Module Code: - CS4001NP

Module Title: -Programming

Coursework-1

Submitted By: -

Student Name: - Manoj Neupane

London Met Id: - 23048832

Group: - C1

Date:-2024/01/24

Submitted To:-

Mr. Sushil Poudel

Module Leader

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

Table of Contents

Table of Figures

Table of Tables

1 Introduction to Project

The goal of this assignment is to teach students the concept of object-oriented programming. To accomplish this, I will be creating a class for a teacher and two subclasses for a lecturer and a tutor. The main aim of this coursework is to enhance our skills and gain a deeper knowledge of how coding works in real-world scenarios.

Java is an object oriented programming language which is very popular. We can write programs that run on a variety of devices using this programming language. Because of its extensive scope, we can design software for a variety of devices, including web pages, ATMs, mobile devices, and Internet of Things (baeldung, 2024).

In coursework, I used bluej tool because of its simple and user friendly GUI (Graphical User Interface). Bluej an Integrated Development Environment is free software which was developed to help beginners and students who has just move in this field of Java.

In this coursework, there is a Teacher parent class and two sub classes of it Lecturer and Tutor which inherits the methods and properties of Teacher class in this coursework. Each has their own attributes and methods. The teacher class contains six attributes: workingType, employment Status, address, teacherName, teacherId, and workingHours. The teacher name, address, working type, employment status are each represented as a string of text and Teacher ID, and working hours as a number. Each has their accessor method. There is a mutator method to assign value to workingHours and also there is display method to display the details by checking certain conditions. The Lecturer class is a subclass of Teacher class and it has four attributes: Department - a String, YearsOfExperience - an integer, gradedScore - an integer, hasGraded – Boolean .Each has their own accessor method and a mutator for gradedScore. There is a gradeAssignment method which accepts three parameter gradedScore, department and yearsOfExperience and checks the certain cases as mentioned in appendix below and assigns the grade to the students. There is also a display method which is overridden from Teacher class which prints the details of Lecturer by checking certain conditions.

The Tutor class is also a subclass of Teacher class and has five attributes: salary - a double specialization - a String, academic qualifications - a String, performanceIndex - an Integer, isCertified - a Boolean. Each has their accessor. There is a method to set a salary of tutor by analysing his/her workingHours and performanceIndex. It has a contructor which accepts 10 parameters i.e. (Teacher Id, teacher name, address, working type, employmentStatus, workingHours, salary, specialization, academic Qualifications and performanceIndex). There is also a display method which is overridden from Teacher class which displays details of Tutor by checking certain cases.

2 Class Diagram

A class diagram is like a framework of a model system, outlining class objects and their categories. Each class is presented as a rectangle with three parts: the name, attributes, and operations. These diagrams are very valuable in the design process as they helps developers in discussions and decision-making regarding methods and features, ultimately leading to a streamlined and organized codebase (Fonseca, 2023). A class possesses unique characteristics, known as its attributes, which are necessary pieces of information that must be stored. Interactions with other classes and the instructions for creating specific messages are referred to as methods. These methods can also be referred to as operations (Nalimov, 2021).

For instance, In our scenario there are three class i.e. a parent class Teacher and two child classes Lecturer and Tutor which contains their separate attributes, method and a method which are overridden from parent class. Class diagram of each is given below:-

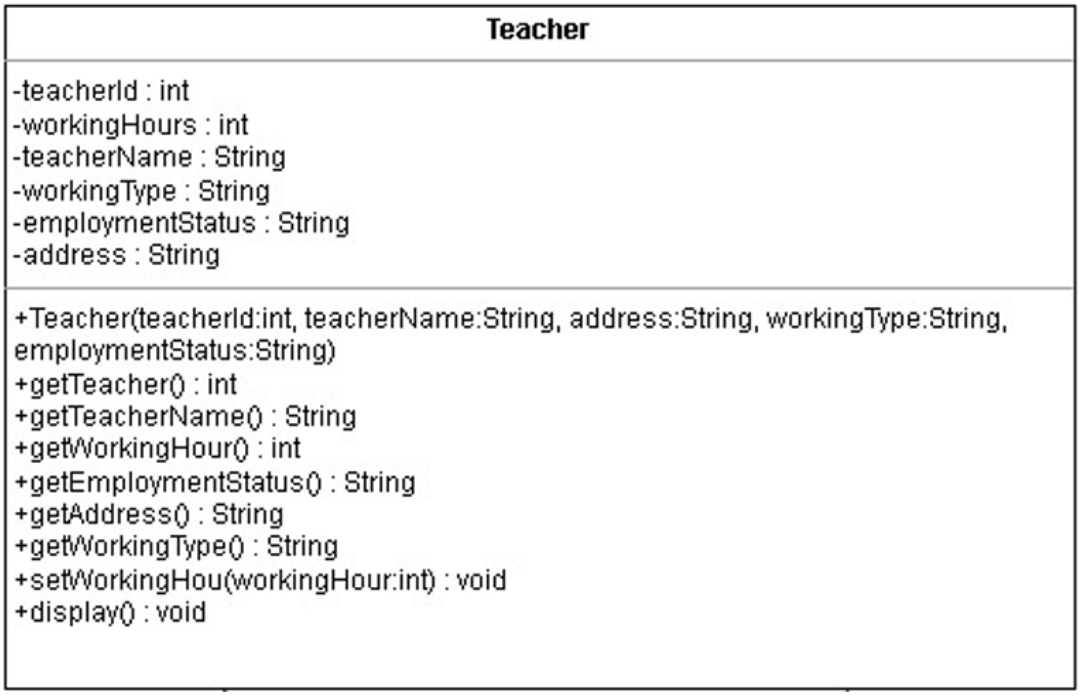

Figure 1: Class Diagram of Teacher

This is a class diagram of Teacher. It contains different methods, attributes.

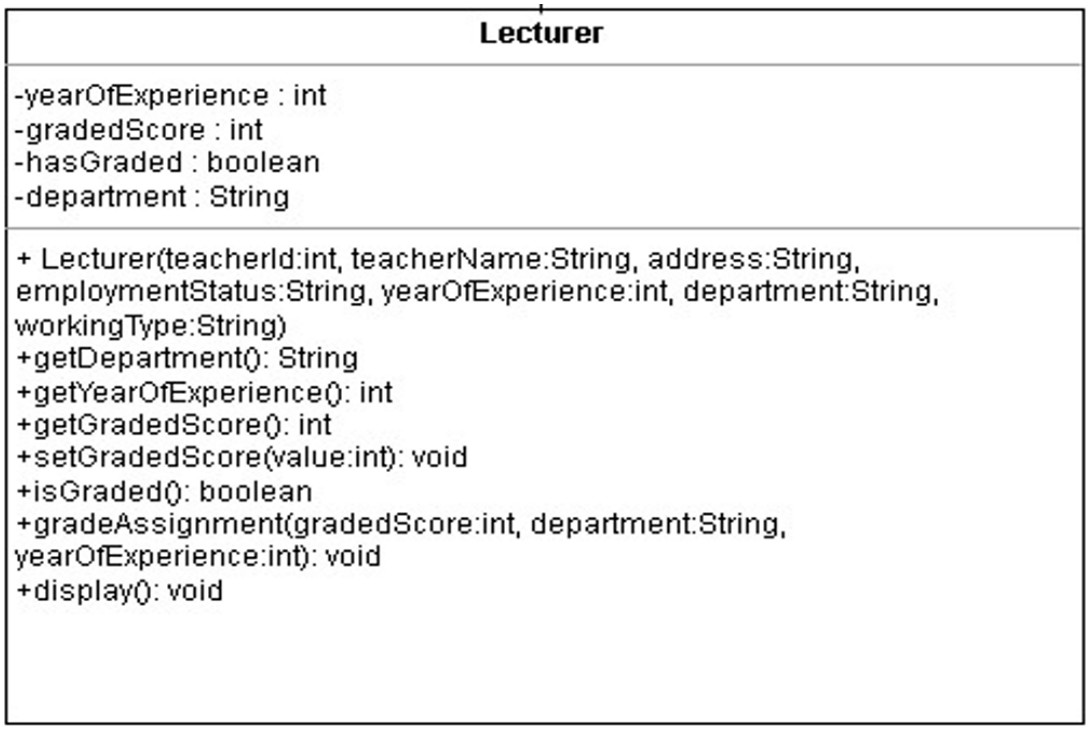

Figure 2: Class Diagram of Lecturer

This is a class diagram of Lecturer class which is extended from Teacher class. It contains different methods, attributes where some of them are its own and some of them are inherited from parent class Teacher.

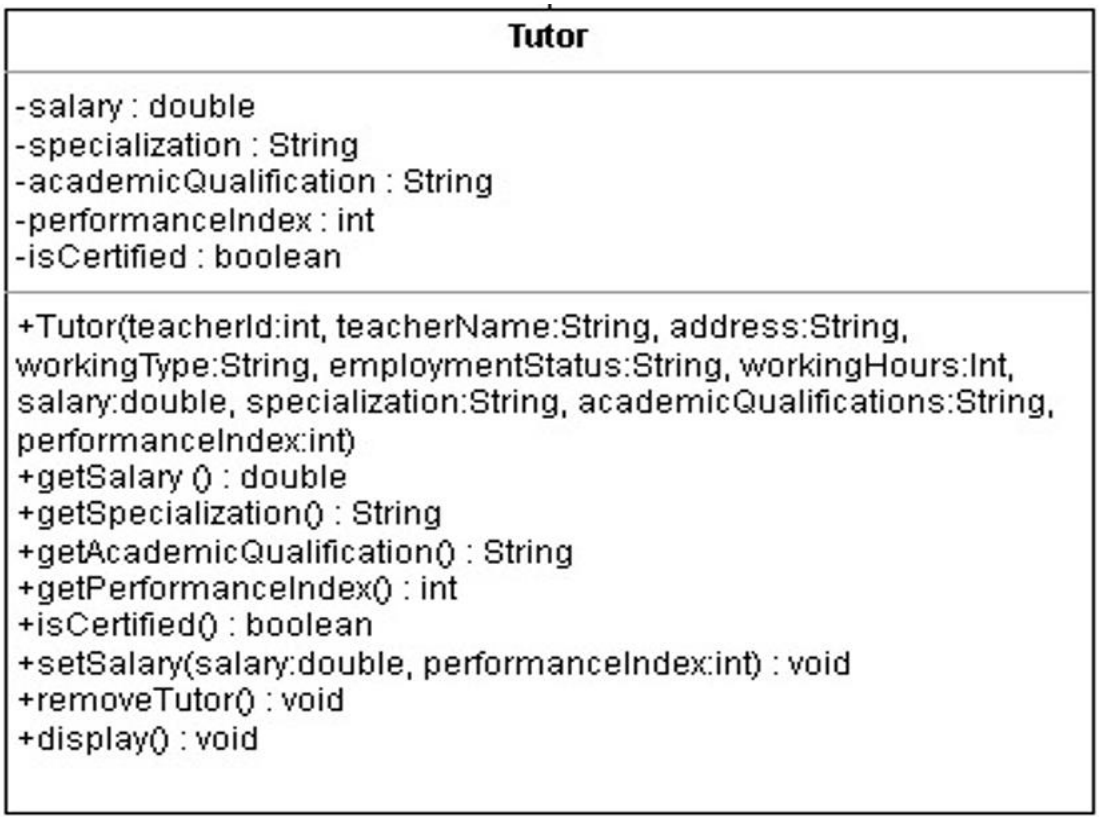

Figure 3: Class Diagram of Tutor

This is a class diagram of Tutor class which is extended from Teacher class. It contains different methods, attributes where some of them are its own and some of them are inherited from parent class Teacher

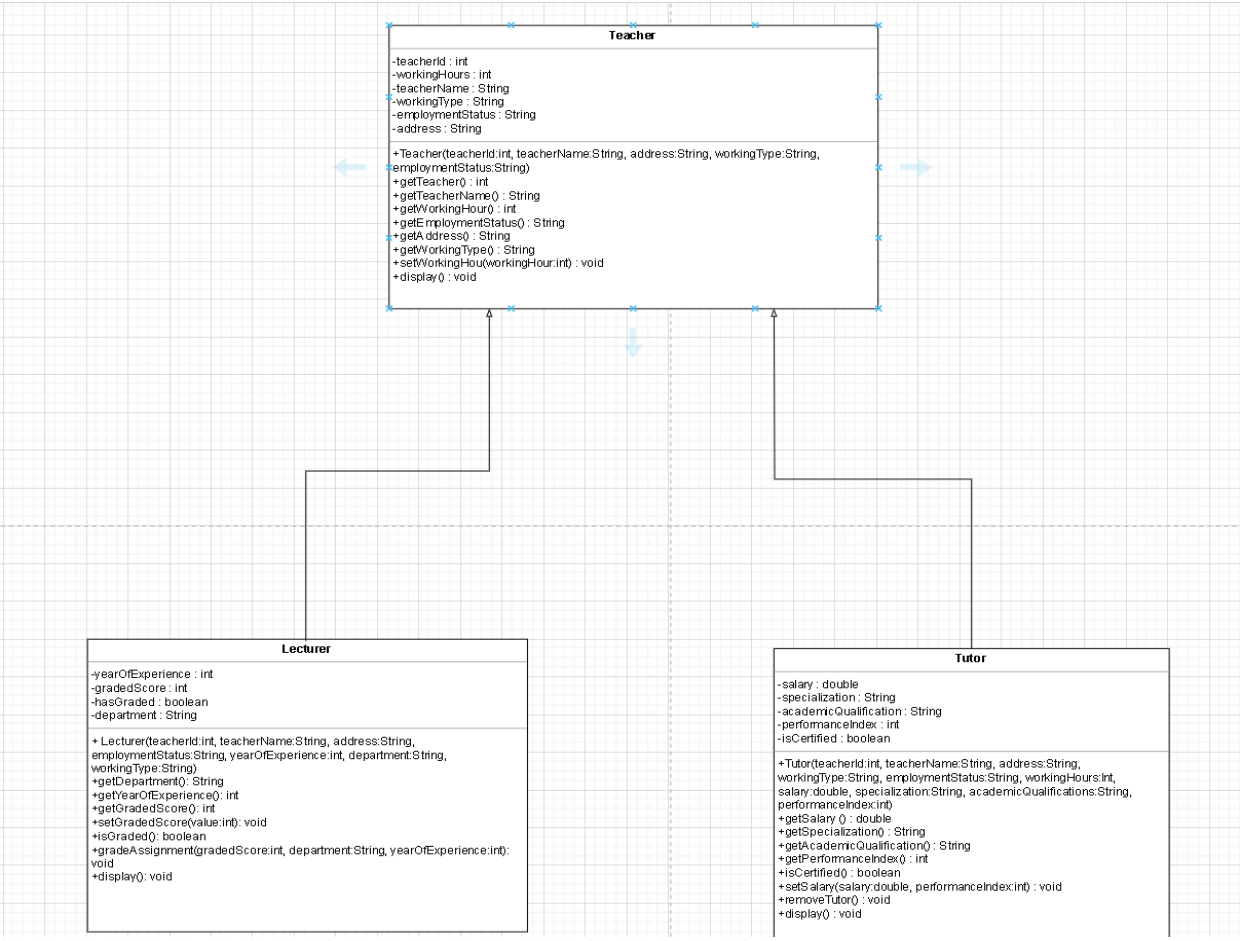

Figure 4: Class Diagram of parent class teacher and sub classes.

This is a class diagram of parent class Teacher and sub classes Lecturer and Tutor which shows how they are interrelated. From the figure, we can see Lecture and Tutor are derived from Teacher class.

3 Pseudocode

Pseudocode is a powerful tool for translating complex processes and programming concepts into everyday language. It acts as a bridge between programming code and human understanding, providing a comprehensive and clear outline of what the code should achieve. Essentially, it serves as a preliminary draft or blueprint for a program before it is written in a specific programming language. Some alternate names for pseudocode include "dummy code" and "code representation." Its main purpose is to simplify the algorithm or process by providing a step-by-step explanation of the actions

without any specific language syntax (Mahr, 2023).Here is the actual pseudocode of Teacher, lecturer, Tutor class.

3.1 Pseudocode of Teacher class

BEGIN

DECLARE a class named Teacher

DECLARE a private Integer teacherId

DECLARE a private Integer workingHours

DECLARE a private String teacherName

DECLARE a private String workingType

DECLARE a private String employmentStatus

DECLARE a private String address

CREATE a parameterized constructor for Teacher with parameter

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

START

INITIALIZE instance variable teacherName with parameter value

INITIALIZE instance variable address with parameter value

INITIALIZE instance variable workingType with parameter value

INITIALIZE instance variable teacherId with parameter value

INITIALIZE instance variable emplomentStatus with parameter value

END

CREATE an accessor method to get teacherId

CREATE an accessor method to get teacherName

CREATE an accessor method to get workingHour

CREATE an accessor method to get employmentStatus

CREATE an accessor method to get address

CREATE an accessor method to get workingType

CREATE a mutator method to set workingHours

CREATE a method named display

START

IF workingHours is empty

THEN DISPLAY "Kindly, fill the working hours. (workingHours is empty)"

ELSE

DISPLAY"teacherId, workingHours, teacherName,

workingType, employmentStatus, address"

ENDIF

END

END

3.2 Pseudocode of Tutor class

START PROGRAM

CREATE class named Tutor which sub-class of Teacher class

DECLARE a private double salary

DECLARE a private String specialization

DECLARE a private String academicQualification

DECLARE a private int performanceIndex

DECLARE a private boolean isCertified

CREATE a parametrized constructor with parameter values teacherId(int), teacherName(String), employmentStatus(String), workingHours(int), salary(double), specialization(String), academicQualification(String), performanceIndex(int)

START

CALL the parent constructor with values teacherId(int), teacherName(String), address(String), workingType(String), employmentStatus(String) using super keyword

CALL the mutator of workingHours of Parent class Teacher by passing workingHours as an argument

INITIALIZE the instance variable salary, specialization, academicQualificaion, performanceIndex with respective parameter value

INITIALIZE the instance variable isCertified to false

END

CREATE accessor methods to get salary, specialization, academicQualification, performanceIndex, isCertified for each separately

CREATE a mutator method to set salary which accepts three parameter values newSalary(double), performanceIndex(int), workingHours(int)

START

IF performanceIndex is greater than 5 and workingHours is greater than 20

IF performanceIndex is greater than 5 and performanceIndex is smaller than or equals to 7

SET instance variable salary to (newSalary + 5% of newSalary)

ELSE IF performanceIndex is greater than 8 and performanceIndex is smaller than or equals to 9

SET instance variable salary to (newSalary + 10% of newSalary)

ElSE IF performanceIndex is greater than or equals to 10

SET instance variable salary to (newSalary + 20% of newSalary)

ENDIF

SET isCertified to true

ELSE

DISPLAY a suitable message

ENDIF

END

CREATE a method named removeTutor

START

IF isCertified is equals to false

THEN SET instance varirable salary to 0, (specialization, academicQualification) to empty String, performanceIndex to 0 and isCertified to false;

ELSE

DISPLAY a suitable message

ENDIF

END

CREATE a display method which overrides display method of Teacher class

START

IF isCertified is equals to True

THEN CALL the display method of parent class

DISPLAY "salary, specialization, academicQualification, performanceIndex"

ELSE

CALL the display method of parent class

ENDIF

END

END program

3.3 Pseudocode of Lecturer class

BEGIN program

DECLARE a class named Lecturer which is sub-class the Teacher class

DECLARE a private Integer yearOfExperience

DECLARE a private Integer gradedScore

DECLARE a private boolean hasGraded

DECLARE a private String department

CREATE a parameterized constructor for Lecture with parameter values teacherId(int), teacherName(String), address(String), workingType(String), employmentStatus(String), yearOfExperience(Integer), department(String), workingType(String)

START

Call the constructor of teacher class with values teacherId(int),teacherName(String),address(String),workingType(String),employmentStatus(String)

ASSIGN gradedScore variable as 0

ASSIGN hasGraded variable as false

INITIALIZE instance variable yearOfExperience with parameter value

INITIALIZE instance variable department with parameter value

END

CREATE an accessor method to get department

CREATE an accessor method to get yearOfExperience

CREATE an accessor method to get gradedScore

CREATE an accessor method to get employmentStatus

CREATE an accessor method to get address

CREATE an accessor method to get hasGraded

CREATE a mutator method to set gradedScore

CREATE a method named gradeAssignment with parameter values gradedScore(int), department(String), yearOfExperience(Integer)

START

IF yearOfExperience >= 5

THEN IF instance variable department is equals to local department variable

THEN IF gradedScore is smaller than 0 and greater than 100

DISPLAY "gradedScore is invalid"

ELSE IF gradedScore is greater than or equals to 70

DISPLAY "Graded Score is A"

ELSE IF gradedScore is greater than or equals to 60 and smaller than 70

DISPLAY "Graded Score is B"

ELSE IF gradedScore is greater than or equals to 50 and smaller than 60

DISPLAY "Graded Score is C"

ELSE IF gradedScore is greater than or equals to 40 and smaller than 50

DISPLAY "Graded Score is D"

ELSE

DISPLAY "Graded Score is E"

ENDIF

ASSIGN hasGraded to true

ELSE

DISPLAY "teacher is not in same area of interest"

ENDIF

ELSE

DISPLAY "Teacher has low experience"

ENDIF

END

CREATE a display which override the method of teacher class

START

IF hasGraded is true

THEN call the display method of parent class

DISPLAY "department, yearOFExperience, gradedScore"

ELSE

DISPLAY "grade has not assigned yet"

ENDIF

END

END program

4 Description of Methods

4.1 Description of Methods of Teacher Class

- **Constructor: - Teacher**

  It accepts five parameter i.e. :-(teacherId, teacherName, address, workingType, employmentStatus) and assign them to respective instance variable.

- **Methods**
  - **getTeacherId()**

    It returns value of teacherId variable which is integer.

  - **getTeacherName()**

    It returns value of teacherName instance variable which is String.

  - **getWorkingHours()**

    It returns value of workingHours instance variable which is integer.

  - **getEmploymentStatus()**

    It returns value of employmentStatus instance variable which is String.

  - **getAddress()**

    It returns value of address instance variable which is String.
  - **getWorkingType**()

    It returns value of workingType instance variable which is String.
  - **setWorkingHour**()

    It accepts a parameter and set given parameter value to instance variable workingHour
  - **display**()

    It displays “warning message if workingHours variable is empty”Otherwise it displays the details i.e. (teacherName, address, workingType, employmentStatus, teacherId, workingHours) of teacher

4.2 Description of Methods of Lecturer Class

- **Constructor : Lecturer()**

  Lecturer class constructor with 7 parameters i.e. (teacherId, teacherName, address, workingType, employmentStatus, yearOfExperience, department, workingType) where call to the constructor of parent class is made with five parameters with the assignment of different variables like (department, yearOfexperience) with respective parameters and hasGraded, gradedScore to false and 0 respectively

- **Methods**

  - **getDepartment**()

    It returns String value stored in department variable.
  - **getYearOfExperience**()

    It returns integer value stored in yearOfExperience variable.
  - **getGradedScore**()

    It returns integer value stored in gradedScore variable.
  - **setGradedScore**()

    It accepts an integer value as a parameter and assign it to gradedScore instance variable.
  - **isGraded**()

    It returns Boolean variable stored in isGraded variable.
  - **gradeAssignment**()

    This method accepts 3 parameters and assigns the gradedScore by checking whether the teacher is qualified or not to grade the students. Condition (lecturer experience must be greater than 5 and lecturer must be same area of department)
  - **display**()

    Display method displays the details of Lecturer if lecturer has assign the grade to students otherwise displays warning message to grade the assignment.

4.3 Description of methods of Tutor Class

- **Constructor**: Tutor()

  It accepts 10 parameters i.e. (teacherId, teacherName, address, workingType, employmentStatus, workingHours, salary, specialization, academicQualification, performanceIndex) and assign them to respective instance variable along with call to the accessor of workingHours of parent class and assignment of isCertified to false.

- **Methods**
  - **getSalary**()

    It returns double value stored in salary.
  - **getSpecialization**()

    It returns String value stored in specialization.
  - **getAcademicqualification**()

    It returns String value stored in academicQualification.
  - **getPerformanceIndex**()

    It returns integer value stored in isCertified.
  - **isCertified**()

    It returns boolean value stored in isCertified.
  - **setSalary**()

    It accepts two parameters and calculate the salary on the basis of different condition like checks tutor's workingHours and performanceIndex to assign the salary.
  - **removeTutor**()

    This methods sets the different variable like salary to 0, specialization to empty string, academicQualification to emptyString, performanceIndex to 0 and isCertified to false if isCertified is false otherwise shows warning message by displaying "removing tutor invalid teacher is certified"
  - **display**()

    This methods shows the details i.e (name, id, workingHours, workingType, employmentStatus, salary, specialization,

academicQualification, performanceIndex) of tutor if he/she is certified, otherwise shows details only like (name, id, workingHours, workingType, employmentStatus).

5 Testing

5.1 Test-1

Table 1: Test 1

Objectives	Inspect the Lecturer class, grade the assignment, and re-inspect the Lecturer Class
Action	1. Create an object of Lecturer class i.e. (Lecturer1). 2. Fill the details and made call to the gradeAssignment() method. 3. Re-inspect the object of Lecturer class
Expected Outcomes	It should Display “B grade” and hasGraded should be true.
Actual Output	“B grade displayed” and hasGraded value changed to true
Conclusion	Test successful

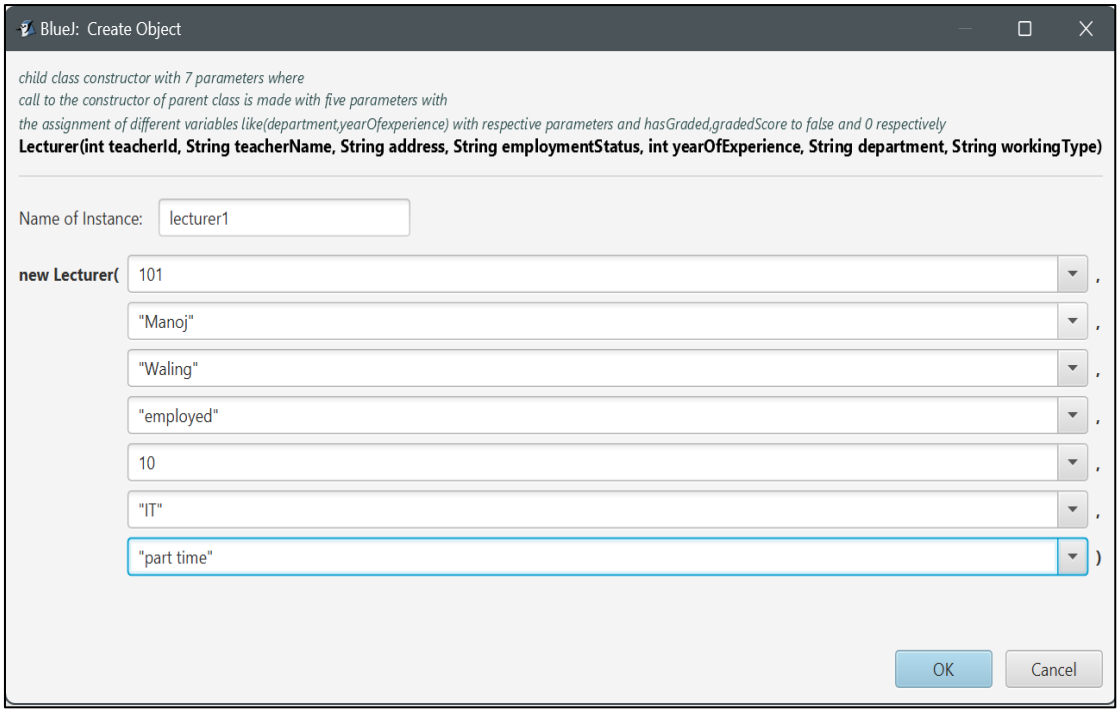

Figure 5: Creation of Lecturer class object with values

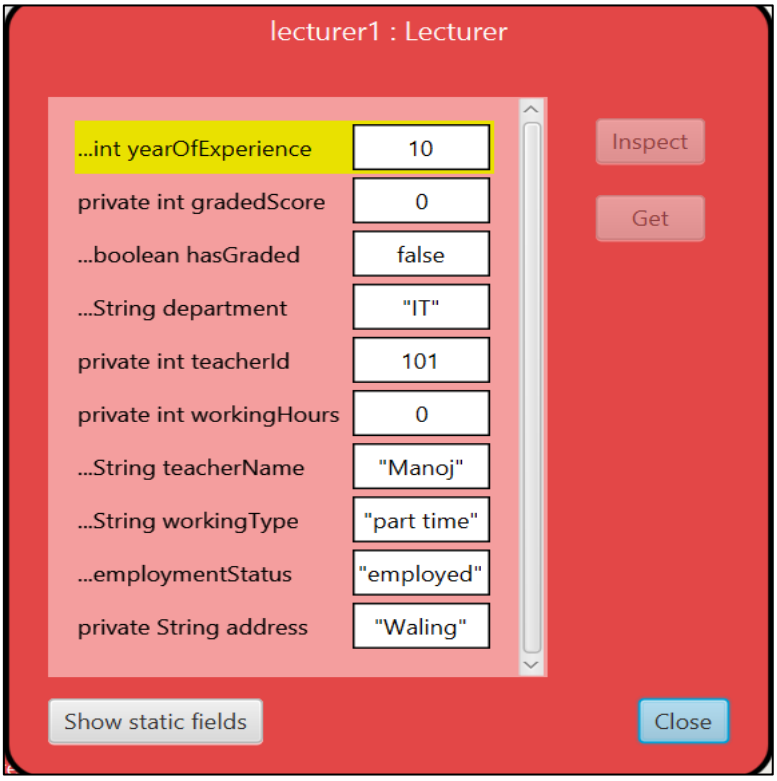

Figure 6: Inspecting Lecturer before grading the assignment

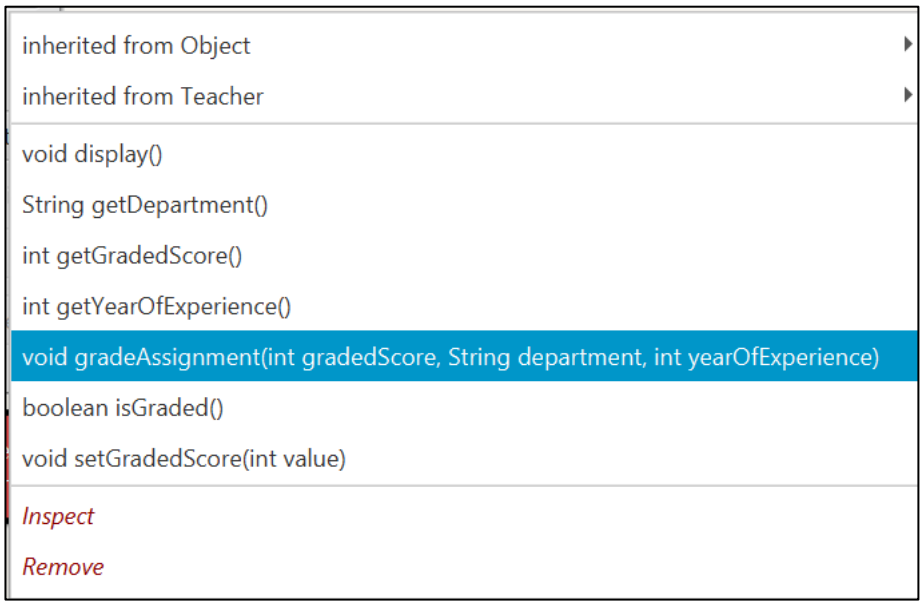

Figure 7: Call to the gradeAssignment method

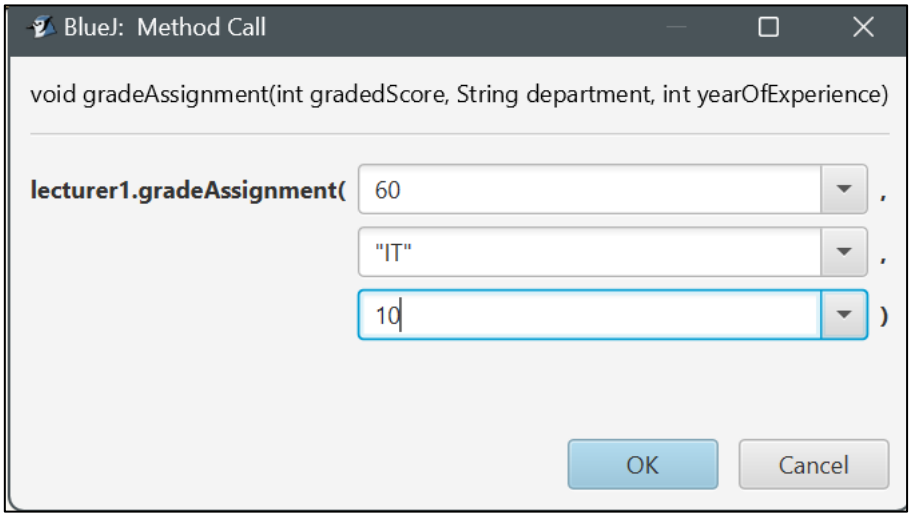

Figure 8: Filling the parameter values of gradeAssignment

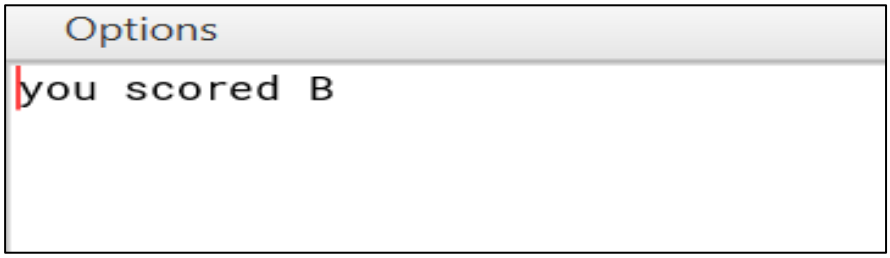

Figure 9: Score

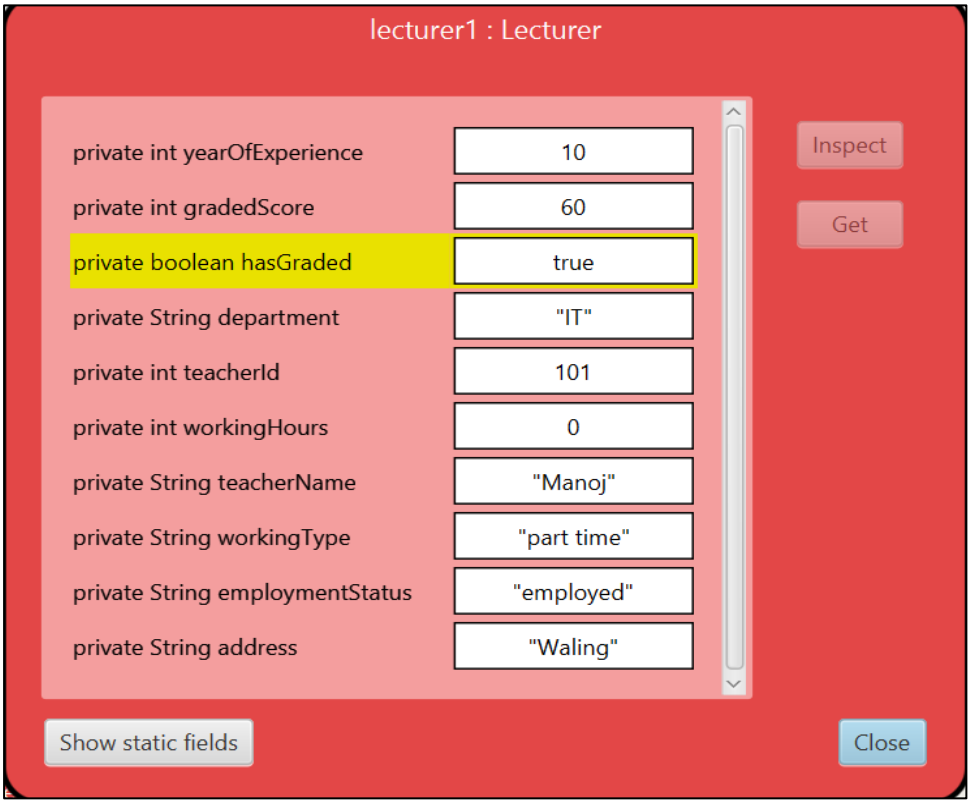

Figure 10: Inspecting after grade assignment

5.2 Test-2

Table 2: Test-2

Objectives	Inspect Tutor class, set salary and re-inspect the Tutor class
Action	1. Create an object of Tutor class i.e. (Tutor1). 2. Fill the details and made call to the setSalary() method. 3. Re-inspect the Tutor1 object.
Expected Output	Salary should change to 24000 and isCertified should be true.
Actual Output	Salary changed to 24000 and isCertified is true.
Conclusion	Test Successful

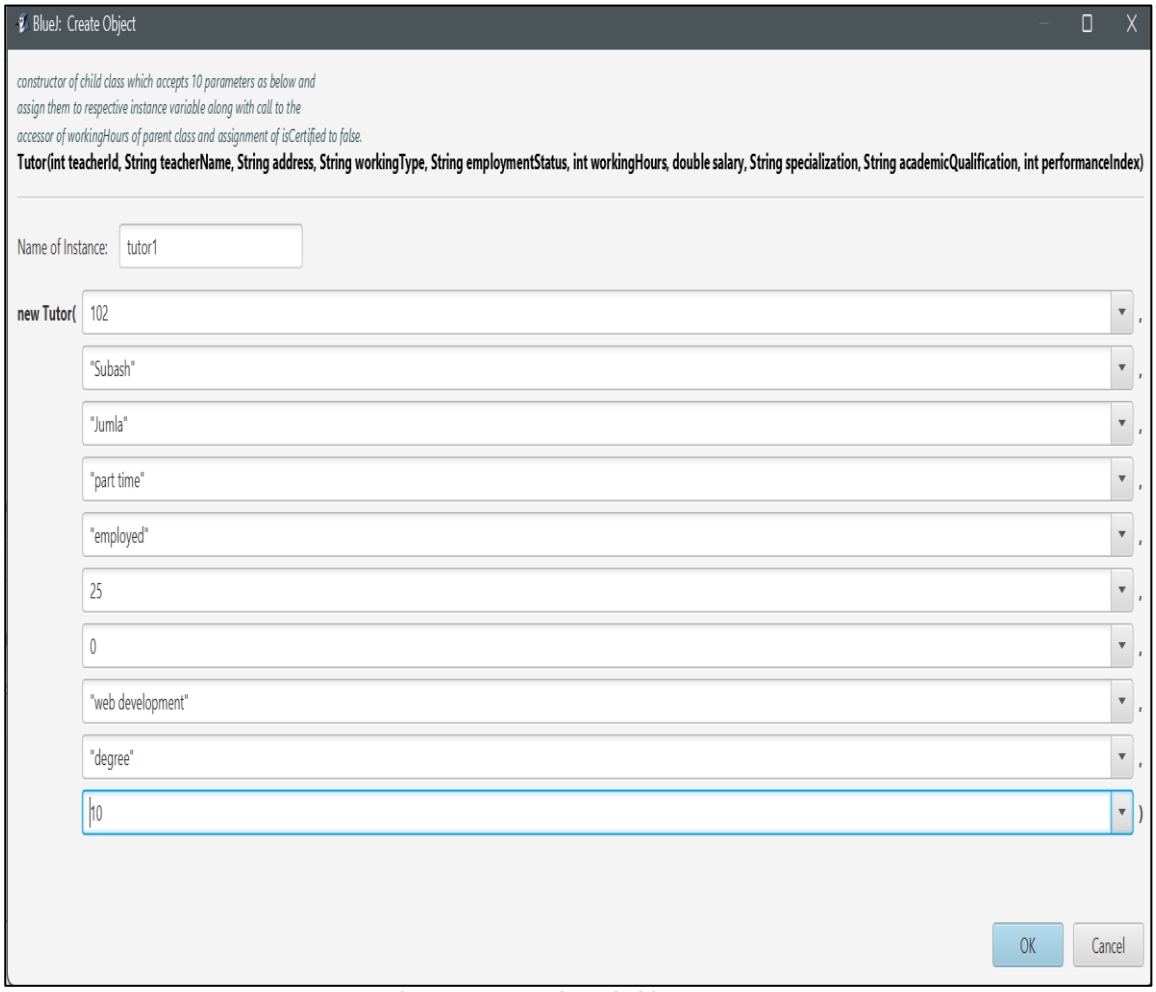

Figure 11: Creation of object tutor1

Figure 12: Inspecting before assigning salary

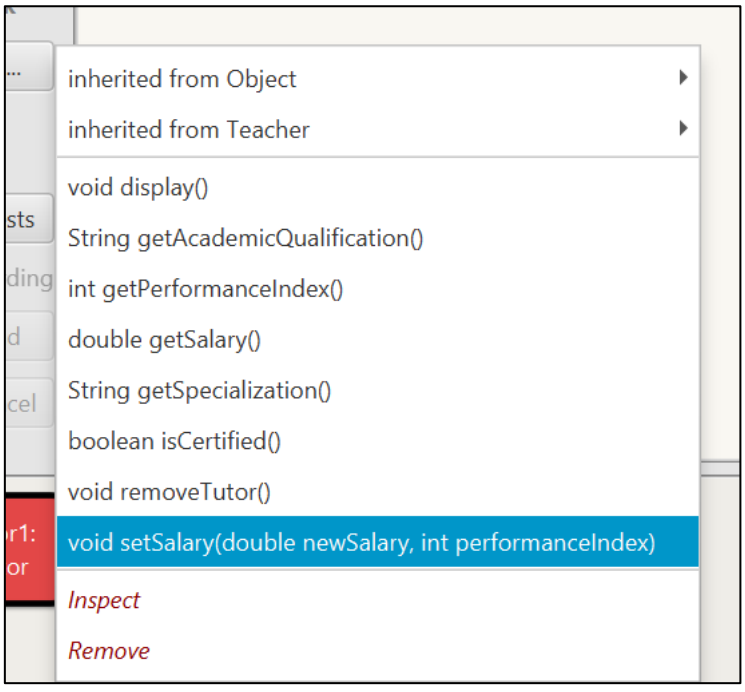

Figure 13: Call to the setSalary method

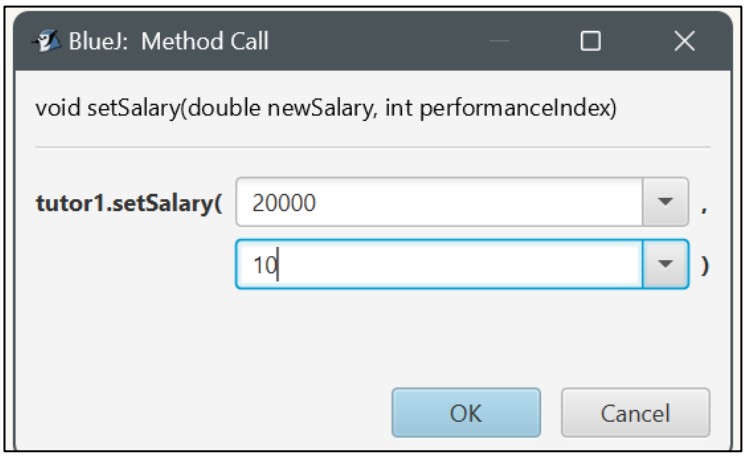

Figure 14: Assigning value to parameters

Figure 15: Inspecting after assigning salary

5.3 Test-3

Table 3: Test 3

Objectives	Inspect Tutor class again after removing the tutor.
Action	1. Create object of Tutor class i.e. (Tutor1) and fills the details. 2. Call removeTutor method after filling values and inspect the values again 3. Re-inspect the Tutor1 object
Expected Output	Details of the tutor like salary, performanceIndex, academicQualification should be empty.
Actual Output	All the above listed details became empty.
Conclusion	Test Successful

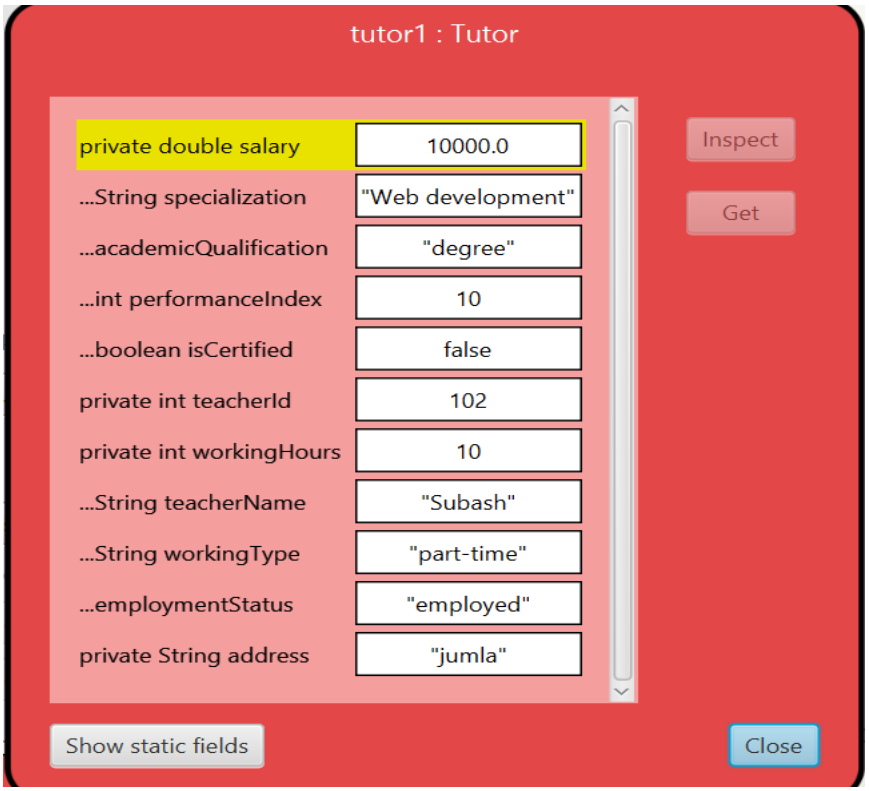

Figure 16: Inspecting before calling removeTutor

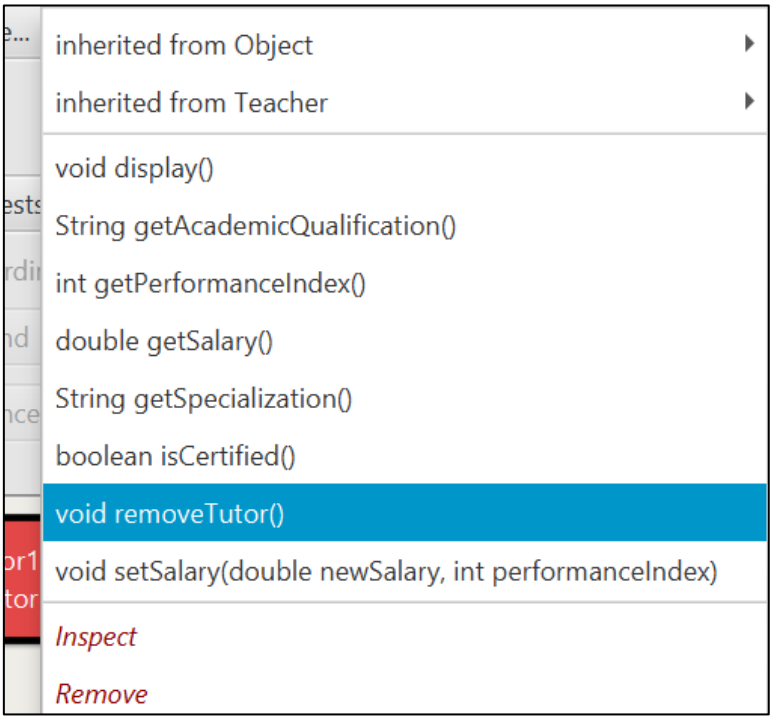

Figure 17: Call to the removeTutor () method

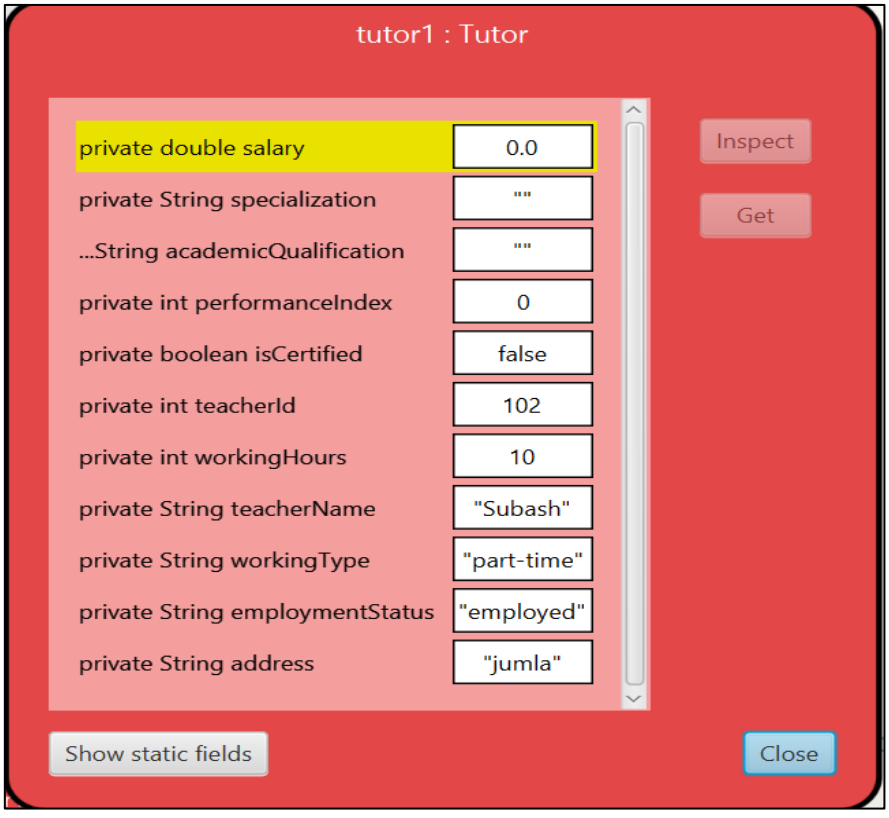

Figure 18: Inspecting after calling remove tutor

5.4 Test-4

Table 4: Test-4

Objectives	Display the details of Lecturer and Tutor classes.
Action	1. Create an object of Lecturer class and Tutor Class i.e.(Lecturer1) 2. Made call to display() method after filling details i.e.(teacherName, teacherId, workingTypes, workingHours , gradedScore, department and performanceIndex) of lecture and Tutor 3. Display the details of lecturer and tutor
Expected Outcomes	There should display all the details
Actual Output	All the details displayed
Conclusion	Test Successful

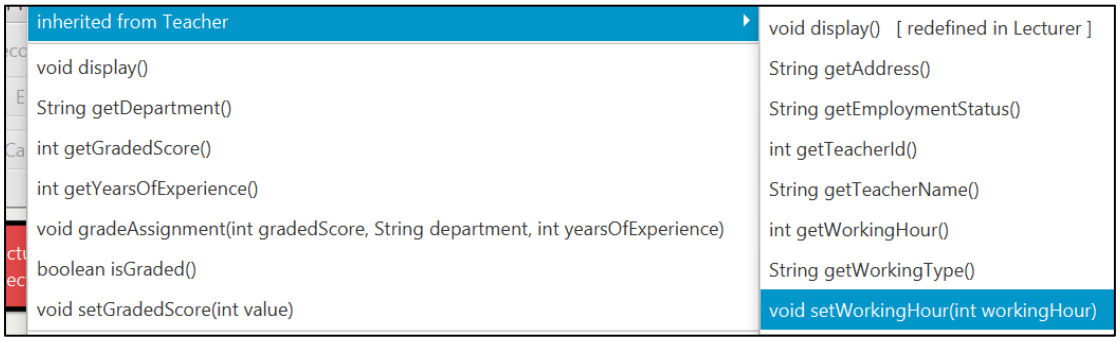

Figure 19: Calling setWorkingHours method

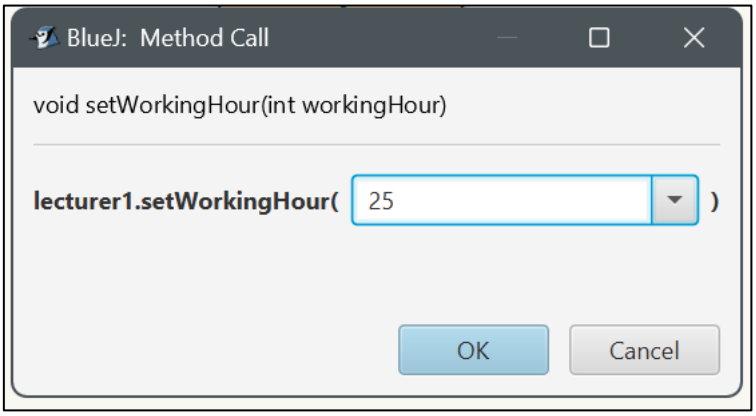

Figure 20: Entering value of setWorkingHours

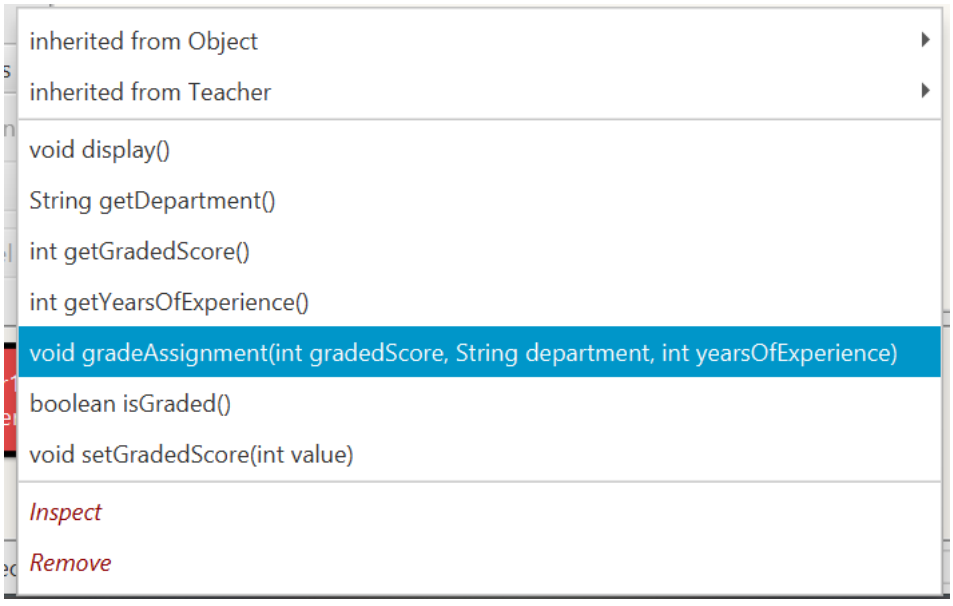

Figure 21: Call to the grade Assignment Method

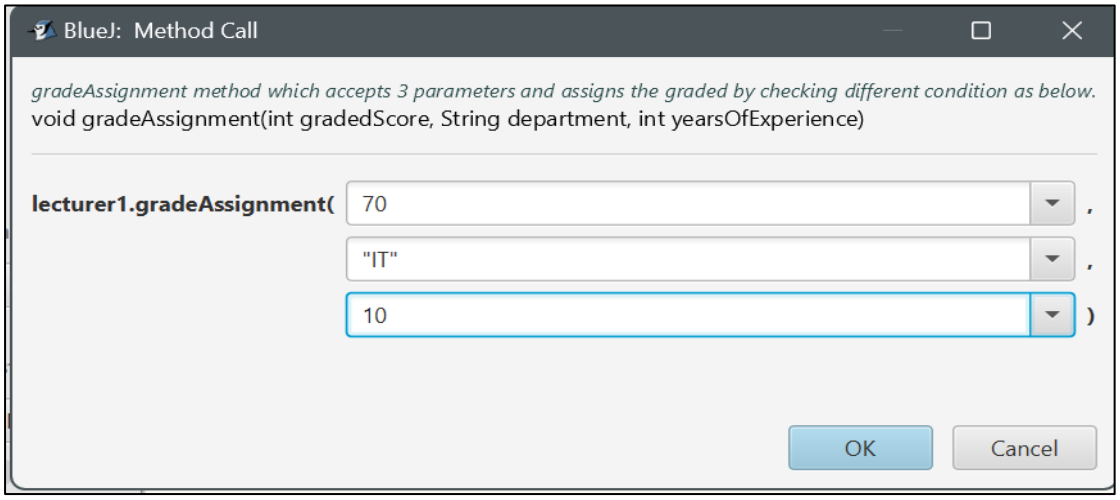

Figure 22: Filling data of gradeAssignment

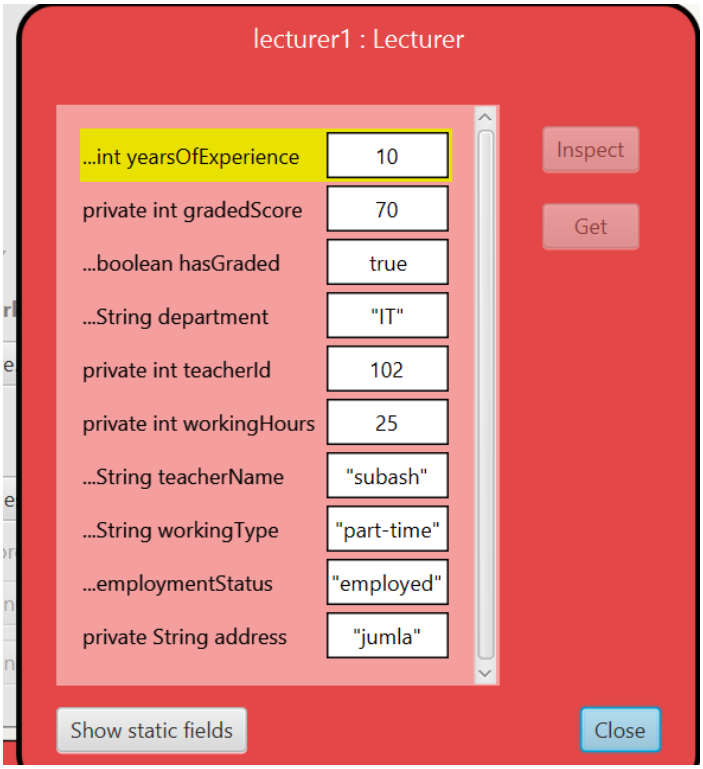

Figure 23: Inspecting lecturer Details

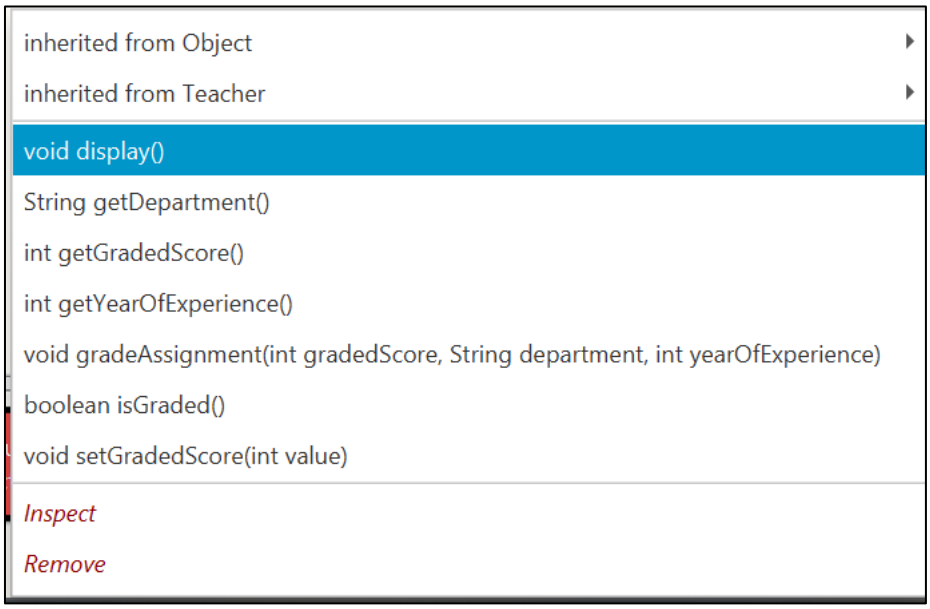

Figure 24 : Call to the display method of Lecturer

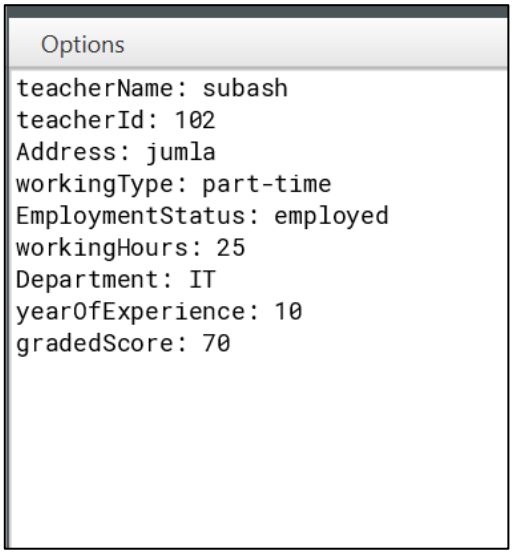

Figure 25: Displaying details of Lecturer

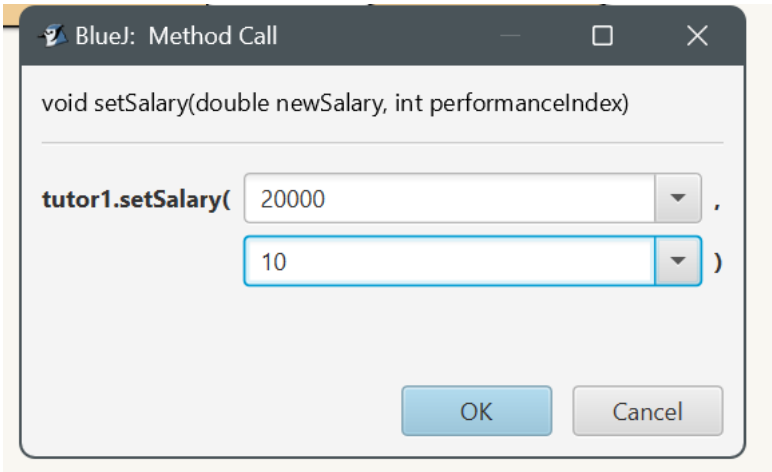

Figure 26: Passing values in setSalary method of Tutor

Figure 27: Inspecting Tutor Details

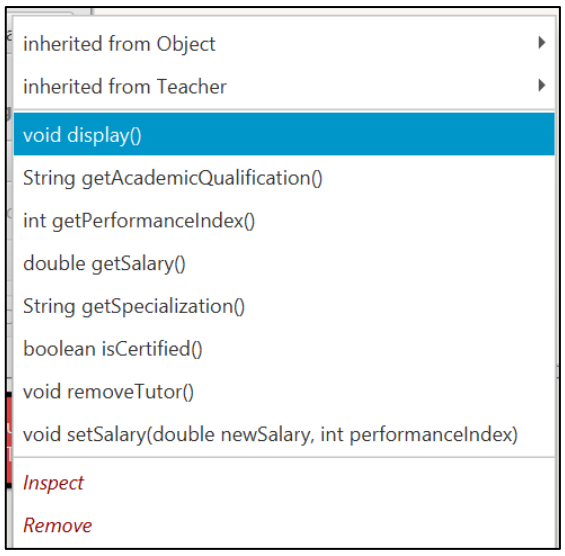

Figure 28 : Calling display method of Tutor class

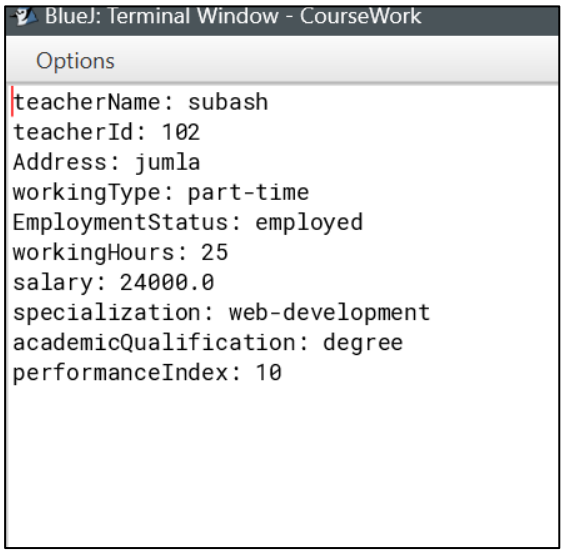

Figure 29: Displaying details of Tutor

6 Error Detection and Correction

6.1 Logical Error Detection and Correction

Logical error usually occurs when user or programmer's mistakenly calculate their logic wrong. It mainly occurs in logical section like replacing "+" operator with "-" or using greater than">" operator while actual output is expecting smaller than "<" operator. In my scenario, I also experienced some of the logical error while doing coursework which is given below:-

While giving input 80 in gradeAssignment section the expected output is "A" but "E" grade was given. I found the error was in my (else if section) where is I mistakenly wrote the "<" instead of ">" so I corrected later and I got as expected output.

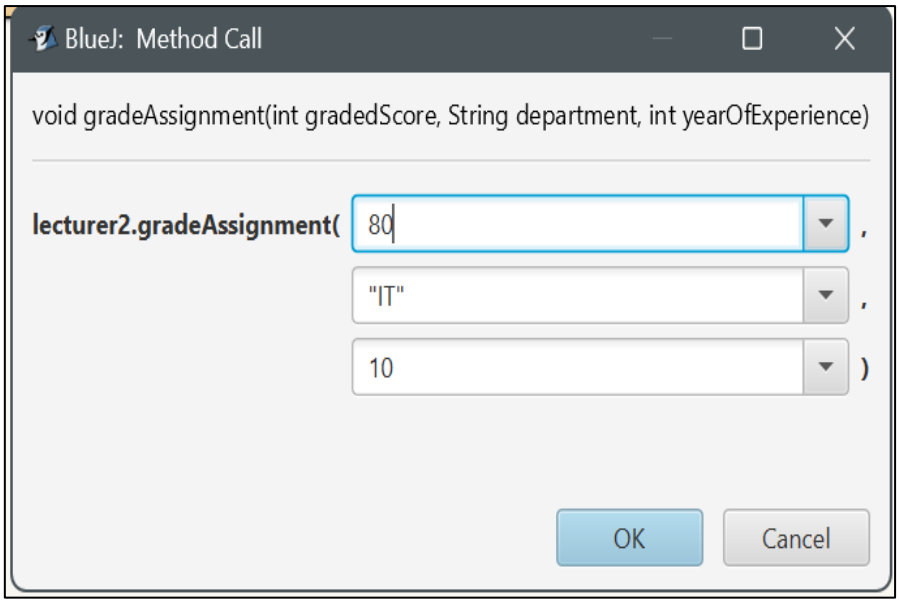

Figure 30: Input to the gradeAssignment method

```
if(gradedScore<0 && gradedScore>100){
    System.out.println("gradedScore is invalid");
}
else if(gradedScore <= 70) {
    System.out.println("you scored A");

}
else if(gradedScore >= 60 && gradedScore < 70) {
    System.out.println("you scored B");
}
else if(gradedScore >= 50 && gradedScore < 60) {
    System.out.println("you scored C");
}
else if(gradedScore >= 40 && gradedScore < 50) {
    System.out.println("you scored D");
}
else {
    System.out.println("you scored E");
}
```

Figure 31: Logical Error details

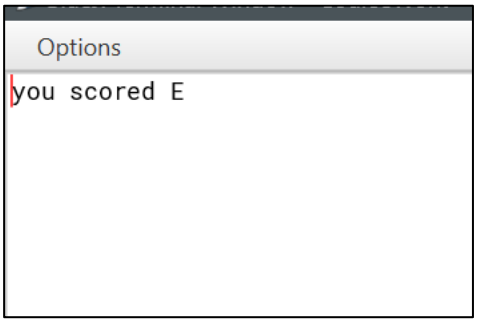

Figure 32: Error Output (Expected A))

```
if(gradedScore<0 && gradedScore>100){
    System.out.println("gradedScore is invalid");
}
else if(gradedScore >= 70) {
    System.out.println("you scored A");

}
else if(gradedScore >= 60 && gradedScore < 70) {
    System.out.println("you scored B");
}
else if(gradedScore >= 50 && gradedScore < 60) {
    System.out.println("you scored C");
}
else if(gradedScore >= 40 && gradedScore < 50) {
    System.out.println("you scored D");
}
else {
    System.out.println("you scored E");
}
```

Figure 33: Logical Error Corrected Details

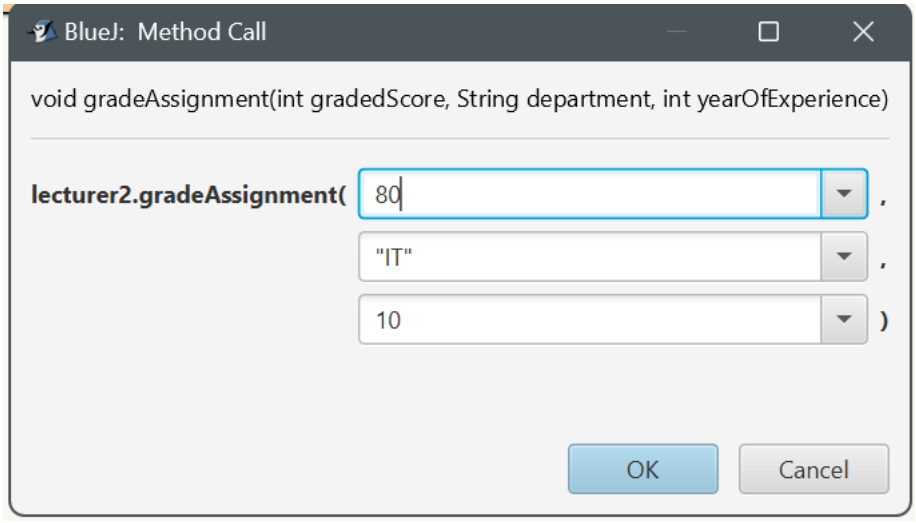

Figure 34: Input to the grade Assignment 2

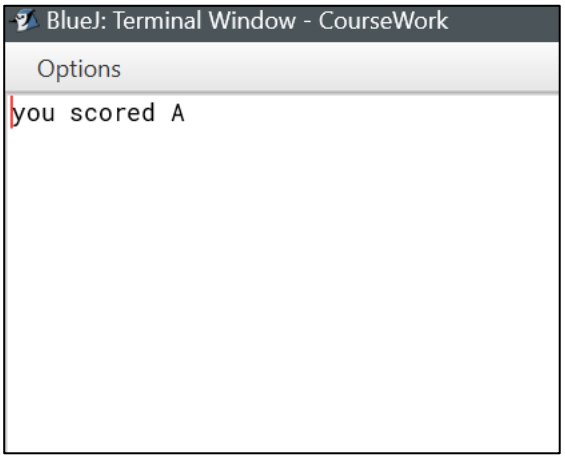

Figure 35: Expected Output

6.2 Runtime Error Detection and Correction.

This type of error usually occurs after successfully compiled a program and during execution of the program. While doing coursework, I found a runtime error in code while calling display method of Teacher class within an overridden display method of lecturer sub class without super keyword. Here is details of error:

In my code, I got runtime error while running the display method. This error occurred because of the confusion to compiler. It calls the same displayed method again and again which is also known as recursive call and eventually after continuous call to the same method. Call stack became full and started to overflow that's why I got stack overflow error. I corrected it just by using super keyword to clear the confusion between display method of parent class Teacher and child class.

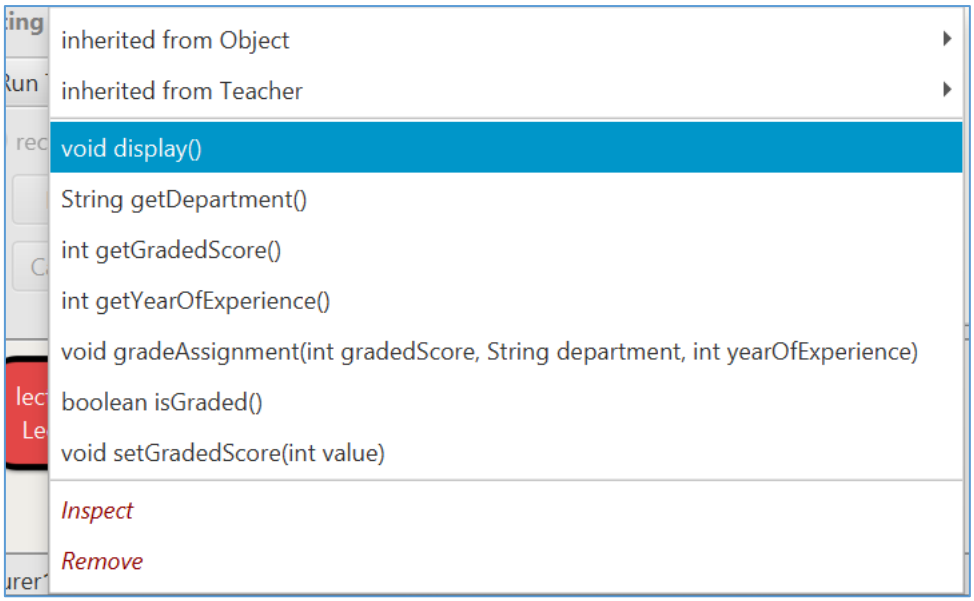

Figure 36: Call to the display method

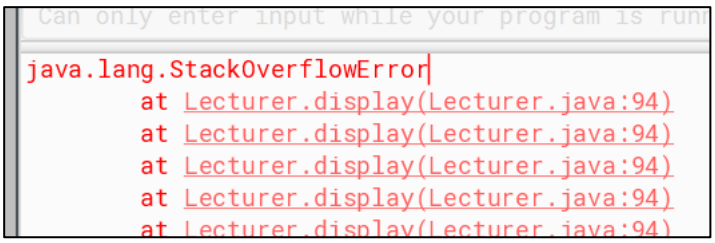

Figure 37: Stack Overflow Error

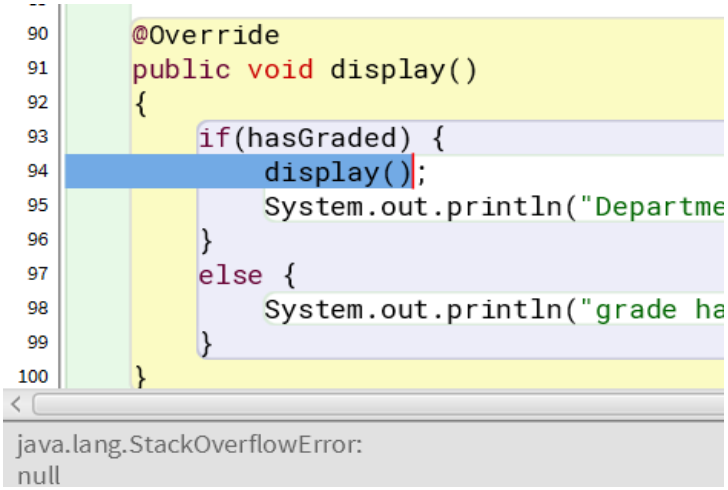

Figure 38: Error detected-1

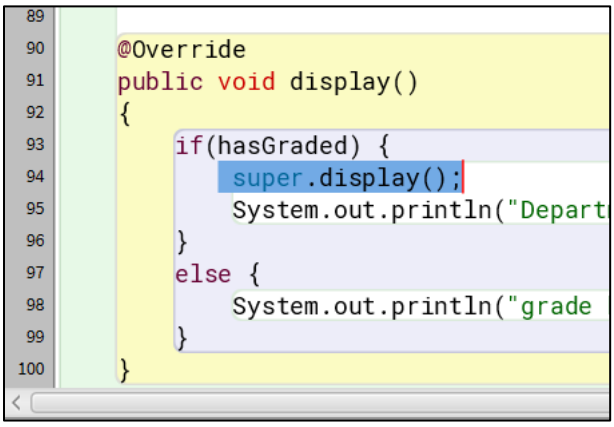

Figure 39: Correction of Run Time Error

6.3. Compile Time Error

This type of error generally occurs when a programmer misses some mandatory things like semicolon at the end of code, two parenthesis while creating methods, Representation of string using double quotes “” , connecting two type of variable using “+” operator while displaying or concatenating. While doing coursework, I faced some of the by mistakenly which is given below:

I got this error while missing semicolon and also the parenthesis while ending constructor of Tutor class. I corrected them by adding those missing characters.

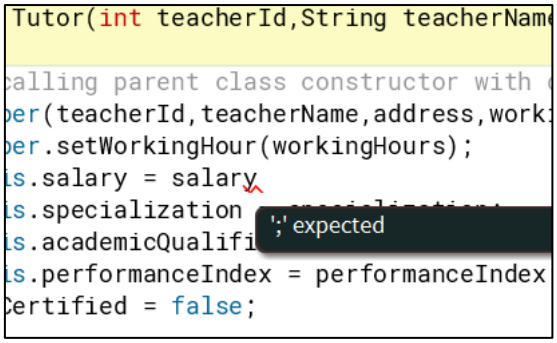

Figure 40: Compile Error Detection-1

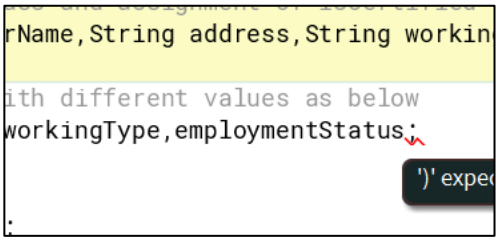

Figure 41: Compile Error Detection-2

```
//calling parent class constructor with different values as below
super(teacherId,teacherName,address,workingType,employmentStatus);
super.setWorkingHour(workingHours);
this.salary = salary;
this.specialization = specialization;
this.academicQualification = academicQualification;
```

Figure 42: Compile Error Correction

7 Conclusion

7.1 Evaluation of my work

Firstly, I read coursework question that was provided in our My Second Teacher platform. To start the code, I started writing pseudocode which will be helpful while writing my actual code. I used bluej to complete code of Teacher, Lecturer, and Tutor class with comments including in necessary field and then I drew the class diagram of each class using the online tool i.e. (Draw.io).I check each every line to make sure there is not any error. To make sure there is no error, I tested them in bluej with questions that was provided in My Second Teacher, while checking I found some of the error which is listed above in error section. Later on, I corrected them.

7.2 Reflection of What I have learned.

I wanted to say that this coursework helped me a lot to implement all the concepts that I have learned from classes and online platforms. From this coursework, I am able to implement the concepts like inheritance and encapsulation, different naming conventions like upper camel case and lower camel case where to use which convention. This actually helped me to learn how parent's attributes and methods are inherits to its child classes and how sensitive information are secured with in a class using private accessor modifier. While testing different cases, I learned about different types of error like runtime error, compile time error, logical error and how it can be corrected and also help to know about how methods of parent class are override in its sub class.

7.3 Difficulties and Overcame

Because of the unfamiliarity of new platform like (draw.io, blueJ), I found quite difficult while using it to draw class diagram but later on after spending some time I found it very useful. Also with ms-word, I was unfamiliar but because of the teacher who helped me to deal with problems and also I web surfed to find a solution while developing documentation part. It was difficult to find the exact cause of logical error

and run time error. I consult to my teacher to solve the run time error whereas to solve logical error I scanned each and every of logical part and eventually I found the main cause and solve it Because of the helpful teacher, it became easy to find some of the error whereas I dealt most of them with on my own.

8 References

baeldung, 2024. *baeldung.* [Online] Available at: https://www.baeldung.com/java-history [Accessed 14 1 2024].

Fonseca, L., 2023. *Vengage.* [Online] Available at: https://venngage.com/blog/class-diagram/ [Accessed 14 01 2024].

Mahr, N., 2023. *study.* [Online] Available at: https://study.com/learn/lesson/pseudocode-examples-what-is-pseudocode.html [Accessed 16 01 2024].

Nalimov, C., 2021. *Gleek.* [Online] Available at: https://www.gleek.io/blog/atm-system-class.html [Accessed 14 01 2024].

9 Appendix

Teacher class:

```
/** Teacher class which contains details of teacher
 * @author (Manoj Neupane)
 * @version (1 and date:- 2024/01/12)
 */
public class Teacher
{
    //instance variables
    private int teacherId;
    private int workingHours;
    private String teacherName;
    private String workingType;
    private String employmentStatus;
    private String address;


    /*Constructor of Teacher class which
     * accepts different parameters as below and assign them to their respective instance variable*/
    public Teacher(int teacherId,String teacherName,String address,String workingType,String employmentStatus)
    {
        this.teacherName = teacherName;
        this.address = address;
        this.workingType = workingType;
        this.teacherId = teacherId;
        this.employmentStatus = employmentStatus;


    }
    //accessor of teacherId, It returns integer value stored in teacherId.
```

```
    public int getTeacherId()
    {
        return teacherId;
    }
    //accessor of teacherName, It returns String value stored in teacherName.
    public String getTeacherName()
    {
        return teacherName;
    }
    //accessor of workingHours, It returns integer value stored in workingHours.
    public int getWorkingHour()
    {
        return workingHours;
    }
    //accessor of employmentStatus, It returns String value stored in
employmentStatus.
    public String getEmploymentStatus()
    {
        return employmentStatus;
    }
    //accessor of address, It returns String value stored in address.
    public String getAddress()
    {
        return address;
    }
    //accessor of WorkingType, It returns String value stored in workingType
    public String getWorkingType()
    {
        return workingType;
    }
```

```
    //mutator of workingHours, It accepts a value and assign it to instance variable working hour
    public void setWorkingHour(int workingHour)
    {
        this.workingHours=workingHour;
    }
    //This method display a details of teacher according to different condition as below
    public void display()
    {    //This block of code will be executed if workingHours is empty or if it is negative which shows invalid message to request user to fill workingHours field.
        if(workingHours <= 0) {
            System.out.println("Kindly,fill the working hours.(workingHours is empty or invalid)");
        }
                //This block of code will be executed if workingHours value is greater than 0.
        else {
            System.out.println("teacherName:      "+teacherName+"\n"+"teacherId: "+teacherId);
            System.out.println("Address:                "+address+"\n"+"workingType: "+workingType);
            System.out.println("EmploymentStatus: "+employmentStatus+"\n"+"workingHours: "+workingHours);
        }
    }


}
```

Tutor class:

```
/** Tutor class which contains details of tutor and inherits the protected,default,public property and methods in teacher class with the help of extends keyword

@author (Manoj Neupane)

@version (1 and date:- 2024/01/12)

 */

class Tutor extends Teacher

{

    //instance variables

    private double salary;

    private String specialization;

    private String academicQualification;

    private int performanceIndex;

    private boolean isCertified;

    /*constructor of child class which accepts 10 parameters as below and

     * assign them to respective instance variable along with call to the

     * accessor of workingHours of parent class and assignment of isCertified to false.*/

    public Tutor(int teacherId,String teacherName,String address,String workingType,String employmentStatus,int workingHours,double salary,String specialization,String academicQualification,int performanceIndex)

    {

        //calling parent class constructor with different values as below

        super(teacherId,teacherName,address,workingType,employmentStatus);

        super.setWorkingHour(workingHours);
```

```
        this.salary = salary;
        this.specialization = specialization;
        this.academicQualification = academicQualification;
        this.performanceIndex = performanceIndex;
        isCertified = false;

    }
    //accessor of salary, It returns double type of value stored in salary variable.
    public double getSalary()
    {
        return salary;
    }
    //accessor of specialization, It returns String value stored in specialization instance variable.
    public String getSpecialization()
    {
        return specialization;
    }
    //accessor of academicQualification, It returns String value stored in academicQualification.
    public String getAcademicQualification()
    {
        return academicQualification;
    }
```

```
//accessor of performanceIndex,It returns integer value stored in performanceIndex.
public int getPerformanceIndex()
{
    return performanceIndex;
}
//accessor of isCertified, It returns boolean value stored in isCertified.
public boolean hasIsCertified()
{
    return isCertified;
}
//mutator of salary, It accepts two parameters and calculate the salary on the basis of different condition as below:-
public  void setSalary(double newSalary,int performanceIndex)
{
    /*checking if performanceIndex is greater than 5 and workingHour is greater than 20,
     * This block of code will be executed if peformanceIndex is greater than 5 and workingHour is greater than 20*/
    if(performanceIndex > 5 && super.getWorkingHour() > 20) {
        /* checking if performanceIndex is greater than 5 and performanceIndex is less than equals to 7,
        This block of code will be executed if peformanceIndex is greater than 5 and  performanceIndex is less than equals to 7*/
        if(performanceIndex > 5 && performanceIndex <= 7) {
            this.salary = newSalary + 0.05 * newSalary;
```

```
        }
        /* checking if performanceIndex is greater than 8 and performanceIndex
is less than equals to 9,
        This block of code will be executed if peformanceIndex is greater than
8 and  performanceIndex is less than equals to 9*/
        else if(performanceIndex > 8 && performanceIndex <= 9) {
            this.salary = newSalary + 0.1 * newSalary;
        }
        /* checking if performanceIndex is equals to 10,
        This block of code will be executed if performanceIndex is greater than
or euquals to 10,*/
        else if(performanceIndex == 10) {
            this.salary = newSalary + 0.2 * newSalary;
        }
        //assigning isCertified to true and instance performanceIndex to
performanceIndex local variable.
        isCertified = true;
        this.performanceIndex=performanceIndex;

    }
    //This block of code will be executed if peformanceIndex is less than or
equals to 5 or workingHour is less than or equals to 20.
    else {
        System.out.println("index number is smaller or equals than 5 or
workingHour is smaller or equals than 20");
    }
```

```
    }

    /* This method assigns the different variable like salary to 0,
     * specialization to empty string,academicQualification to emptyString,performanceIndex to 0
     * and isCertified to false if isCertified is false or display invalid message if isCerified is true*/
    public void removeTutor()
    {   //this condition will work if isCertified is false
        if(!isCertified) {
            salary = 0;
            specialization = "";
            academicQualification = "";
            performanceIndex = 0;
            isCertified = false;
        }
        else {
            System.out.println("removing tutor invalid teacher is certified");
        }
    }

    /* display method which is overrided from teacher class ,also which displays the details of Tutor as below condition*/
    @Override
    public void display()
```

```
    {
        //this block of code will work if isCertified is true
        if(isCertified) {
            //call to the display method of Parent class display method with of help of super keyword.
            super.display();
            //Displaying details like salary,specilization,academicQualification,academicQualfication,perfromanceIndex.
            System.out.println("salary: "+salary+"\n"+"specialization: "+specialization);
            System.out.println("academicQualification: "+academicQualification);
            System.out.println("performanceIndex: "+performanceIndex);
        }
        //this block of code will work if isCertified is false
        else {
            //call to the display method of Parent class display method with of help of super keyword.
            super.display();
        }
    }

}
```

Lecturer class:

```
/** Lecturer class which contains details of Lecturer and inherits the default ,protected,public property and methods in teacher class with the help of extends keyword
 * @author (Manoj Neupane)
 * @version (1 and date:- 2024/01/12)
 */
class Lecturer extends Teacher
{
    //instance variabales
    private int yearsOfExperience;
    private int gradedScore;
    private boolean hasGraded;
    private String department;
    /*child class constructor with 7 parameters where
     * call to the constructor of parent class is made with five parameters with
     * the assignment of different variables like(department,yearOfexperience) with
     * respective parameters and hasGraded,gradedScore to false and 0 respectively*/
    Lecturer(int teacherId,String teacherName,String address,String employmentStatus,int yearsOfExperience,String department,String workingType)
    {
```

```
        super(teacherId,teacherName,address,workingType,employmentStatus);
        this.gradedScore = 0;
        hasGraded = false;
        this.yearsOfExperience = yearsOfExperience;
        this.department = department;
    }
    //accessor of Department,It returns String value stored in department.
    public String getDepartment()
    {
        return department;
    }
    //accessor of yearOfExperience,It returns integer value stored in yearOfExperience.
    public int getYearsOfExperience()
    {
        return yearsOfExperience;
    }
    //accessor  of graded score, It returns integer value stored in gradedScore.
    public int getGradedScore()
    {
        return gradedScore;
    }
    //mutator of gradedScore, It accepts an integer value as a parameter and assign it to gradedScore instance variable.
    public void setGradedScore(int value)
```

```
    {
        gradedScore = value;
    }
    //accessor  of hasGraded ,It returns boolean value stored in hasGraded.
    public boolean isHasGraded()
    {
        return hasGraded;
    }

    /*gradeAssignment method which accepts 3 parameters and assigns the graded by checking different condition as below.*/
    public void gradeAssignment(int gradedScore,String department,int yearsOfExperience)
    {
        //this block of code will work if yearOfExperience is greater than or equal to 5
        if(yearsOfExperience >= 5) {
            //this condition will be true if and only if instance department is equal to local department
            if(this.department == department) {
                /*checks whether the given value of gradedScore is less than 0 and greater than 100, if it's true shows invalid message as below.*/
                if(gradedScore<0 && gradedScore>100){
                    System.out.println("gradedScore is invalid");
                }
```

```
        // checks whether the given value of gradedScore is greater than or equals to 70
        else if(gradedScore >= 70) {
            System.out.println("you scored A");
        }
        // checks whether the given value of gradedScore is greater than or equals to 60 and smaller than 70
        else if(gradedScore >= 60 && gradedScore < 70) {
            System.out.println("you scored B");
        }
        // checks whether the given value of gradedScore is greater than or equals to 50 and smaller than 60
        else if(gradedScore >= 50 && gradedScore < 60) {
            System.out.println("you scored C");
        }
        // checks whether the given value of gradedScore is greater than or equals to 40 and smaller than 50
        else if(gradedScore >= 40 && gradedScore < 50) {
            System.out.println("you scored D");
        }
        //this block of code will work if all the above condition are wrong
        else {
            System.out.println("you scored E");
        }
        //assignment of instance variable with changed value while grading score
```

```
            this.gradedScore=gradedScore;
            this.yearsOfExperience=yearsOfExperience;
            hasGraded = true;
        }
        //this block of code will work if teacher is not in same area of department.
        else {
            System.out.println("teacher is not in same area of interest");
        }

    }
    //this block of code will work if teacher has smaller than 5 years of experience
    else {
        System.out.println("teacher has low experience");
    }

}
//this method displays the details of Lecturer if lecturer has assign the grade to students otherwise displays warning message to grade the assignment
@Override
public void display()
{
    //Checking whether the students has graded or not, this block of code will work if hasGraded is true.
    if(hasGraded) {
```

```
            super.display();
            System.out.println("Department:
"+department+"\n"+"yearOfExperience:
"+yearsOfExperience+"\n"+"gradedScore: "+gradedScore);
        }
        // This block of code will execute if hasGraded has value false.
        else {
            System.out.println("grade has not assigned yet.");
        }
    }
}
```