Informatics College Pokhara

Programming

CS4001NP

Coursework 1

Submitted By:

Student Name: Rohit Giri

London Met ID: 23048897

Group: Y1C2

Date: 26-Jan-2024

Submitted To:

Mr. Sushil Paudel

Module Leader

Contents

Table of Figures

1. Introduction

This coursework was delivered to us by our respective teacher Mr. Sushil Paudel which was an individual task need to be done by the respected students. This course work was cut into two sections, one being program consisting of 56 marks and another one being Report containing 44 marks. The aim of this coursework was to implement students with a real-world problem scenario using the Object-oriented concept of Java where in coding part, we had to create three separate classes. Among the three classes, one had a parent class where the other two had to be children class. In report part, we had to show our codes with the help of Class Diagram, Pseudocode, Short Description of what each one of the method does with of course the appropriate Screenshot's and detection of errors faced during the execution.

Overall, with the help of this coursework it gave us the idea of mainly how we should use Microsoft Word properly and the working mechanism of how coding works and how we should present them.

2. Class Diagram

Class diagram is a neat way of visualizing the classes in your system before you actually start coding and is also used in designing and modelling software to describe classes and their relationships. They're a static representation of your system structure. They are composed of three sections: Upper section where it contains the name of the class, Middle section where it contains the attributes of the class and lastly Bottom section where it includes the class methods(geeksforgeeks). Usually, all classes have different access levels depending on the access modifier(visibility) and here are their corresponding symbols:

. Public (+)

. Private (-)

. Protected (#)

. Package (~)

. static (underlined)

Down below we have the class diagrams of the Classes; Teacher, Lecturer and Tutor simultaneously.

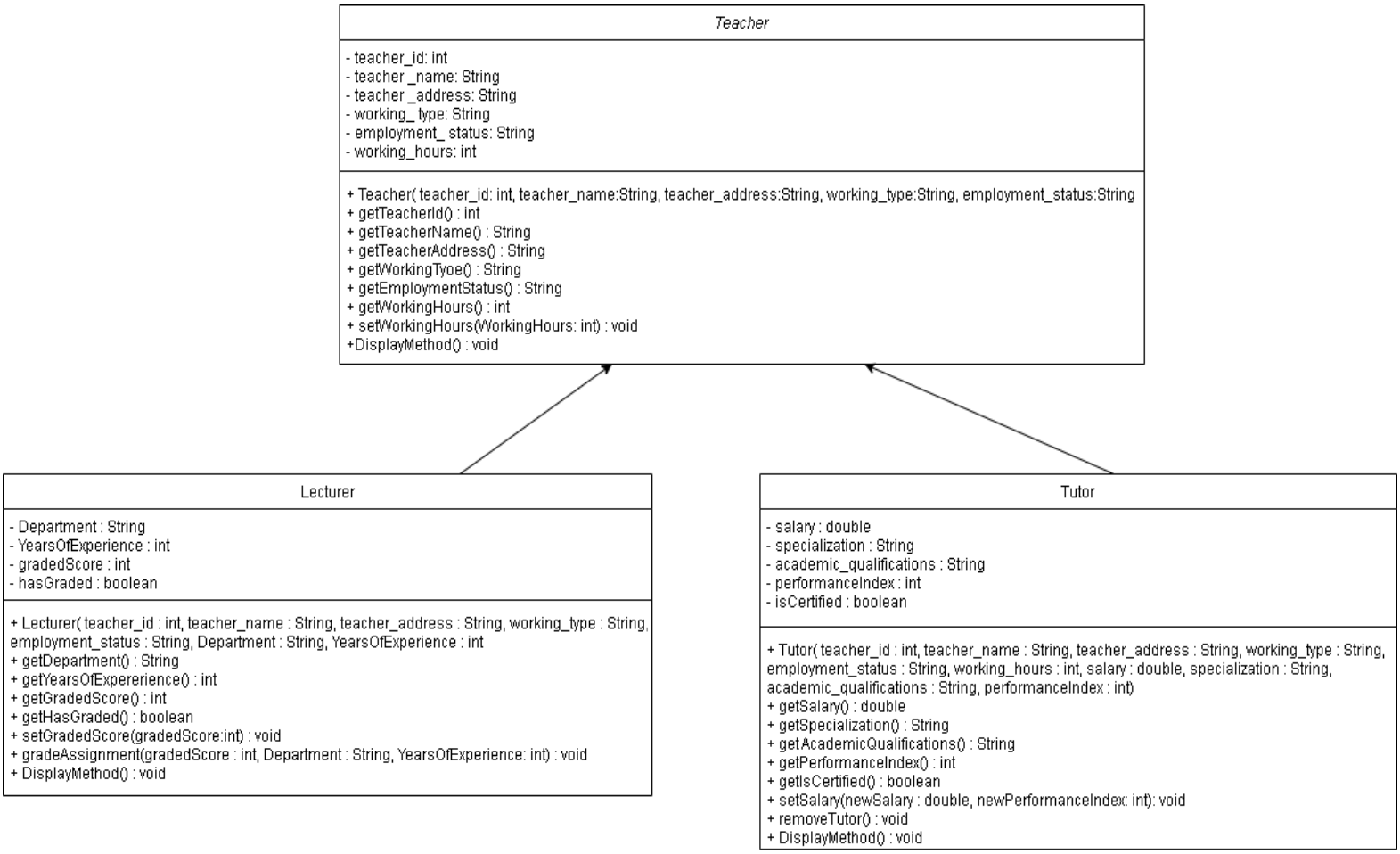


Figure 1: Class Diagram of Teacher, Lecturer and Tutor

2.1 Class Diagram of Teacher Class

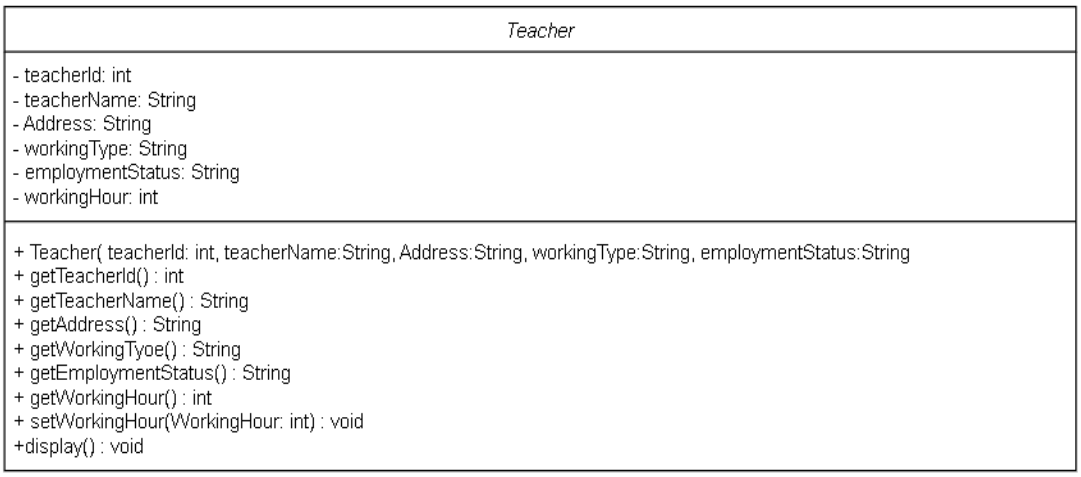


Figure 2: Class Diagram of Teacher Class

2.2 Class Diagram of Lecturer Class

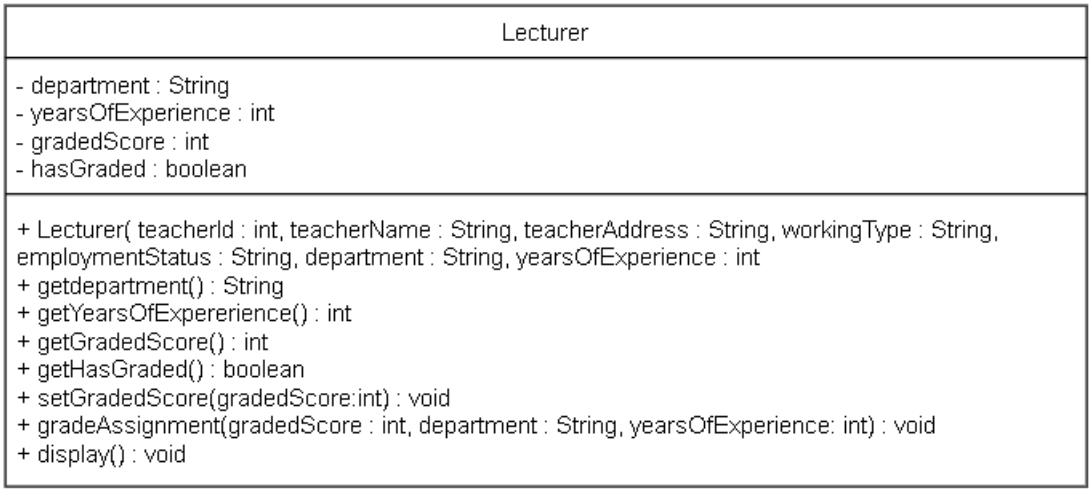


Figure 3: Class Diagram of Lecturer Class

2.3 Class Diagram of Tutor Class

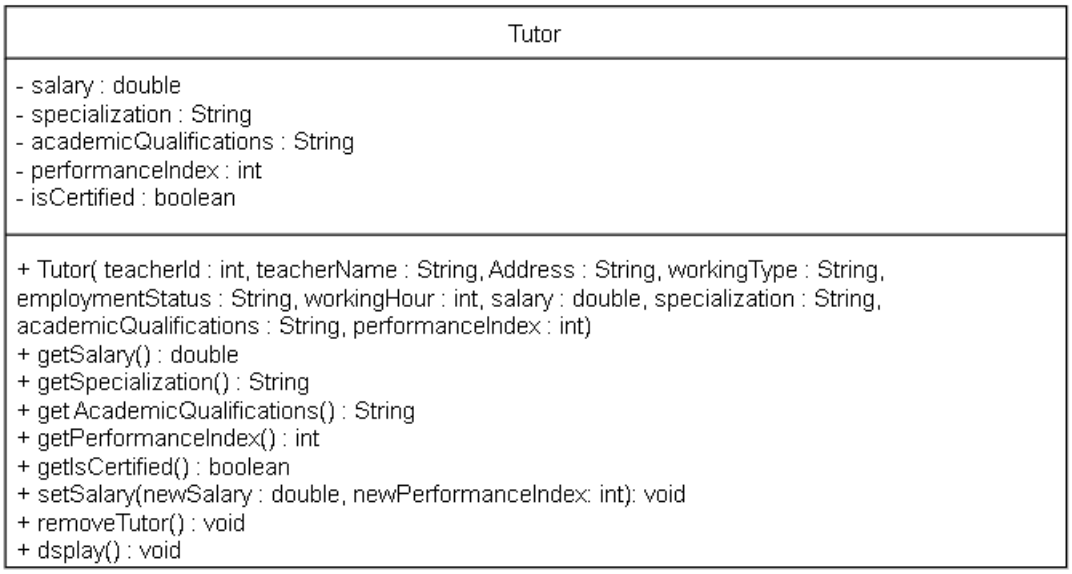


Figure 4: Class Diagram of Tutor Class

3. Pseudocode

Pseudocode is a high-level description of a computer program or algorithm that uses a mix of natural language and programming-like constructs. It uses the structural conventions of a programming language, but is intended for human reading rather than machine reading. Pseudocode typically uses common programming concepts such as variables, loops, conditionals, and functions, but it doesn't adhere to the strict syntax rules of any specific programming language. This makes it a flexible and universal way to express algorithms that can later be translated into actual code in a chosen programming language. No standard for pseudo code syntax exists, so we don't have to follow any strict syntax like computer programming language. Pseudo can vary in different style from one author to another author. It usually borrows its syntax from popular programming languages like C, Python, Pascal, Java and many more.(Tech Target)

Below, here are the Pseudocodes for each of the classes in Java.

3.1 Pseudocode for Teacher Class

```
Teacher Class:
Create Parent Class Teacher
        // private variables
        DECLARE num teacherId
        DECLARE variable teacherName
        DECLARE variable Address
        DECLARE variable workingType
        DECLARE variable employmentStatus
        DECLARE num variable workingHour


CREATE a constructor for class Teacher
        // Constructor
        STORE Such as Teacher (teacherId, teacherName, address,
workingType, employmentStatus)
        Initialize value for teacherId
        Initialize  value for teacherName
        Initialize value for address
        Initialize value for workingType
        Initialize value for employmentStatus
        End Constructor

CREATE Getter method for teacher Class
        // Getter methods
        Initialize by getTeacherId():
                Return value for teacherId
        End Mehod
        Initialize by getTeacherName():
                Return value for teacherName
```

```
        End Method
        Initialize by getAddress():
                Return value for address
        End Method
        Initialize by getWorkingType():
                Return value for workingType
        End Method
        Initialize by getEmploymentStatus():
                Return vale for employmentStatus
        End Method
        Initialize by getWorkingHour():
                Return value for workingHour
        End Method

CREATE setter method
        // Setter method for workingHour
        Initialize by setWorkingHour(newWorkingHour):
                Return value for newWorkingHour
        End Method
CREATE a method  named display
        // display Method
        Create display():

                DETERMINE teacherId
                DETERMINE teacherName
                DETERMINE address
                DETERMINE working_type
                DETERMINE employmentStatus
                IF workingHours is <=0
                        PRINT a suitable message
```

```
            ELSE
                DISPLAY workingHour
            END IF
        END DISPLAY Method
END TEACHER CLASS
```

3.2 Pseudocode for Lecturer

```
CREATE a Child class of Teacher class Lecturer
        // private attributes
        DECLARE  String department
        DECLARE num yearsOfExperience
        DECLARE num gradedScore
        DECLARE Boolean hasGraded\
CREATE a constructor for Lecturer Class
        // Constructor
        DETERMINE constructor of Lecturer by (teacherId, teacherName,
address, workingType, employmentStatus, gradedScore, yearsOfExperience)
                Call parent class constructor
                KEEP num value for yearsOfExperience
                SET value for gradedScore to 0
                SET value for hasGraded to false
        End Constructor
CREATE a getter method
        // Getter methods
        Initialize by getdepartment():
                Return value for department
        End Method
        Initialize by getyearsOfExperience():
                Return value for yearsOfExperience
        End Method
        Initialize by getGradedScore():
                Return value for gradedScore
        End Method
        Initialize by getHasGraded():
                Return value for  hasGraded
        End Method
```

CREATE a Setter Method

```
// Setter method for gradedScore
Initialize by setGradedScore(gradedScore):
        Return value for gradedScore
End Method
```

CREATE method named gradeAssignment

```
//Method to gradeAssignment
    PASS VALUE in gradeAssignment by  (gradedScore, department,
                                       yearsOfExperience)
    If yearsOfExperience >=5 && department is the same corresponding
    value
        IF gradedScore >=70 && gradedScore <=100:
            PRINT A
        Else IF gradedScore >=60 && gradedScore <=70:
            PRINT B
        Else IF gradedScore >=50 && gradedScore <=50:
            PRINT C
        Else IF gradedScore >=40 && gradedScore <=50
            PRINT D
        ELSE
            PRINT E
        END IF
    ELSE
        PRINT suitable message
    END IF
```

CREATE A method named display

```
// display method
Create display():
        Call superclass method named display from Parent Class
```

```
            SHOW department
            SHOW yearsOfExperience
            CREATE an IF method
            IF hasGraded is set to true
                PRINT a suitable message
            ELSE
                PRINT a suitable message
            END IF
        END DISPLAY Method
END LECTURER CLASS
```

3.3 Pseudocode for Tutor Class

```
CREATE another child class named Tutor
    // Private attributes
    DECLARE salary
    DECLARE specialization
    DECLARE academicQualifications
    DECLARE num variable performanceIndex
    DECLARE Boolean variable isCertified

CREATE a constructor for class Tutor
    // Constructor
    PASS value in Tutor by (teacherId, teacherName, address, workingType, employmentStatus, workingHour, salary, specialization, academicQualifications, performanceIndex)
        Call superclass constructor from the parent Class
        //DETERMINE tutor-attributes
        Call setWorkingHour from parent class
        KEEP value for salary
        DECLARE value for gradedScore to 0
        DECLARE value for hasGraded to false
    END constructor

CREATE a Getter Method
    // Getter methods
    Initialize by getSalary():
        Return value for Salary
    End Method
    Initialize by getSpecialization():
        Return value for specialization
    End Method
```

```
Initialize by getAcademicQualifications():
        Return value for academicQualifications
End Method
Initialize by getPerformanceIndex():
        Return value for performanceIndex
End Method
Initialize by getIsCertified():
        Return value for isCertified
End Method
```

CREATE a setter Method

```
//Setter Method
PASS set value in Salary by (newSalary, newPerformanceIndex):
        //For the conditions, check for certification and salary
        // Calculate and set new salary properly
        IF newPerformanceIndex >=5 && newPerformanceIndex <=7:
                DECLARE this.salary to newSalary With addition of
                0.05 and * of newSalary
            ELSE IF newPerformanceIndex >+8 && and
            newPerformanceIndex <=9:
                DECLARE this.salary to newSalary with addition of
                0.1 and * of newSalary
            ELSE IF newPerformanceIndex ==10:
                DECLARE this.salary to newSalary with addition of
                0.2 and * of newSalary
            DECLARE isCertified to true
        ELSE:
            PRINT a suitable message
        END IF
END method
```

```
CREATE a method to named removeTutor
        // Method to remove tutor-details
        CREATE removeTutor():
                //IF method
                IF not isCertified:
                        // It resets tutor-specific attributes
                        DETERMINE salary to 0.0
                        DETERMINE specialization to “”
                        DETERMINE academicQualifications to “”
                        DETERMINE performanceIndex to 0
                        DETERMINE isCertified to false
                ELSE:
                        PRINT a suitable message
                END IF
        END Method

CREATE a method named display
        //display Method
        Create display():
                Call superclass method display from Parent Class
                IF not isCertified:
                        PRINT a suitable message
                ELSE:
                        PRINT a suitable message
                        PRINT a suitable message
                        PRINT a suitable message
                        PRINT a suitable message
                        PRINT a suitable message
                END IF
        END Method
END Tutor Class
```

4. A short description of what each method does(Tutorials point)

4.1. Methods for Teacher Class:

Teacher Class Constructor	Declares an object named 'Teacher' with specific parameters.
TeacherId Method	It returns the teacher's ID
TeacherName Method	It returns the teacher's name
Address Method	It returns the teacher's address
WorkingType Method	It returns the type of work (e.g. Full-Time).
EmploymentStatus Method	It returns the employment status (e.g. active).
WorkingHour Method	It returns the working hours.
WorkingHour Setter Method	It sets a new Value.
display Method	It Displays the information about the teacher Class attributes. And also displays a suitable message.

4.2. Methods for Lecturer Class:

Lecturer Class Constructor	Declares an object named 'Lecturer' with parameters and attributes. It can also call the Parent class.
Department Method	It returns the department of the lecturer class.
YearsOfExperience Method	It returns the years of experience of the lecturer class.
GradedScore Method	It returns the graded Score of the lecturer class.
HasGraded Method	It returns the Boolean whether an assignment has been graded by lecturer.
GradedScore Setter Method	It sets the graded score.
gradeAssignment Method	It grades an assignment based on related attributes. Checks the required years of experience of a lecturer. Prints the grade based on gradedScore.
display Method	It displays all the information about the lecturer class with attributes from the parent class. It also informs whether the assignment has been graded or not graded.

4.3. Method for Tutor Class:

Tutor Class Constructor	It Declares a new object named 'Tutor' with specific parameters and attributes. It also Calls the superclass to set the related attributes.
Salary Method	It returns the salary of the tutor class.
Specialization Method	It returns the specialization of the tutor class.
AcademicQualifications Method	It returns the academic qualifications of the tutor class..
PerformanceIndex Method	It returns the performance index of the tutor class.
IsCertified Method	It returns a Boolean whether the salary of the tutor class is certified or not.
Salary Setter Method	It sets salary and certification based on the new attributes of the corresponding values. Checks if performance index and working hours are greater than 20.
removeTutor Method	It removes the attributes by resetting to default values if not certified. It Prints a suitable message.
display Method	It Displays all the information about the tutor class also the common attribute from the parent class.

5. Testing

5.1. Test 1:

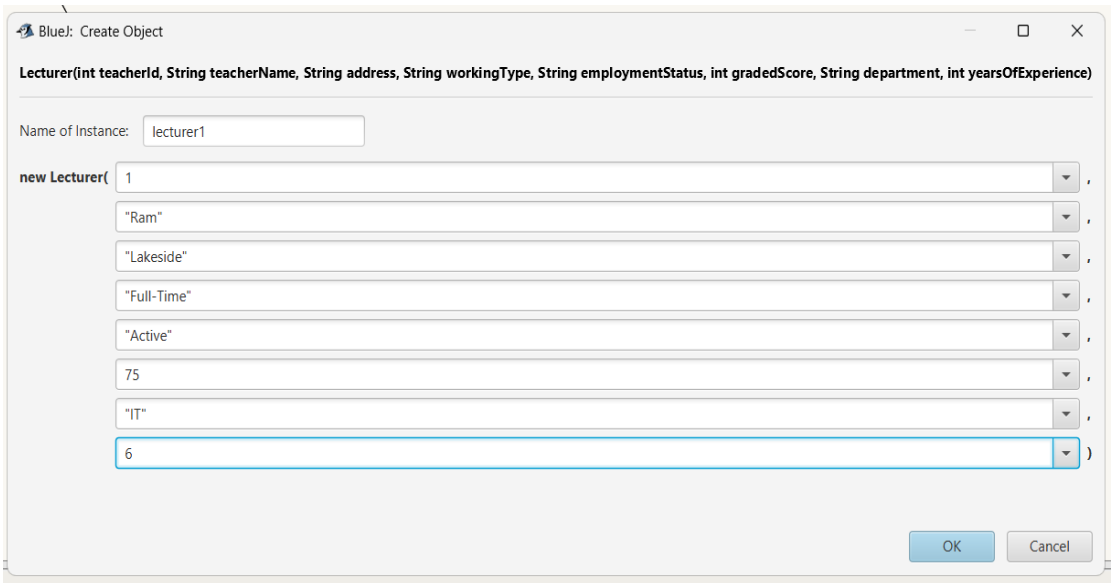


Figure 5: Filling parameters of Lecturer Class

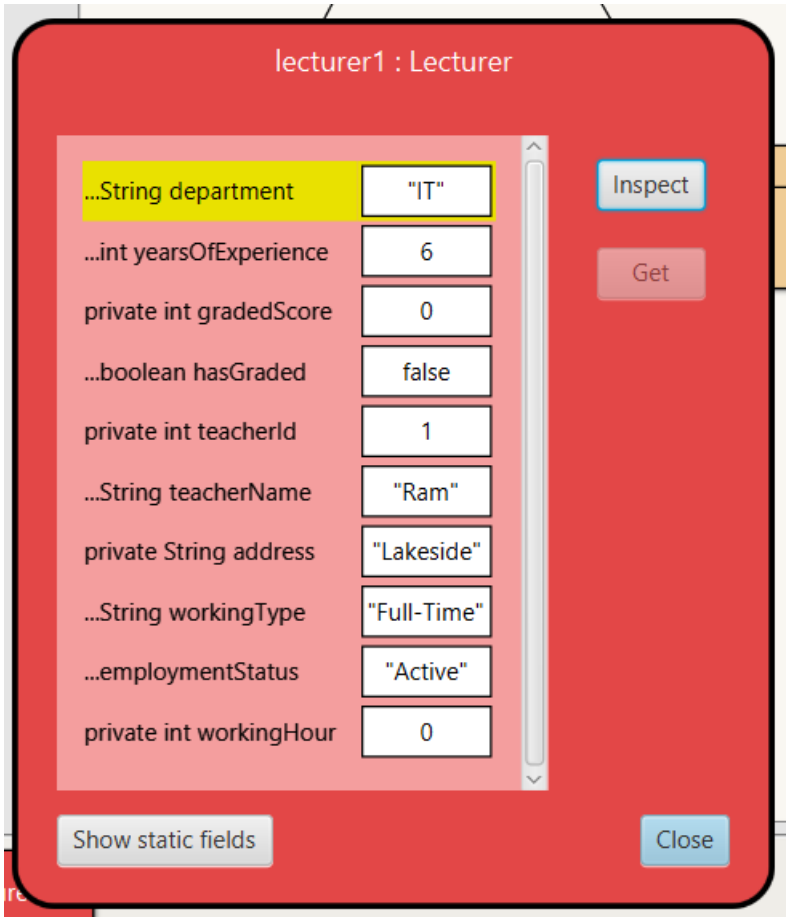


Figure 6: Inspecting the Lecturer Class

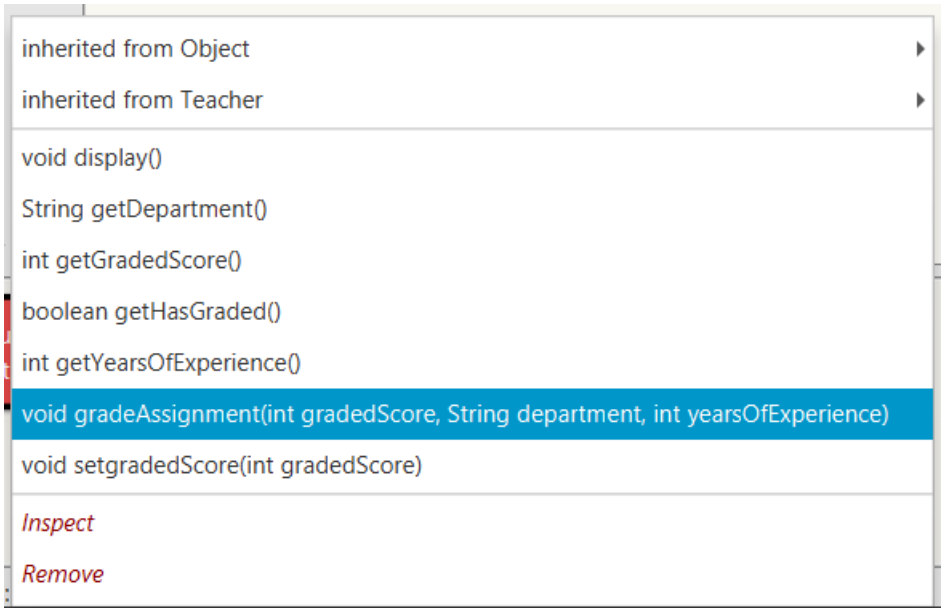


Figure 7: Calling gradeAssignment Method

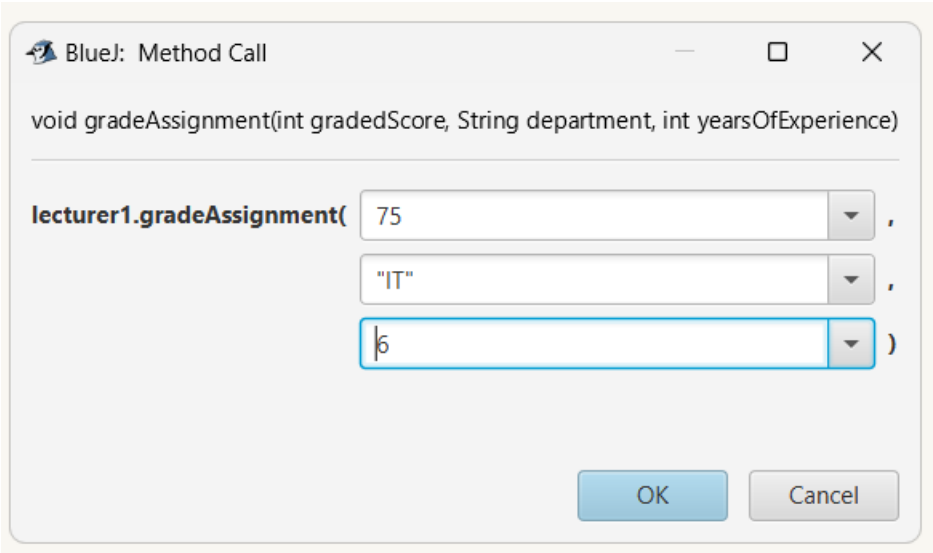


Figure 8: Passing the value in gradeAssignment

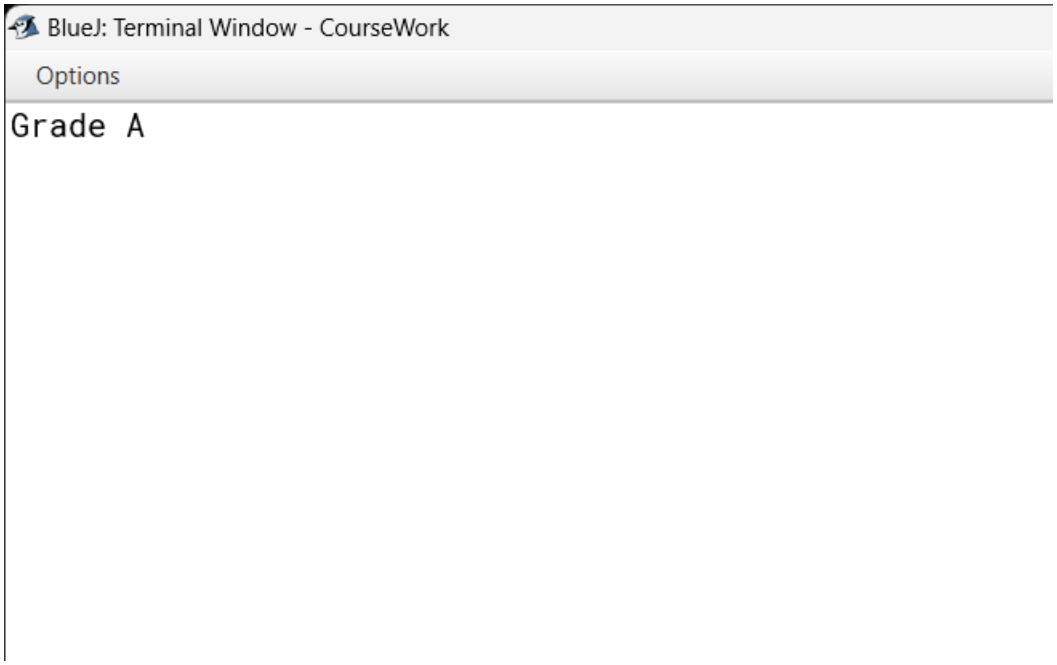


Figure 9: Output is Displayed

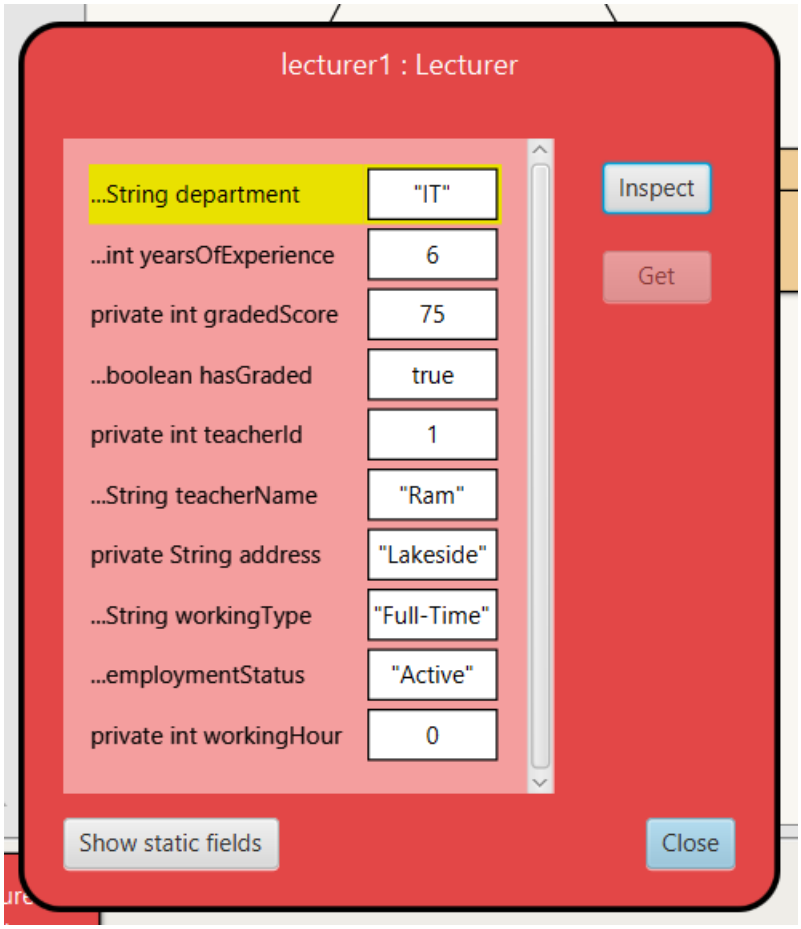


Figure 10: Re-inspecting Lecturer Class

Table 1: Test 1

Objectives	To inspect the Lecturer Class, grade the assignment, and re-inspect the Lecturer Class
Action	Firstly created an object of lecturer and filled all the parameters, Then, inspected the object and called the gradeAssignment and Re-inspected the class.
Expected Outcomes	Grade A should be printed, and then the hasGraded should be true and gradedScore should be set to 75.
Actual Outcome	Grade A is printed, hasGraded is set to true and gradedScore changed to 75.
Conclusion	Test is Successfully executed.

5.2. Test 2:

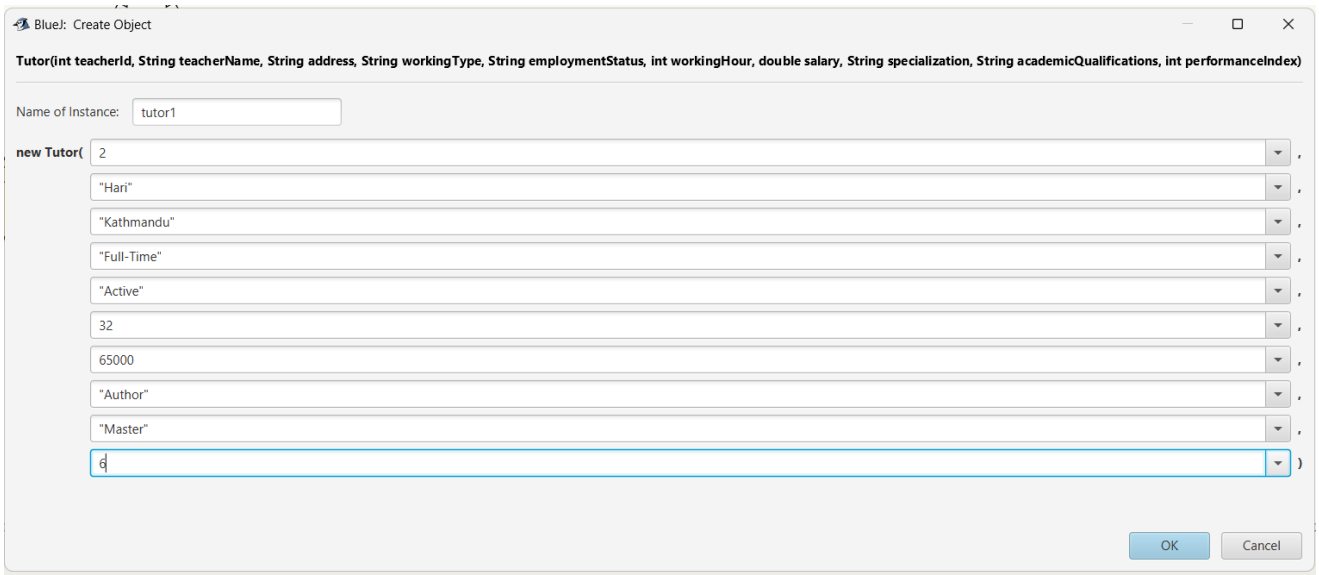
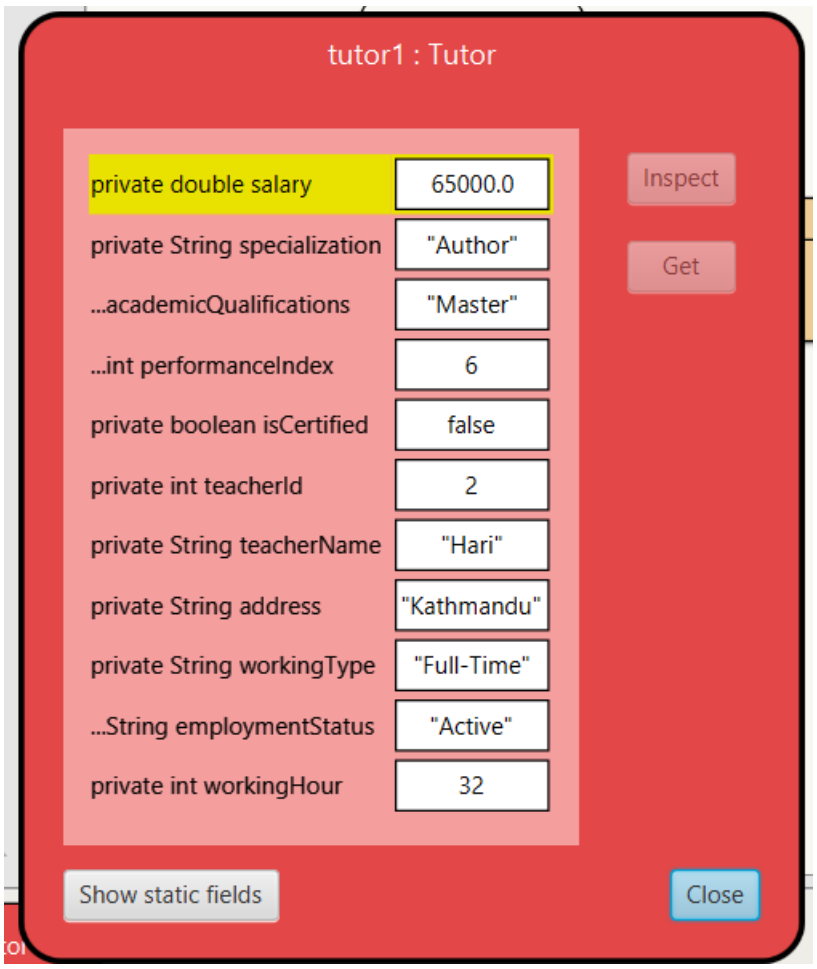


Figure 11: Filling the parameters of Tutor Class



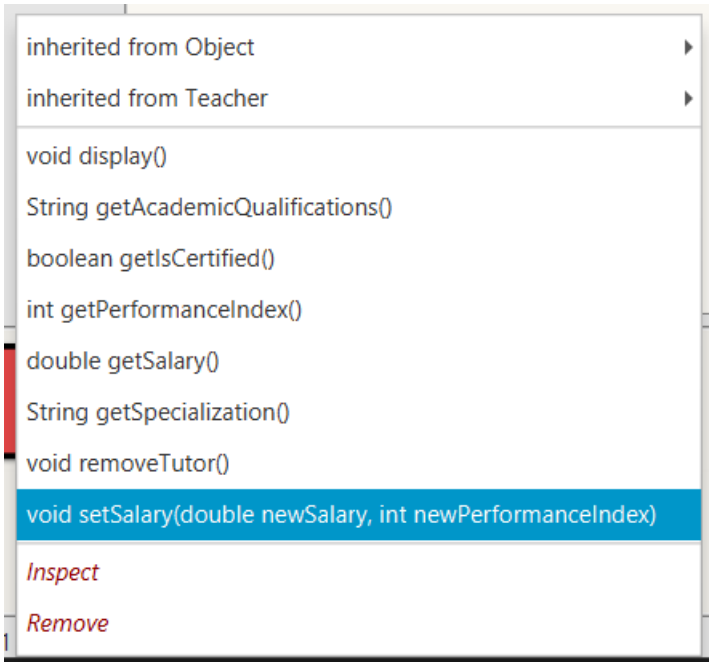


Figure 12: Calling setSalary Method

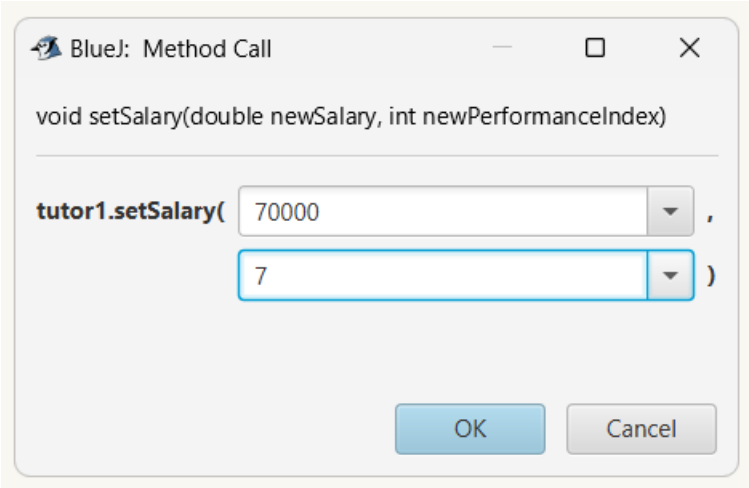


Figure 13: Setting a salary

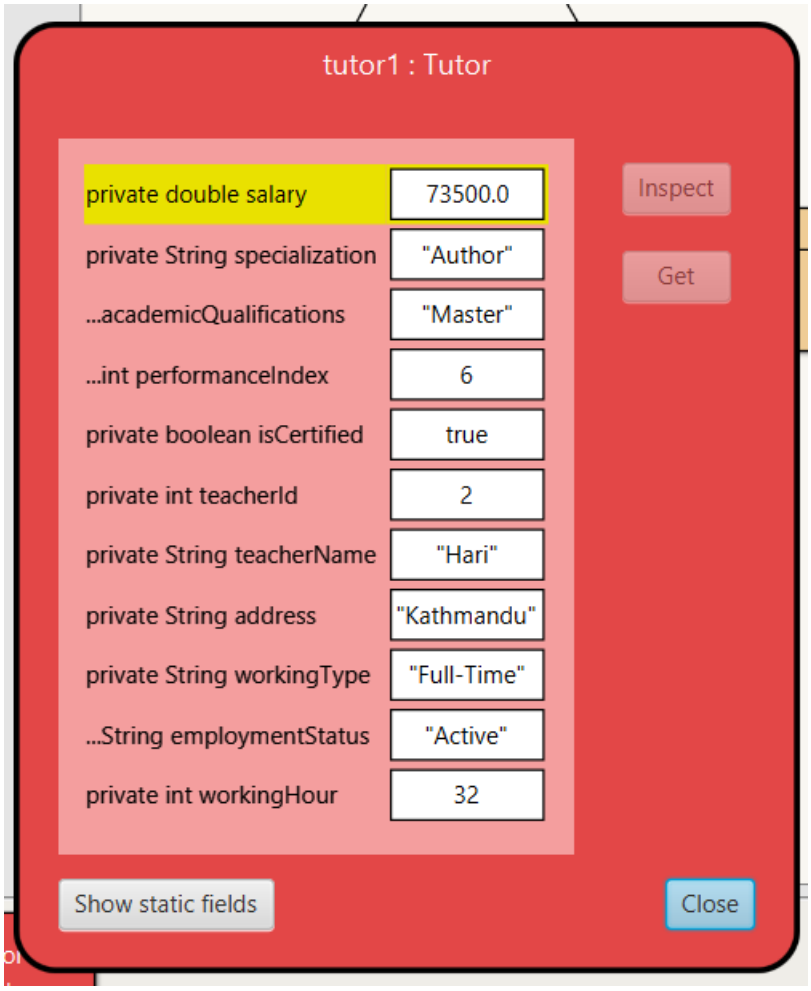


Figure 14: Re-inspecting the Tutor Class

Table 2: Test 2

Objective	To inspect Tutor Class, setSalary and re-inspect the Tutor Class.
Action	Created an object and filled all the required parameters. Then as required, inspected the Tutor object. Then setSalary method newSalary 70000 and newPerformanceIndex(7). Again, re-inspected the Tutor Class
Expected outcome	The Salary should be changed to 73500 and isCertified should be true.
Actual Outcome	Salary set to 73500 and isCertified is set to true.
Conclusion	Test is successfully executed.

5.3. Test 3:

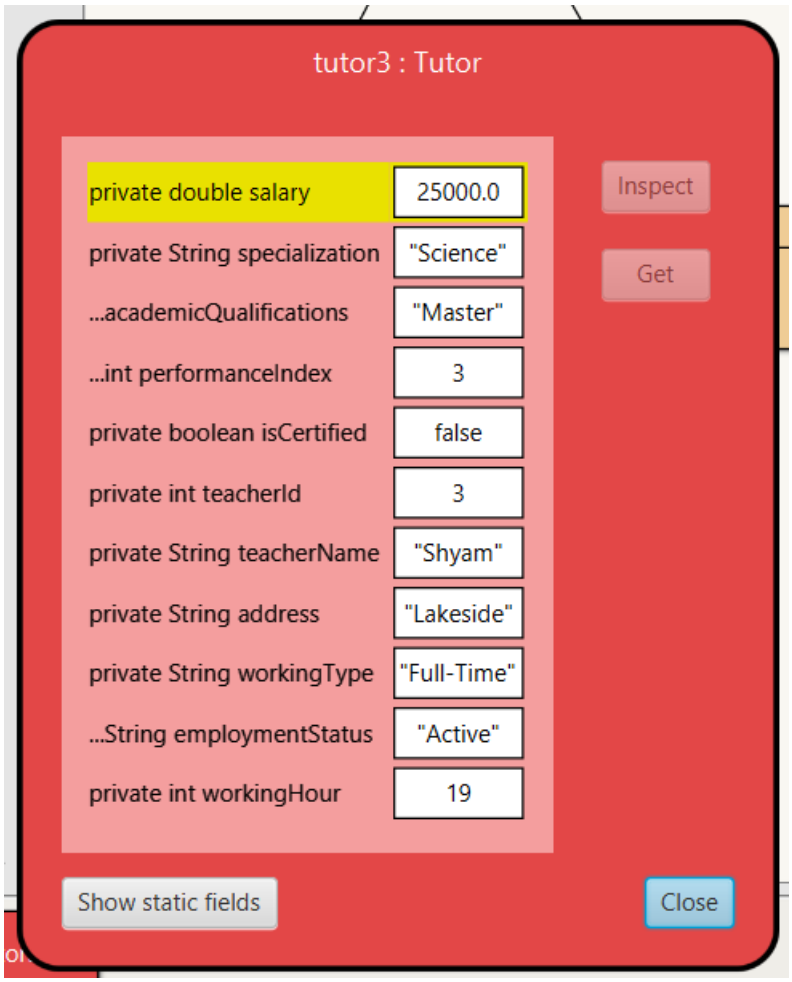


Figure 15: Inspecting after passing parameters on Tutor Class

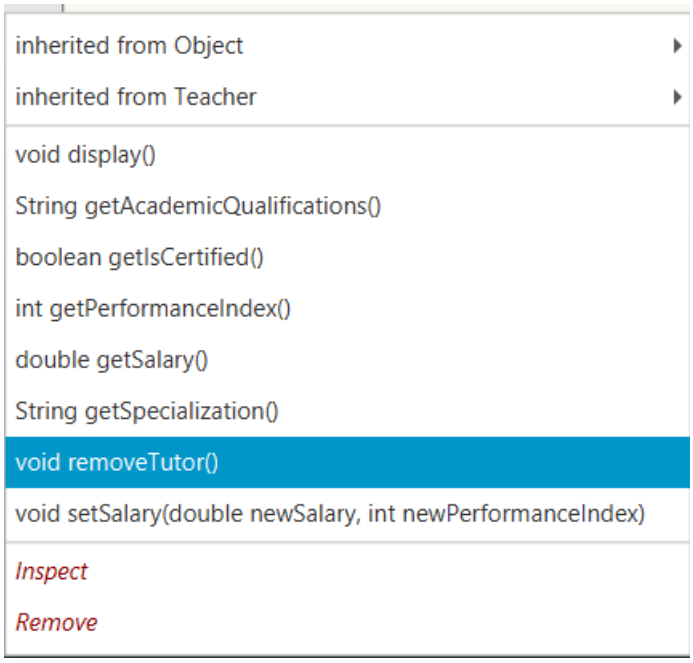


Figure 16: Calling the removeTutor Method

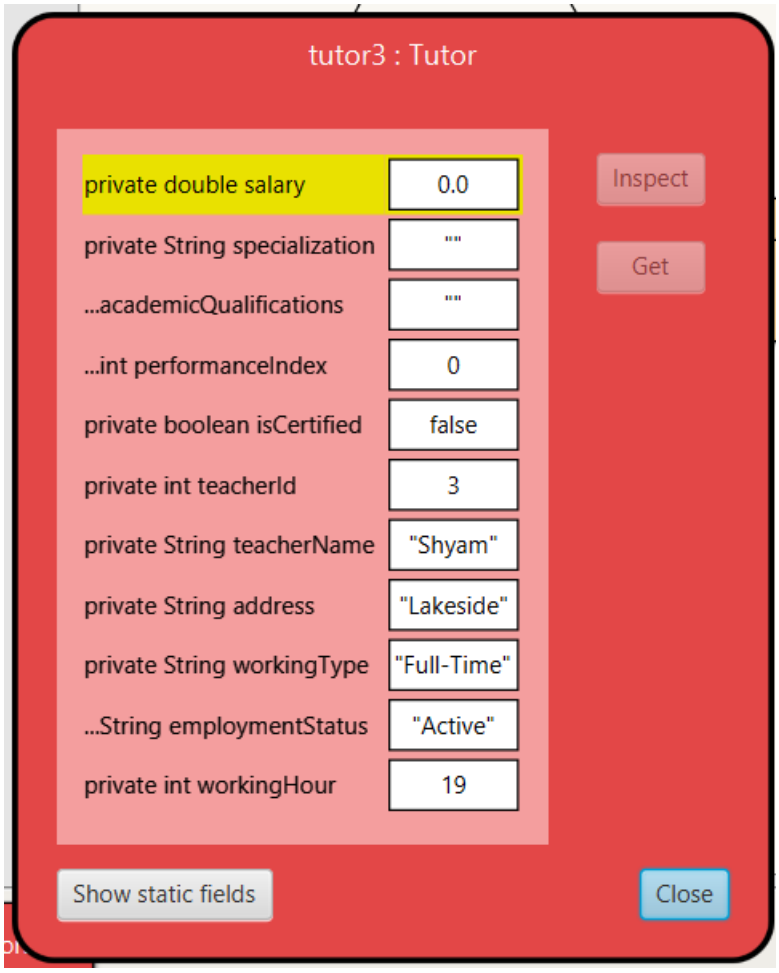


Figure 17: Re-inspecting the Tutor Class

Table 3: Test 3

Objectives	To inspect the Tutor Class again after removing the tutor
Action	We created an object of the class and filled all the required parameters. Then inspected the tutor object. Then we called the removeTutor Method. Then we again re-inspected the tutor Class.
Expected Outcome	Attributes such as; salary, specialization, academicQualifications and performanceIndex should be set to zero and isCertified to false.
Actual Outcome	Attributes such as; salary, specialization, academicQualifications and performanceIndex are set to zero and isCertified to false.
Conclusion	Test is successfully executed.

5.4. Test 4:

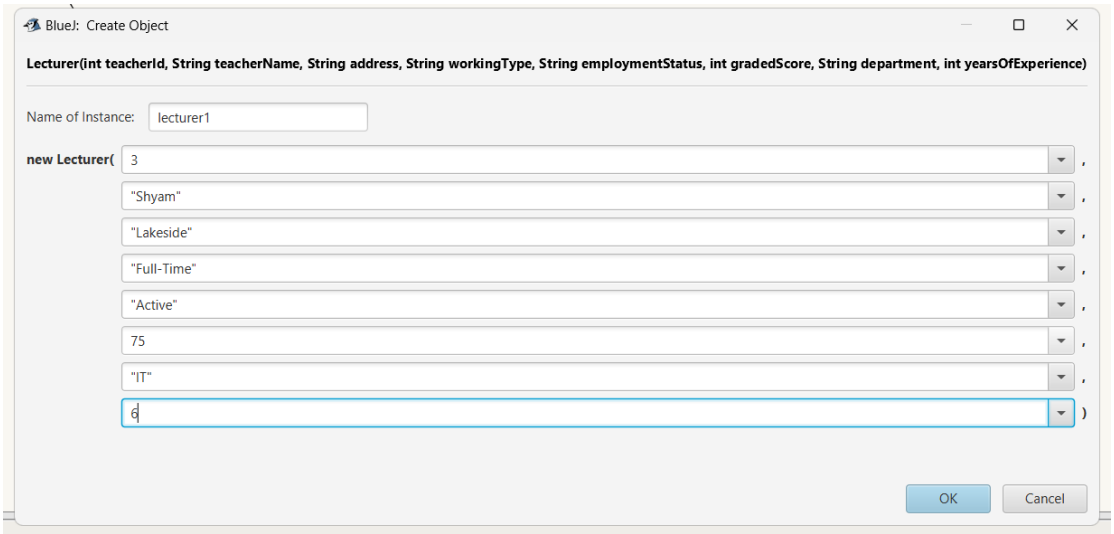


Figure 18: Filling the parameters of Lecturer Class

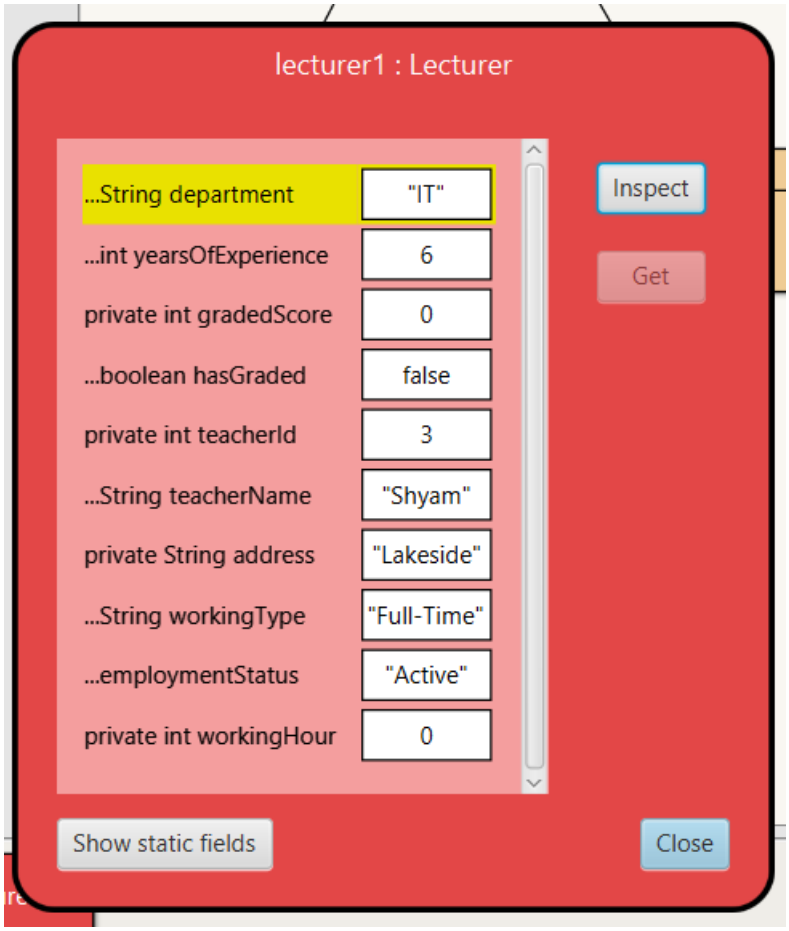


Figure 19: Inspecting the Lecturer Class

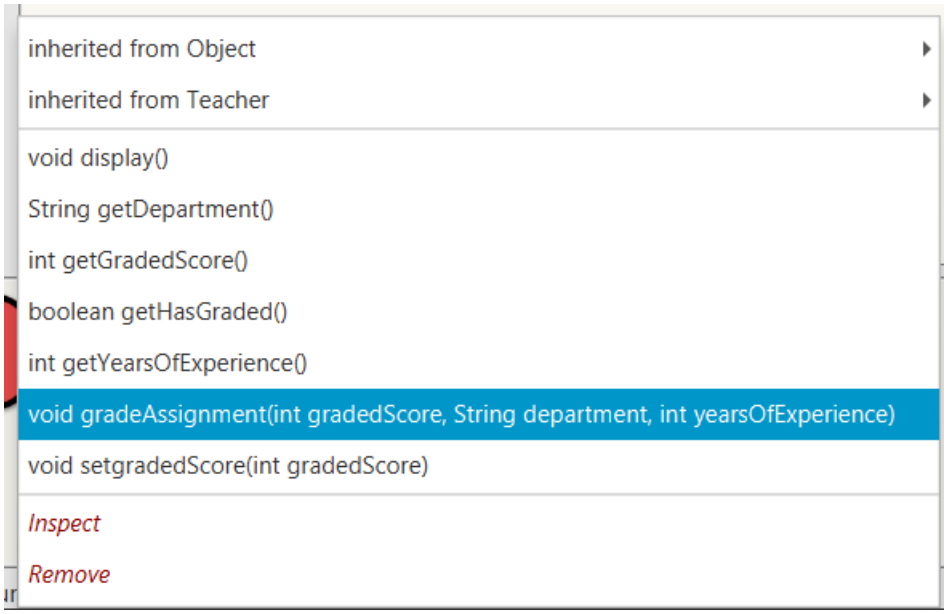


Figure 20: Calling gradeAssignment Method

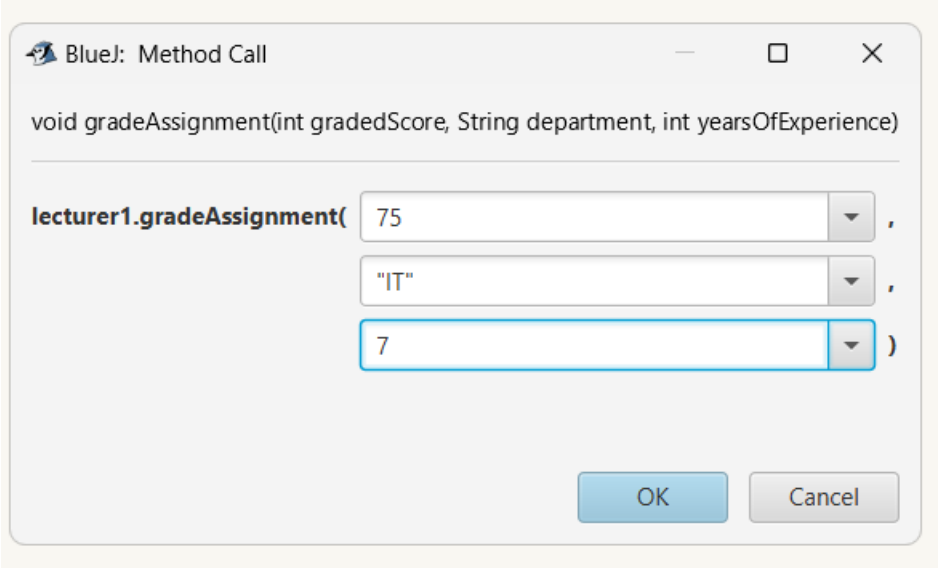


Figure 21: Passing the Value

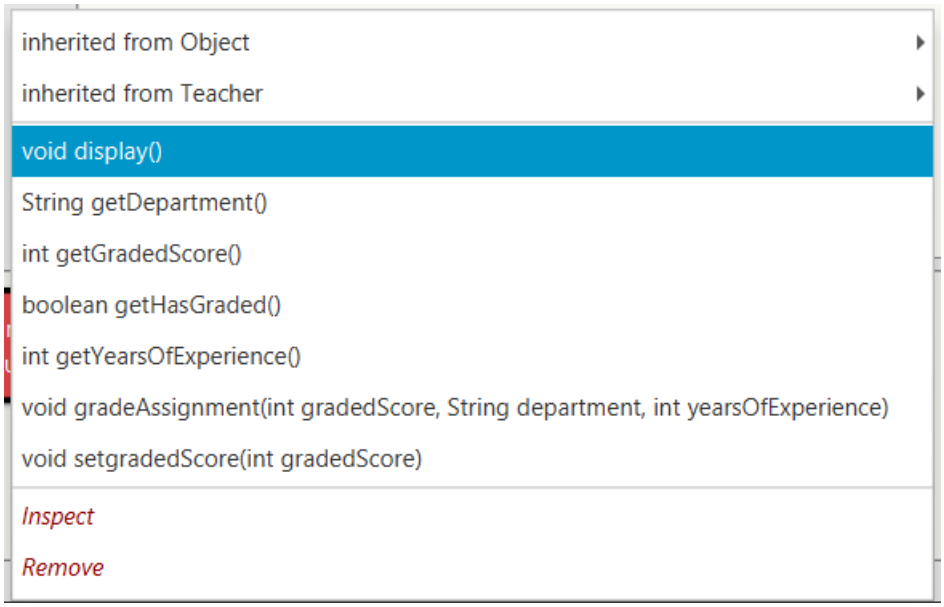


Figure 22: Calling display Method

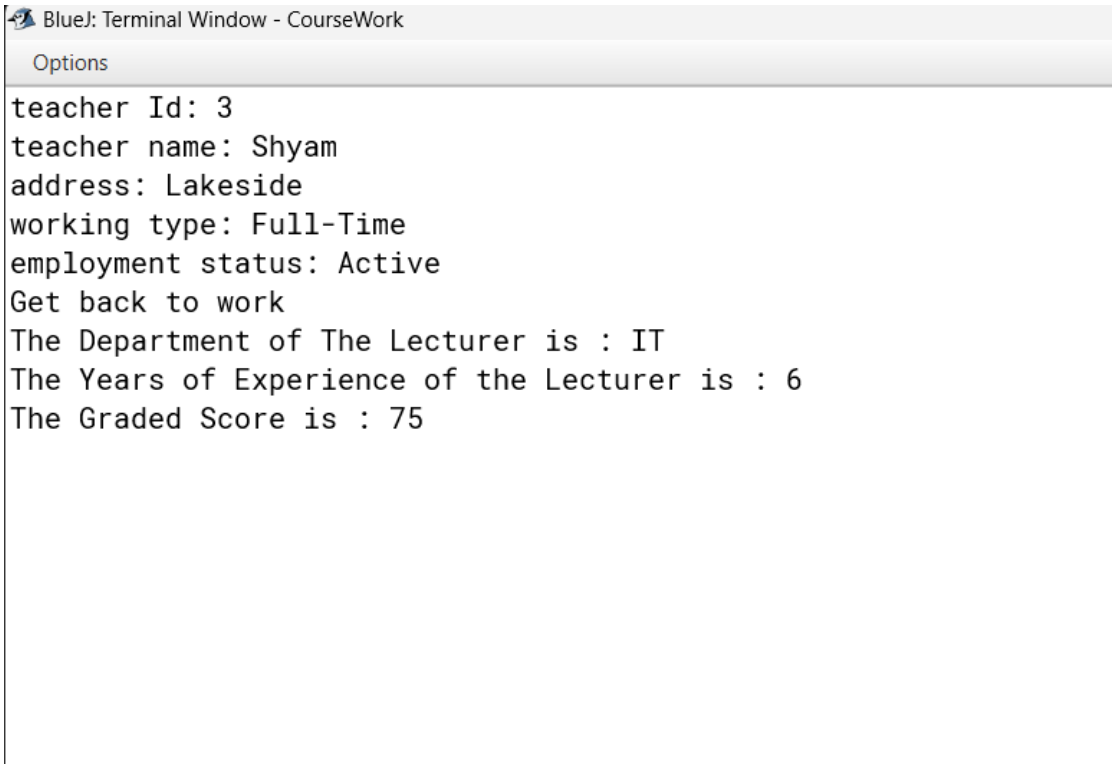


Figure 23: Displayed Output

5.4.1. Table: Lecturer test 4

Objective	To Display all the details in the Lecturer Class
Action	Created an object and filled all the required parameters. Then as required, inspected the Tutor object. Then setSalary method newSalary 70000 and newPerformanceIndex(7). Again, re-inspected the Tutor Class Then, we called the display Method.
Expected Outcome	All the attributes should be Displayed.
Actual Output	All the attributes are Displayed.
Conclusion	Test is Successfully executed.

5.4.2. Test 4: Tutor

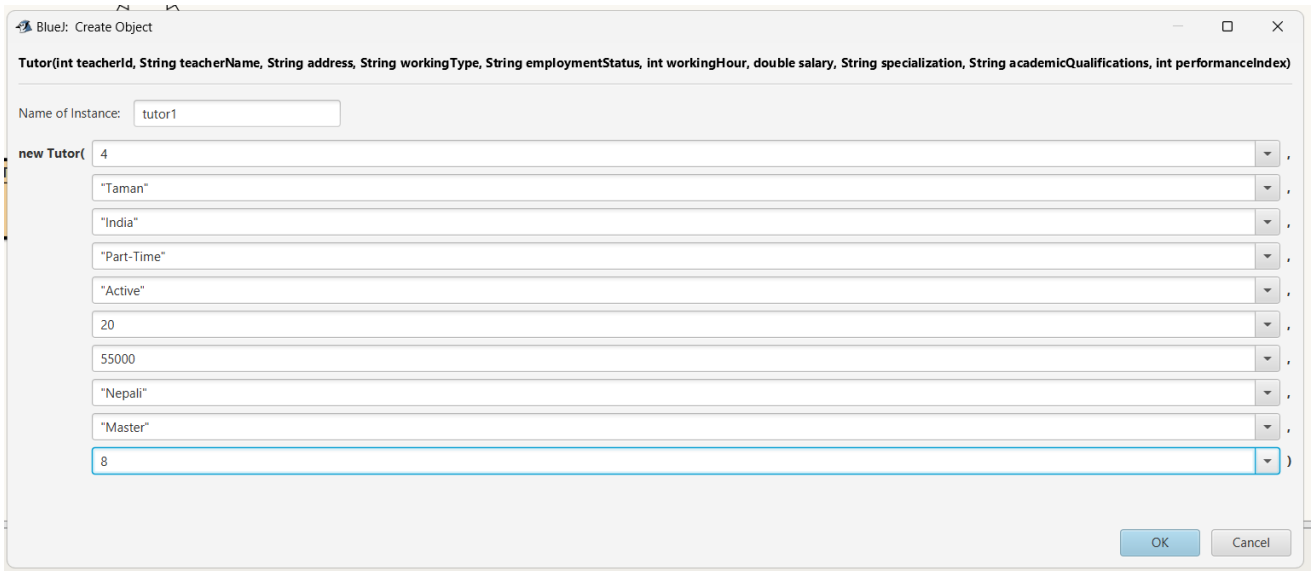


Figure 24: Filling the Parameters of Tutor Class

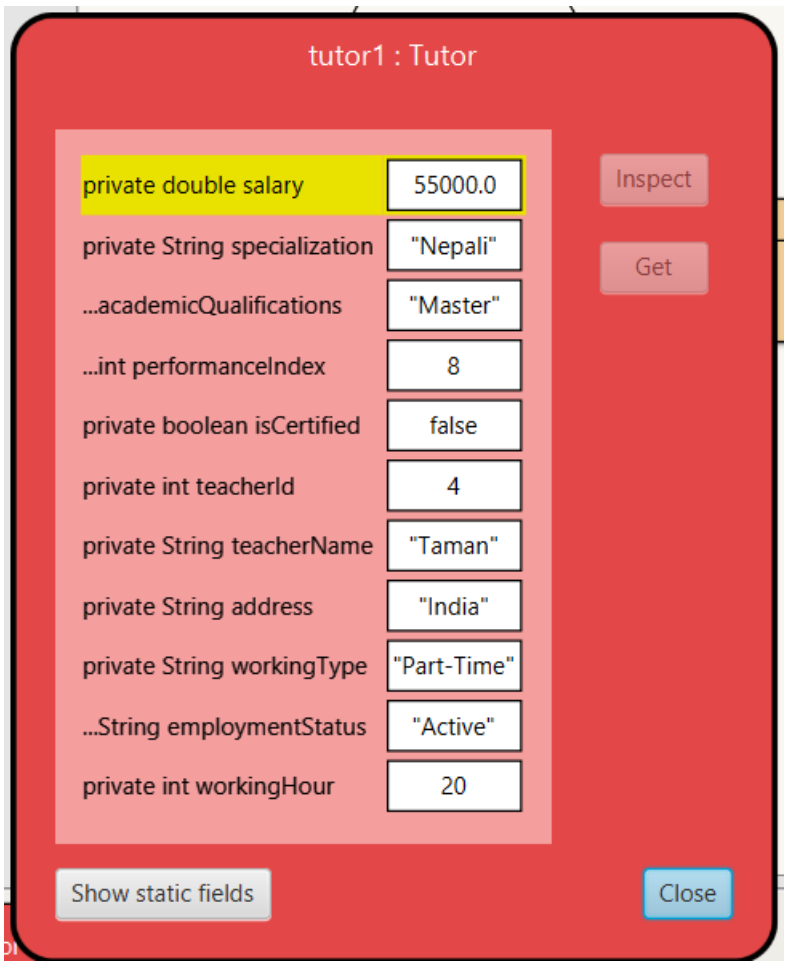


Figure 25: Inspecting the Tutor Class

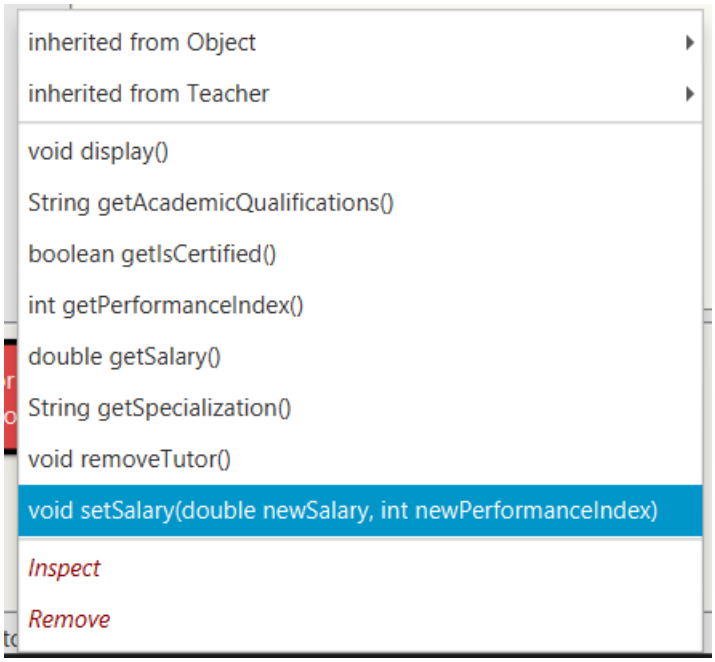


Figure 26: Calling the setSalary Method

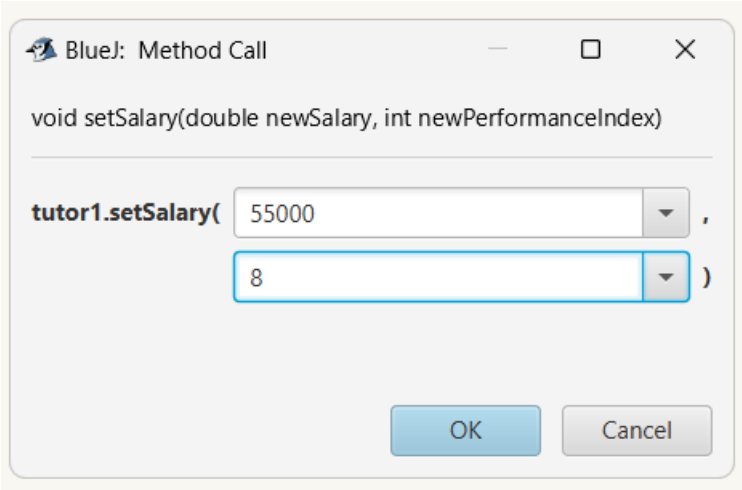


Figure 27: Passing the value

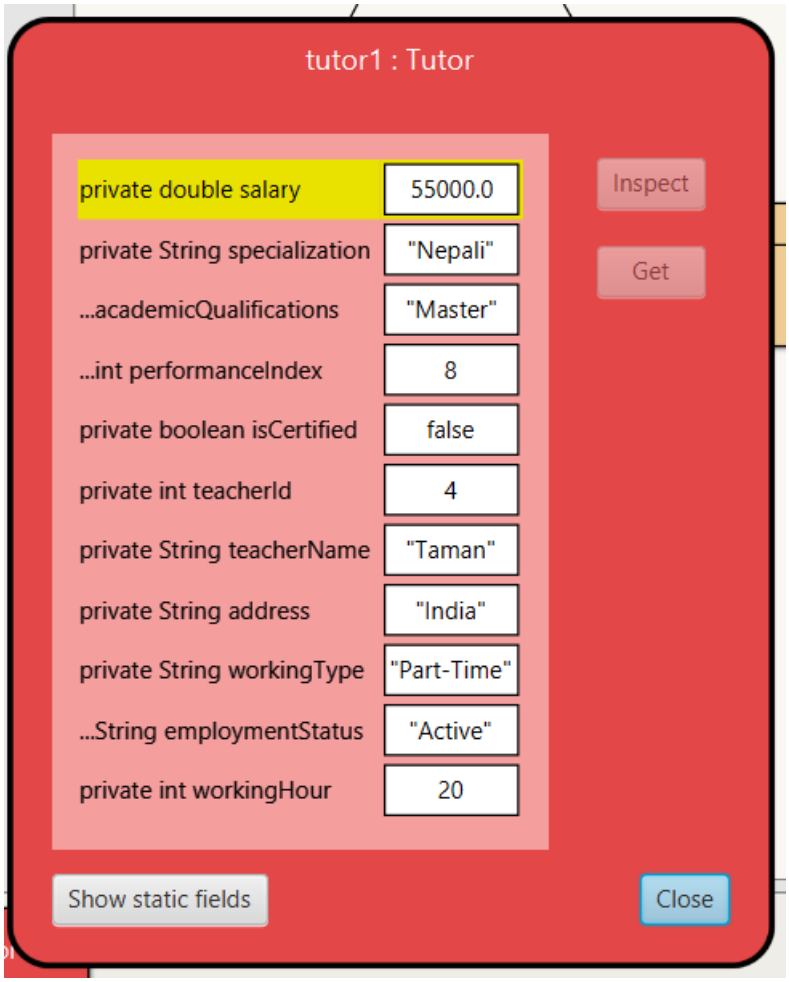


Figure 28: Re-inspecting the Tutor Class

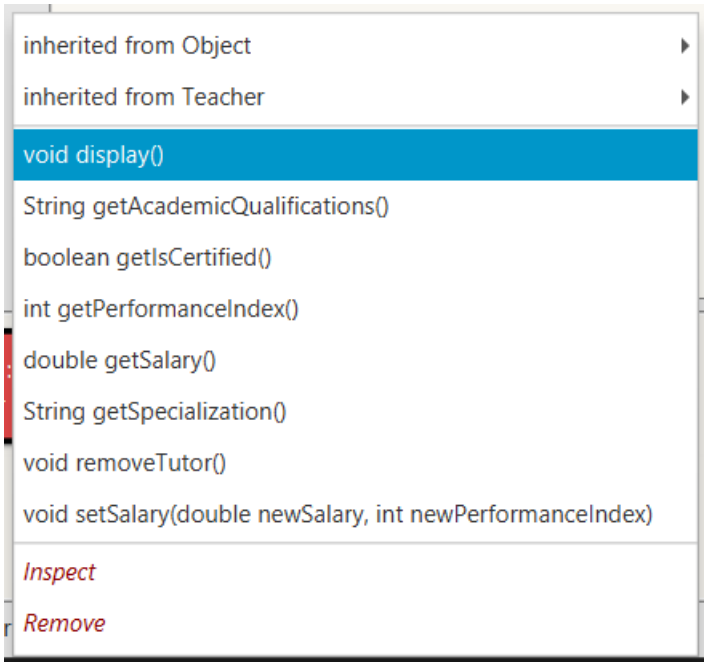


Figure 29: Calling the display Method

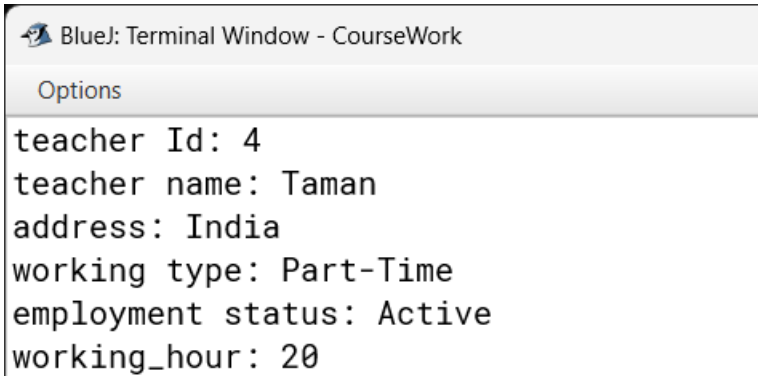


Figure 30: Displayed Output

5.4.3. Table: Test 4

Objective	To Display the details of the Tutor Class.
Action	Created an object and filled all the required parameters. Then as required, inspected the Tutor object. Then setSalary method newSalary 55000 and newPerformanceIndex(8). Then, Re-inspected the class Then we called the display method
Expected Outcome	All the attributes should be displayed.
Actual Outcome	All the attributes displayed.
Conclusion	Test is Successfully executed.

6. ERROR

Error is the subclass of throwable that indicates serious problems that a reasonable application should try not to catch. Generally there are three errors in java.(Code cademy)

6.1. Syntax Error

Syntax Error means the type of error which occurs during compile time .

Expected Cause	Missing semi-colon at the end of the code.
Solution	Adding a semi-colon at the end of the statement.

```
public String getEmploymentStatus(){
    return employmentStatus
}
public int getWorkingHour(){
    return workingHour;
}
```

Figure 31: Expected Syntax error

```
public String getEmploymentStatus(){
    return employmentStatus;
}
public int getWorkingHour(){
    return workingHour;
```

Figure 32: Solution

6.2. Run Time Error

Run time error occurs when a program produces a wrong output.

Expected cause	Missing of super before the display method in tutor class.
Solution	Adding super class before the display method in tutor class.

```
public void display(){
    display();
    System.out.println("The Department of The Lecturer is : " + department)
    System.out.println("The Years of Experience of the Lecturer is : " + ye
    if(hasGraded = true){
        System.out.println("The Graded Score is : " + gradedScore );
    }else {
        System.out.println("Sorry, your assignment hasn't yet been graded "

    }

}
```

Figure 33: Expected Output

```
public void display(){
    super.display();
    System.out.println("The Department of The Lecturer is : " + department)
    System.out.println("The Years of Experience of the Lecturer is : " + ye
    if(hasGraded = true){
        System.out.println("The Graded Score is : " + gradedScore );
    }else {
        System.out.println("Sorry, your assignment hasn't yet been graded "

    }

}
```

Figure 34:Solution

6.3. Logical Error

Logical error is the error found when a program runs to completion without error, but produces wrong output.

Expected output	Incorrect num value 80.
Solution	Corrected num value to 0.1

```
if(newPerformanceIndex >=5 && newPerformanceIndex <=7
    this.salary =(newSalary + 0.05 * newSalary);
}else if(newPerformanceIndex >=8 && newPerformanceInd
    this.salary =(newSalary + 80 * newSalary);
}else if(newPerformanceIndex ==10){
    this.salary =(newSalary + 0.2 * newSalary);
}
this.isCertified = true;
}else{
    System.out.println("Not Possible");
```

Figure 35: Expected output

```
if(newPerformanceIndex >=5 && getWorkingHour() >20){
    if(newPerformanceIndex >=5 && newPerformanceIndex <=7){
        this.salary =(newSalary + 0.05 * newSalary);
    }else if(newPerformanceIndex >=8 && newPerformanceIndex
        this.salary =(newSalary + 0.1 * newSalary);
    }else if(newPerformanceIndex ==10){
        this.salary =(newSalary + 0.2 * newSalary);
    }
    this.isCertified = true;
}else{
        System.out.println("Not Possible");
```

Figure 36: Solution

7. Conclusion

The thing about this whole coursework journey makes me fell that it really helps us students how to think properly, be creative and bring out the best solutions to the problems. I had fun doing all the tasks. The java coding was little bit confusing at first but at the end of the day, it was easy and is all about the mental games. The use of MS Word and all that stuff was fun doing and there is still a lot to learn about the Ms Word, hopefully I will during my freshman years. All the things seem to look hard at first but once you get a hang of it, its like your thing. That is the best way I could describe about my journey to the coursework.

I learned using my brain properly, be creative, how to use Class Diagram and Pseudo coding.

The coding part was fun with all the methods, parent class, child class, the access modifiers and all.

Difficulties I would say, I didn’t face mostly, I just needed to be familiar with my tasks and once I was getting a hang of It I loved doing it. But yeah the one thing I would love to say is the coding part, there was little errors during the testing phase, thanks to my amazing friends guided me there and thankful to them.

This coursework was just a little taste of our journey to being programmers and I would say that procrastinating the coursework is the worst thing to do and I will try my best to overcome that. This coursework not only helped me but also others by getting out of their comfort zone and be social to others more, get to know people exchange knowledge and most importantly have fun doing all the tasks together. There will be a lot of challenging tasks in the future and this coursework made me ready to prepare and welcome all the difficulties.

Also thank you to our subject module teachers, U guys help us push each other for good. Thank You again.

References

Bhumika, 2022. *Geeksforgeeks.* [Online] Available at: https://www.geeksforgeeks.org/unified-modeling-language-uml-class-diagrams/
[Accessed 26 jan 2024].

SHeldon, R., 2021. *techtarget.com.* [Online] Available at: https://www.techtarget.com/whatis/definition/pseudocode [Accessed 26 Jan 2024].

Thompson, K., 2022. *code cademy.* [Online] Available at: https://www.codecademy.com/resources/docs/java/errors [Accessed 26 Jan 2024].

tutorials, 2021. *Tutorial points.* [Online] Available at: https://www.tutorialspoint.com/java/java_methods.htm [Accessed 26 Jan 2024].

8. Appendix

8.1 Teacher Class

```
//question 1
public class Teacher
{
    private int teacherId;
    private String teacherName;
    private String address;
    private String workingType;
    private String employmentStatus;
    private int workingHour;
    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.workingHour=workingHour;
    }
    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 getWorkingHour(){
        return workingHour;
    }
    //setter method
    public void setWorkingHour(int newWorkingHour){
        this.workingHour = newWorkingHour;
    }

    public void display(){
        System.out.println("teacher Id: "+getTeacherId());
        System.out.println("teacher name: "+getTeacherName());
        System.out.println("address: "+getAddress());
        System.out.println("working type: "+getWorkingType());
        System.out.println("employment status: "+getEmploymentStatus());
        if(workingHour > 0){
            System.out.println("working_hour: "+workingHour);
        }else{
            System.out.println("Get back to work");
        }
    }

}
```

8.2 Lecturer Class

```
public class Lecturer extends Teacher{
   private String department;
   private int yearsOfExperience;
   private int gradedScore;
   private boolean hasGraded;

   public Lecturer(int teacherId, String teacherName, String address, String workingType, String employmentStatus, int gradedScore, String department, int yearsOfExperience){
      super(teacherId, teacherName, address, workingType, employmentStatus);
      this.department = department;
      this.yearsOfExperience = yearsOfExperience;
      this.gradedScore = 0;
      this.hasGraded = false;

   }

   public String getDepartment(){
      return department;
   }

   public int getYearsOfExperience(){
      return yearsOfExperience;
   }

   public int getGradedScore(){
      return gradedScore;
   }

   public boolean getHasGraded(){
      return hasGraded;
   }

   public void setgradedScore(int gradedScore){
      this.gradedScore = gradedScore;
```

```
    }

    public void gradeAssignment(int gradedScore, String department, int
yearsOfExperience){

            if(yearsOfExperience >= 5 && this.department== department){
                if(gradedScore >=70 && gradedScore <=100){
                    System.out.println("Grade A");
                }else if(gradedScore >=60 && gradedScore ==69){
                    System.out.println("Grade B");
                }else if(gradedScore >=50 && gradedScore ==59){
                    System.out.println("Grade C");
                }else if(gradedScore >=40 && gradedScore ==49){
                    System.out.println("Graded D");
                }else{
                    System.out.println("Graded E");
                }
                this.hasGraded = true;
                this.gradedScore = gradedScore;
            }else{
                System.out.println("Sorry, The Lecturer hasn't yet graded your
score");
            }
        }


    public void display(){
        super.display();
        System.out.println("The Department of The Lecturer is : " +
department);
        System.out.println("The Years of Experience of the Lecturer is : " +
yearsOfExperience);
        if(hasGraded = true){
            System.out.println("The Graded Score is : " + gradedScore );
        }else {
            System.out.println("Sorry, your assignment hasn't yet been
graded ");

        }
```

```
    }

    }
```

8.3. Tutor Class

```
/**
 * Write a description of class Tutor here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Tutor extends Teacher
{
    private double salary;
    private String specialization;
    private String academicQualifications;
    private int performanceIndex;
    private boolean isCertified;
    public Tutor(int teacherId,String teacherName,String address,String workingType,String employmentStatus,int workingHour,double salary,String specialization,String academicQualifications,int performanceIndex){

super(teacherId,teacherName,address,workingType,employmentStatus);
        super.setWorkingHour(workingHour);
        this.salary= salary;
        this.specialization = specialization;
        this.academicQualifications= academicQualifications;
        this.performanceIndex = performanceIndex;
        this.isCertified = false;
    }
    public double getSalary(){
        return salary;
    }
    public String getSpecialization(){
        return specialization;
    }
    public String getAcademicQualifications(){
        return academicQualifications;
    }
```

```
    public int getPerformanceIndex(){
        return performanceIndex;
    }
    public boolean getIsCertified(){
        return isCertified;
    }
    public void setSalary(double newSalary,int newPerformanceIndex){
            if(newPerformanceIndex >=5 && getWorkingHour() >20){
                if(newPerformanceIndex >=5 && newPerformanceIndex <=7){
                    this.salary =(newSalary + 0.05 * newSalary);
                }else if(newPerformanceIndex >=8 && newPerformanceIndex
<=9){
                    this.salary =(newSalary + 0.1 * newSalary);
                }else if(newPerformanceIndex ==10){
                    this.salary =(newSalary + 0.2 * newSalary);
                }
                this.isCertified = true;
            }else{
                    System.out.println("Not Possible");
                }
            }


    public void removeTutor(){
        if(!isCertified){
            this.salary = 0.0;
            this.specialization ="";
            this.academicQualifications="";
            this.performanceIndex=0;
            this.isCertified=false;
        }else{
            System.out.println("Certified tutor cannot be removed");
        }
    }
    public void display(){
        super.display();
        if(!isCertified){
            System.out.println("Sorry not certified");
```

```
        }else{
            System.out.println("Tutor Details:");
            System.out.println("Salary: "+salary);
            System.out.println("Specialization: "+specialization);
            System.out.println("Academic Qualifications: "+academicQualifications);
            System.out.println("Performance Index: "+performanceIndex);
        }
    }
}
```