, reset important properties to their initial state.
+display()	A method for displaying the Tutor object's details, including those that were taken over from the Teacher class.

Table 3 Method description of Tutor class

5. Testing

5.1 Testing Teacher Class

Test no:	01
Objective	To inspect Teacher Class, appoint Teacher and re-inspect Teacher Class
Activity	The Teacher is called with following Argument: TeacherID = 2004 TeacherName = “Ram Adhikari” Address = “Jadibuti” WorkingType = “Lecturer” WorkinStatus = “Full Time”
Expected Result	The teacher should be added with respective module and ID.
Actual Result	The teacher was appointed for the module and ID.
Conclusion	The test was successful.

Table 4 Testing Teacher Class

Step: 1 Assign the value of Teacher Class

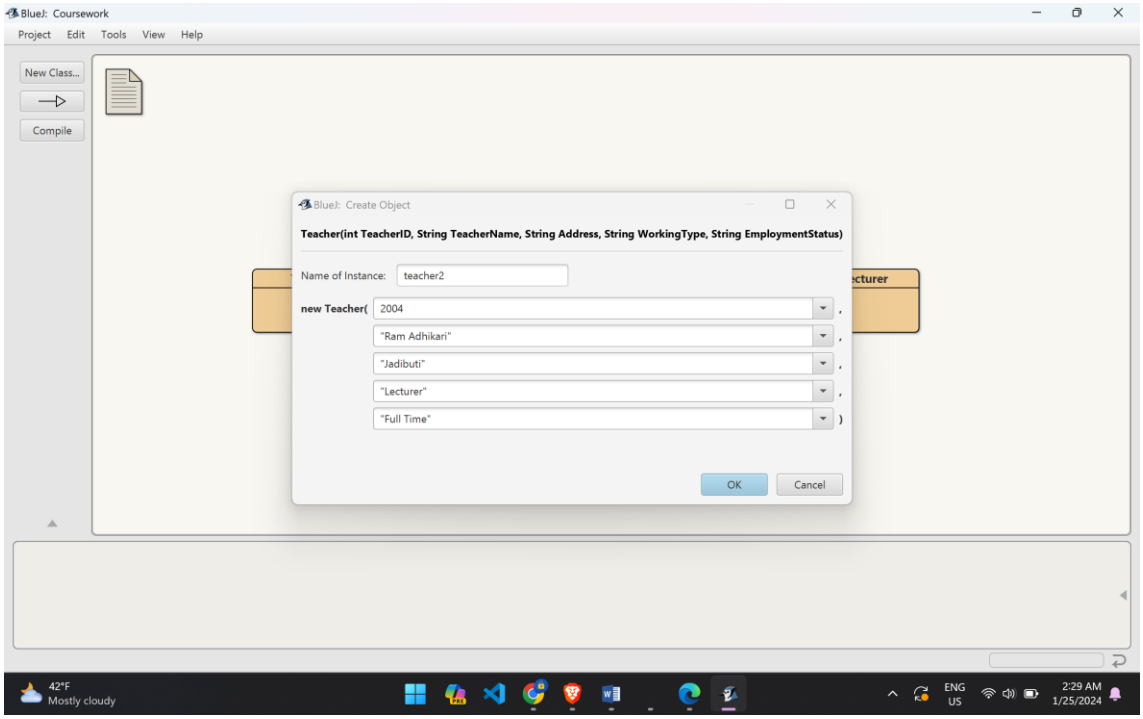


Figure 8 Assign the value of Teacher Class

Step: 2 Inspecting the Teacher Class

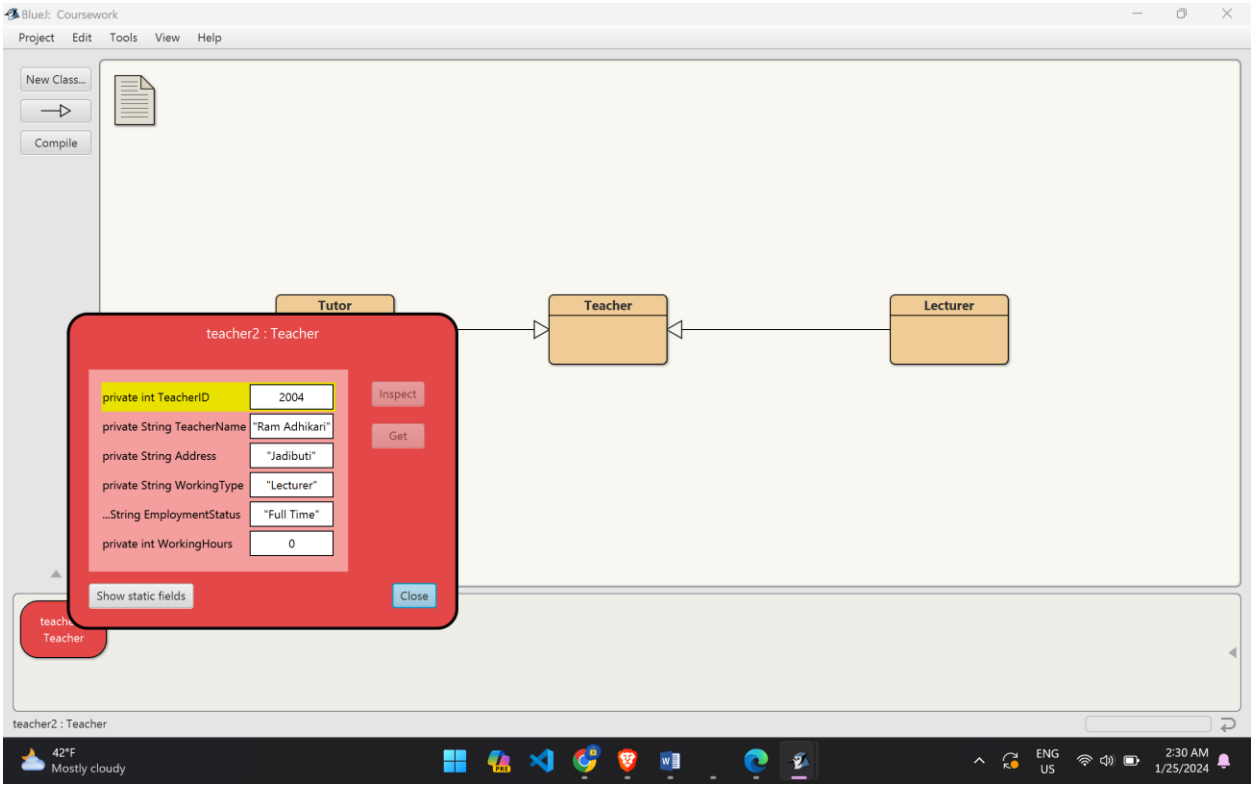


Figure 9 Inspecting the Teacher Class

Step: 3 Display Teacher Class

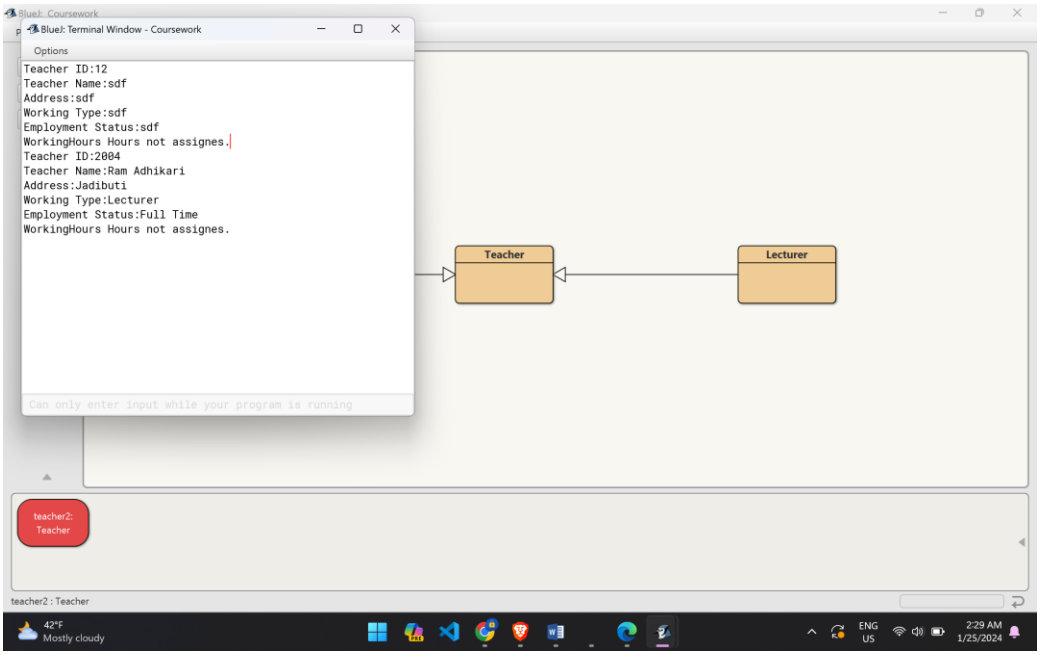


Figure 10 Display Teacher Class

5.2 Testing Lecturer Class

Test No:	2
Objective	To inspect Lecturer Class.
Activity	The Lecturer is called with following argument. Teacher ID:2005 Teacher Name:Gurans Adhikari Address:Raniban Working Type:Lecturer Employment Status:Full Time WorkingHours Hours not assignes. Department: Programming YearsOfExperience: 12 GradeScore: 0
Expected Result	The Lecturer should be added with respective module and ID and years of experience.
Actual Result	The teacher was assigned for the module and ID and years of experience.
Conclusion	Test Was Successful.

Table 5 Testing Lecturer Class

Step: 1 Assign the value of Lecturer Class

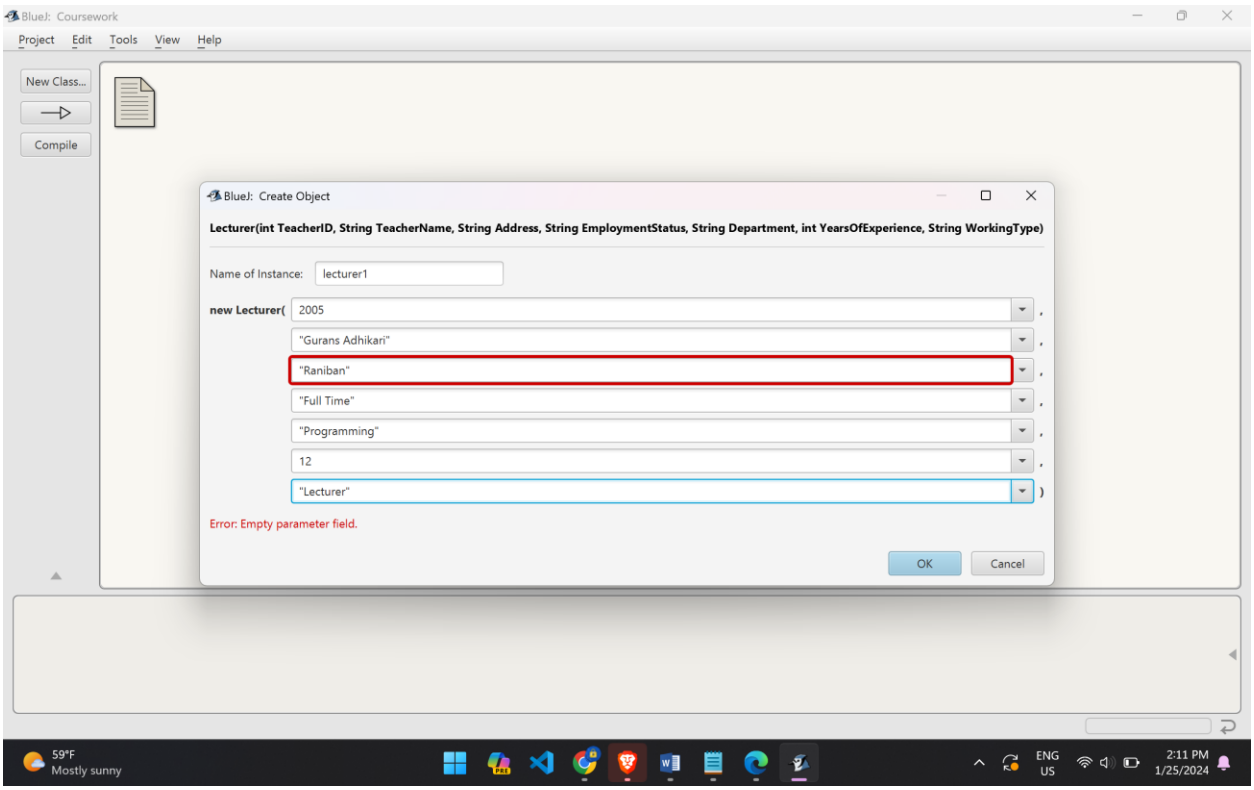


Figure 11 Assign the value of Lecturer Class

Step: 2 Inspecting the value of Lecturer Class

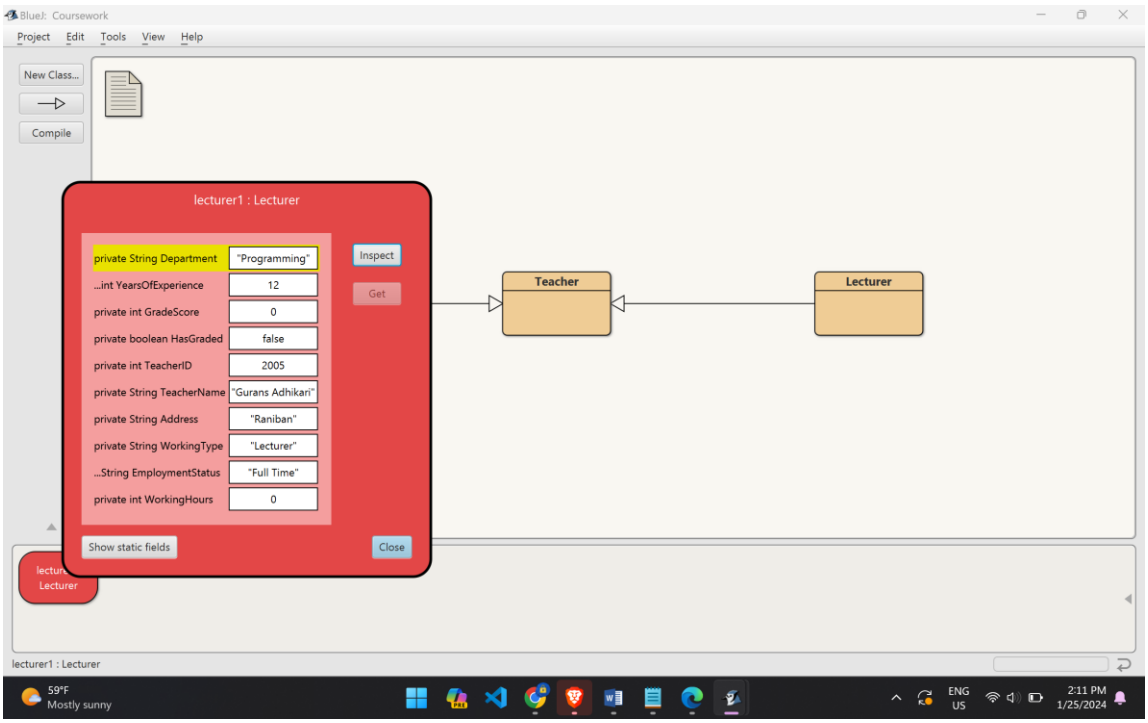


Figure 12 Inspecting the value of Lecturer Class

Step: 3 Display Lecturer Class

```
Teacher ID:2005
Teacher Name:Gurans Adhikari
Address:Raniban
Working Type:Lecturer
Employment Status:Full Time
WorkingHours Hours not assignes.
Department: Programming
YearsOfExperience: 12
GradeScore: 0
```

Figure 13 Display Lecturer Class

5.3 Testing Tutor Class

Test No	3
Objective	To inspect Tutor Class.
Activity	The Tutor is called with following argument. Teacher ID:2006 Teacher Name:Rahul Chaudhary Address:TikaThali Working Type:Tutor Employment Status:Full Time WorkingHours: 7
Expected Result	The Tutor should be added with respective module and ID and working hours.
Actual Result	The tutor was assigned for the module and ID and working hours.
Conclusion	Test Was Successful.

Table 6 Testing Lecturer Class

Step: 1 Assign the value of Tutor Class

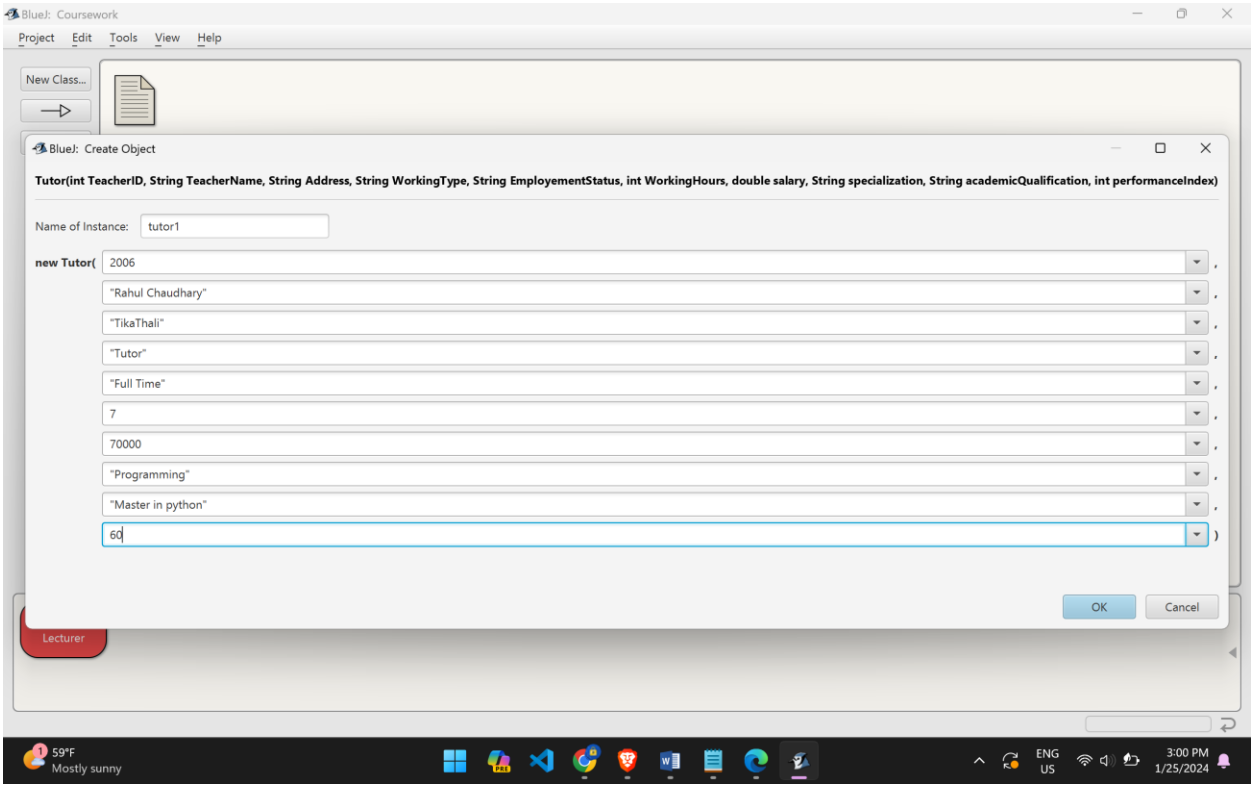


Figure 14 Assign the value of Tutor Class

Step: 2 inspecting the value of Tutor Class

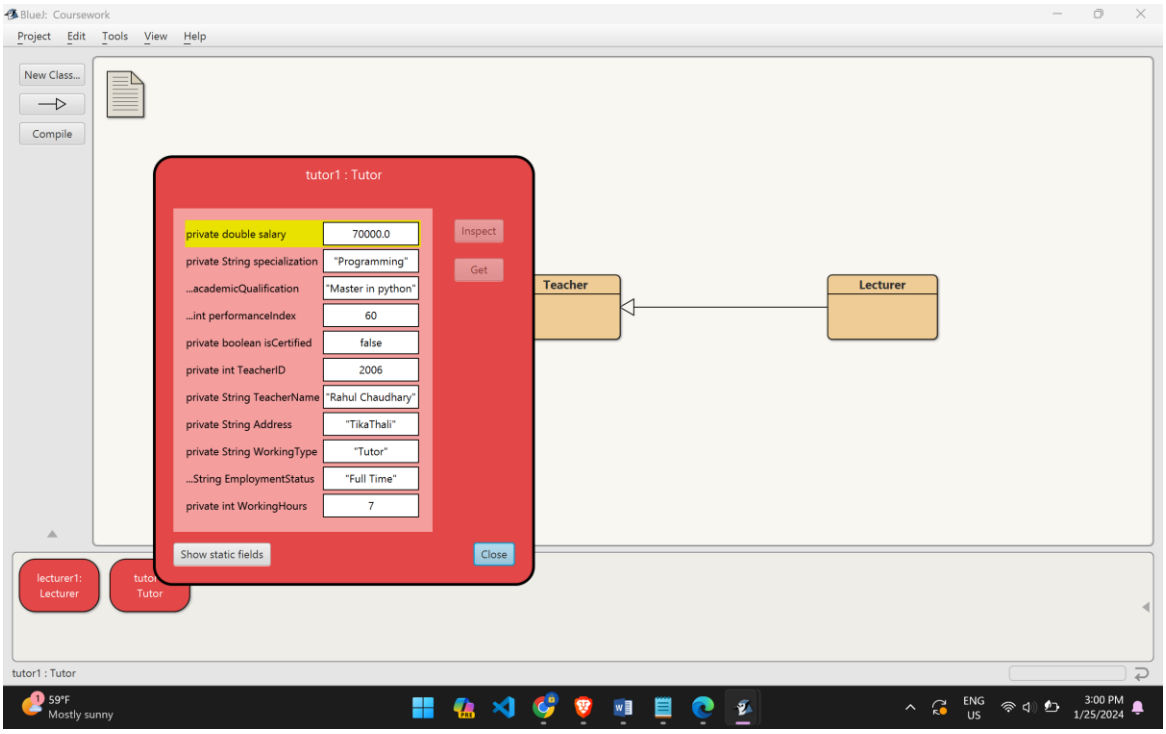


Figure 15 inspecting the value of Tutor Class

Step: 3 Display Tutor Class

```
Teacher ID:2006
Teacher Name:Rahul Chaudhary
Address:TikaThali
Working Type:Tutor
Employment Status:Full Time
WorkingHours: 7
Tutor is not Certified.
```

Figure 16 Display Tutor Class

6. Error detection and correction

When writing computer programs, it is crucial to fix any syntax errors as they can prevent the program from functioning. If there are syntax errors, the software cannot correctly interpret the commands written in the programming language. Syntax in programming can be compared to grammar in language, as it refers to correct syntax and command structure. To address syntax errors, programmers can use debugging methods and manual code-checking best practices within the Interactive Development Environment. (Anon., 2023)

6.1 Syntax Error

```
        return TeacherName;
    }
    public String getAddress(){
        return Address;
    }
    public String getWorkingType(){
        return WorkingType;
    }
    public String getEmploymentStatus(){
        return EmploymentStatus;
    }
    public int getWorkingHours(){
        return WorkingHours;
    }
    public void setWorkingHours(int newWorkingHours){
        this.WorkingHours = newWorkingHours;
    }
    public void display(){
        System.out.println("Teacher ID:" + this.getTeacherID())
        System.out.println("Teacher Name:" + this.getTeacherName());
        System.out.println("Address:" + this.getAddress());
        System.out.println("Working Type:" + this.getWorkingType());
        System.out.println("Employment Status:" + this.getEmploymentStatus());

            if(this.getWorkingHours() == 0){
                System.out.println("WorkingHours Hours not assignes.");
            }else{
                System.out.println("WorkingHours: " + this.getWorkingHours());
            }
    }
}
```

Figure 17 Syntax Error

Correction

```
    }
    public void setWorkingHours(int newWorkingHours){
        this.WorkingHours = newWorkingHours;
    }
    public void display(){
        System.out.println("Teacher ID:" + this.getTeacherID());
        System.out.println("Teacher Name:" + this.getTeacherName());
        System.out.println("Address:" + this.getAddress());
        System.out.println("Working Type:" + this.getWorkingType());
        System.out.println("Employment Status:" + this.getEmploymentStatus());

            if(this.getWorkingHours() == 0){
                System.out.println("WorkingHours Hours not assignes.");
            }else{
                System.out.println("WorkingHours: " + this.getWorkingHours());
            }
    }
}
```

Figure 18 Correction

6.2 Semantics Error

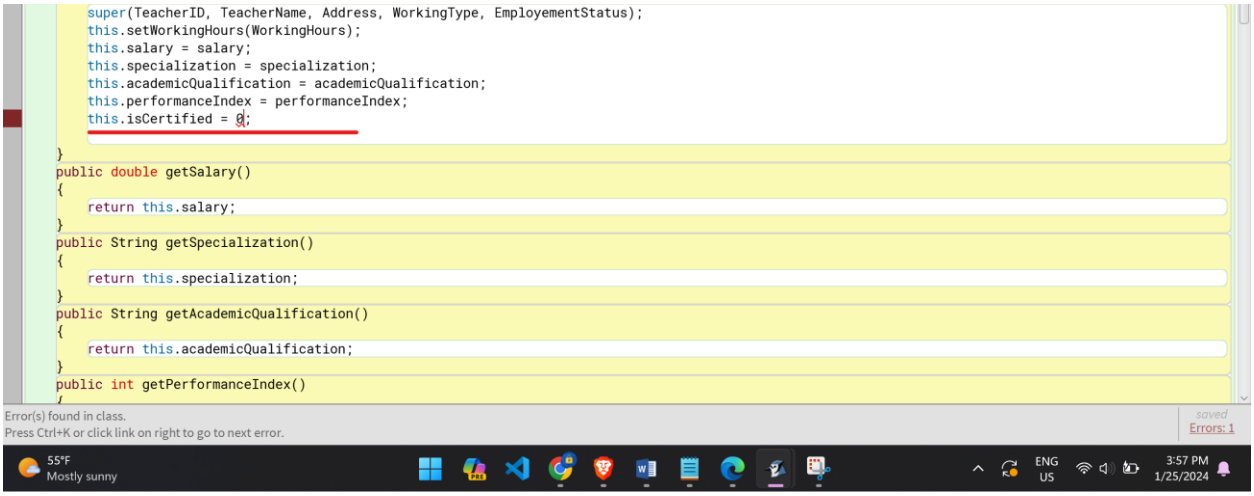


Figure 19 Semantics Error

Correction

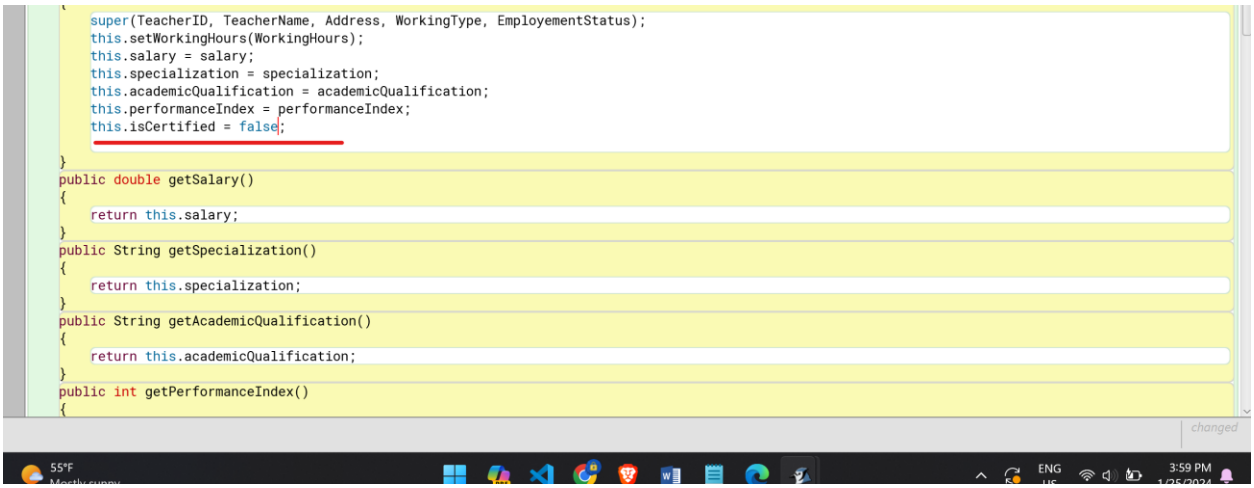
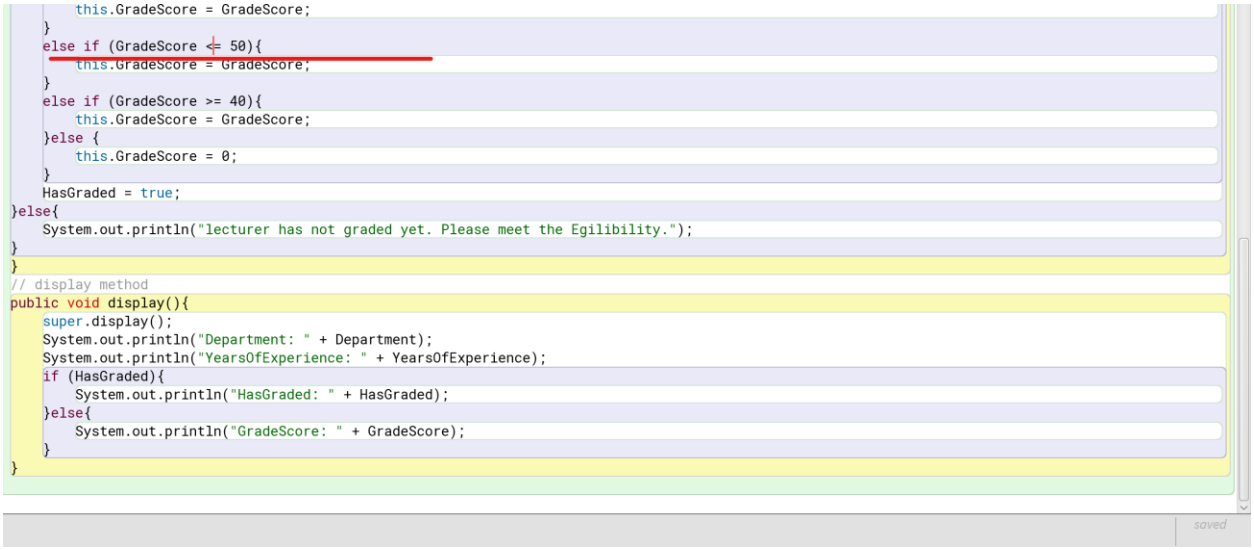


Figure 20 Correction

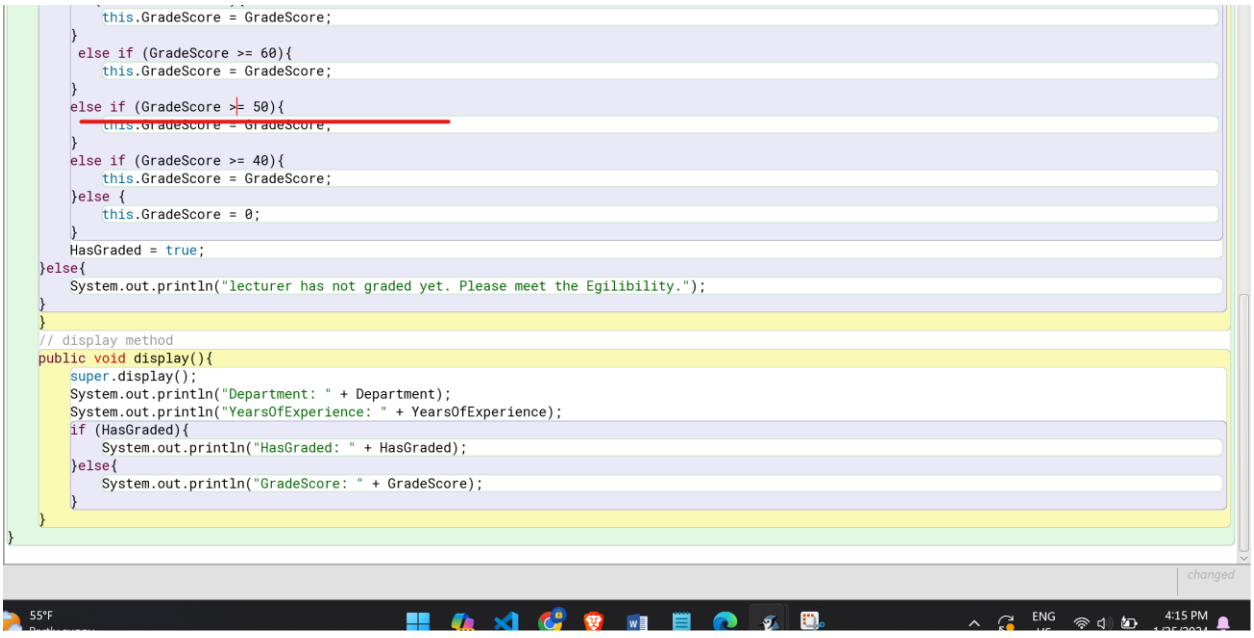
6.3 Logical Error



```
        this.GradeScore = GradeScore;
    }
    else if (GradeScore <= 50){
        this.GradeScore = GradeScore;
    }
    else if (GradeScore >= 40){
        this.GradeScore = GradeScore;
    }else {
        this.GradeScore = 0;
    }
    HasGraded = true;
}else{
    System.out.println("lecturer has not graded yet. Please meet the Egilibility.");
}
}
// display method
public void display(){
    super.display();
    System.out.println("Department: " + Department);
    System.out.println("YearsOfExperience: " + YearsOfExperience);
    if (HasGraded){
        System.out.println("HasGraded: " + HasGraded);
    }else{
        System.out.println("GradeScore: " + GradeScore);
    }
}
```

Figure 21 Logical Error

Correction



```
        this.GradeScore = GradeScore;
    }
     else if (GradeScore >= 60){
        this.GradeScore = GradeScore;
    }
    else if (GradeScore >= 50){
        this.GradeScore = GradeScore;
    }
    else if (GradeScore >= 40){
        this.GradeScore = GradeScore;
    }else {
        this.GradeScore = 0;
    }
    HasGraded = true;
}else{
    System.out.println("lecturer has not graded yet. Please meet the Egilibility.");
}
}
// display method
public void display(){
    super.display();
    System.out.println("Department: " + Department);
    System.out.println("YearsOfExperience: " + YearsOfExperience);
    if (HasGraded){
        System.out.println("HasGraded: " + HasGraded);
    }else{
        System.out.println("GradeScore: " + GradeScore);
    }
}
}
```

Figure 22 Correction

7. Conclusion:

The coursework is divided into two sections: theory and practical. The blueJ app's practical part includes three courses of action: "Tutor Class, Lecturer Class, and Teacher Class." Two subclasses of teacher classes are lecturer and tutor. The theoretical part contains the BlueJ application's class diagram, pseudocode, and method description. To ascertain whether the written codes are legitimate, it involves program testing. After finishing the courses, you will understand what methods, accessor methods, mutator methods, parent and child classes, pseudocode, function Object(), class diagrams, and use are. The class diagram and pseudocode, which had never been done previously, were the most difficult aspects of this course. Overcoming challenges became easier after I learned about the topic, its uses, and how it was created. Utilizing the class diagram and pseudocode reference materials made completing the task much smoother. Consequently, I gained extensive knowledge about Java programming and its various jargon. Working on this project was a great joy.

9. References

Anon., 2022. *ms word.* [Online]
Available at: https://www.geeksforgeeks.org/introduction-to-microsoft-word/
[Accessed 23 1 2024].

Anon., 2023. *bluej.* [Online]
Available at: https://www.techopedia.com/definition/29530/bluej
[Accessed 23 1 2024].

Anon., 2023. *draw.io.* [Online]
Available at: https://readthedocs.web.cern.ch/display/MTA/Introduction+to+draw.io
[Accessed 24 1 2024].

Anon., 2023. *error.* [Online]
Available at: https://www.techwalla.com/articles/how-to-fix-syntax-errors
[Accessed 23 1 2024].

Anon., 2023. *java overview.* [Online]
Available at: https://www.tutorialspoint.com/java/java_overview.htm
[Accessed 21 1 2024].

10. Appendix

10.1 Code of Teacher.java

```
public class Teacher{

    private int TeacherID;

    private String TeacherName;

    private String Address;

    private String WorkingType;

    private String EmploymentStatus;

    private int WorkingHours;


    public Teacher(int TeacherID, String TeacherName,

    String Address, String WorkingType, String EmploymentStatus) {

        this.TeacherID = TeacherID;

        this.TeacherName = TeacherName;

        this.Address = Address;

        this.WorkingType = WorkingType;

        this.EmploymentStatus = EmploymentStatus;

        this.WorkingHours = 0;
```

```
    }

    public int getTeacherID(){
        return TeacherID;
    }
    public String getTeacherName(){
        return TeacherName;
    }
    public String getAddress(){
        return Address;
    }
    public String getWorkingType(){
        return WorkingType;
    }
    public String getEmploymentStatus(){
        return EmploymentStatus;
    }
    public int getWorkingHours(){
        return WorkingHours;
    }
    public void setWorkingHours(int newWorkingHours){
        this.WorkingHours = newWorkingHours;
    }
```

```
    public void display(){
        System.out.println("Teacher ID:" + this.getTeacherID());
        System.out.println("Teacher Name:" + this.getTeacherName());
        System.out.println("Address:" + this.getAddress());
        System.out.println("Working Type:" + this.getWorkingType());
        System.out.println("Employment Status:" + this.getEmploymentStatus());

            if(this.getWorkingHours() == 0){
                System.out.println("WorkingHours Hours not assignes.");
            }else{
                System.out.println("WorkingHours: " + this.getWorkingHours());
            }
    }

}
```

10.2 Code of Lecturer.java

```
public class Lecturer extends Teacher {
    private String Department;
    private int YearsOfExperience;
    private int GradeScore;
    private boolean HasGraded;
```

```
    public Lecturer(int TeacherID, String TeacherName,
    String Address,String EmploymentStatus,
    String Department, int YearsOfExperience, String WorkingType){
        super(TeacherID, TeacherName, Address,
        WorkingType, EmploymentStatus);
        this.Department = Department;
        this.YearsOfExperience = YearsOfExperience;
        this.GradeScore = 0;
        this.HasGraded = false;
    }
    public String getDepartment()
    {
        return this.Department;
    }
    public int getYearsOfExperience()
    {
        return this.YearsOfExperience;
    }
    public int grtGradeScore()
    {
        return this.GradeScore;
    }
    public boolean HasGraded()
```

```
    {
        return this.HasGraded;
    }
    // mutator method for
    public void setGradeScore(int GradeScore){
        this.GradeScore =GradeScore;
    }
    public void GradeScore(int GradeScore,
    String Department, int YearsOfExperience){
        if (this.YearsOfExperience >= 5 && this.Department.equals(Department))
        {
        if (GradeScore >= 70){
            this.GradeScore = GradeScore;
        }
         else if (GradeScore >= 60){
            this.GradeScore = GradeScore;
        }
        else if (GradeScore >= 50){
            this.GradeScore = GradeScore;
        }
        else if (GradeScore >= 40){
            this.GradeScore = GradeScore;
        }else {
```

```
            this.GradeScore = 0;
        }
        HasGraded = true;
    }else{
        System.out.println("lecturer has not graded yet. Please meet the Egilibility.");
    }
    }
    // display method
    public void display(){
        super.display();
        System.out.println("Department: " + Department);
        System.out.println("YearsOfExperience: " + YearsOfExperience);
        if (HasGraded){
            System.out.println("HasGraded: " + HasGraded);
        }else{
            System.out.println("GradeScore: " + GradeScore);
        }
    }
}
```

10.3 Code of Tutor.java

```
public class Tutor extends Teacher{
    private double salary;
```

```
    private String specialization;
    private String academicQualification;
    private int performanceIndex;
    private boolean isCertified;
    //constructor
     public Tutor(int TeacherID, String TeacherName, String Address, String
WorkingType, String EmployementStatus,
     int WorkingHours, double salary, String specialization, String academicQualification,
int performanceIndex)
    {
        super(TeacherID, TeacherName, Address, WorkingType, EmployementStatus);
        this.setWorkingHours(WorkingHours);
        this.salary = salary;
        this.specialization = specialization;
        this.academicQualification = academicQualification;
        this.performanceIndex = performanceIndex;
        this.isCertified = false;

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

```
    public String getSpecialization()
    {
        return this.specialization;
    }
    public String getAcademicQualification()
    {
        return this.academicQualification;
    }
    public int getPerformanceIndex()
    {
        return this.performanceIndex;
    }
    public boolean getIsCertified()
    {
        return this.isCertified;
    }
    //method to set salary
    public void setSalary(double newSalary, int newPerformanceIndex)
    {
        if (newPerformanceIndex > 5 && super.getWorkingHours() > 20) {
        double appraisal = 0;
        if (newPerformanceIndex >= 5 && newPerformanceIndex <=7)
        {
```

```
            appraisal = 0.5;
        }
        else if (newPerformanceIndex >= 8 && newPerformanceIndex <=9)
        {
            appraisal = 0.1;
        }
        else if (newPerformanceIndex == 10)
        {
            appraisal = 0.2;
        }
        this.salary += appraisal * this.salary;
        this.isCertified = true;
    }
    else
    {
        System.out.println("Tutor cannot be certified due to insufficient performanceIndex
or WorkingHours.");
    }
    }
    //method to remove tutor
    public void removeTutor()
    {
        if(isCertified)
```

```
        {
            this.salary = 0;
            this.specialization = null;
            this.academicQualification = null;
            this.performanceIndex = 0;
            this.isCertified = false;
            System.out.println("Tutor removed successfully.");
        }
        else{
            System.out.println("cannot remove certified Tutor.");
        }
    }
    //method to display tutor
    public void displayDetails()
    {
        super.display();  //call method of teacher class
        if(isCertified)
        {
            System.out.println("salary: $" + salary);
            System.out.println("specialization: " + specialization);
            System.out.println("academicQualification: " + academicQualification);
            System.out.println("performanceIndex: " + performanceIndex);
        }
```

```
        else{
            System.out.println("Tutor is not Certified.");
        }
    }
```

The End