CS4001NI Programming

30% Individual Coursework

2023-24 Autumn

Student Name: Shivam Kumar Thakur

London Met ID: 23050396

College ID: NP01CP4A230301

Group: C11

Assignment Due Date: Friday, January 26, 2024

Assignment Submission Date: Thursday, January 25, 2024

Table of Contents

Table Of Figure

Table Of Tables

Introduction

Java

Java is a class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is intended to let application developers write once, and run anywhere (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java was first released in 1995 and is widely used for developing applications for desktop, web, and mobile devices. Java is known for its simplicity, robustness, and security features, making it a popular choice for enterprise-level applications. (GeeksforGeeks, 2022)

JAVA was developed by James Gosling at Sun Microsystems_Inc in the year 1995 and later acquired by Oracle Corporation. It is a simple programming language. Java makes writing, compiling, and debugging programming easy. It helps to create reusable code and modular programs. (GeeksforGeeks, 2022)

About This Project :

The main aim of this project was to get us familiar with Object Oriented Programming System (OOPs) and apply object-oriented principles of java to design and implement a realistic problem scenario of an educational institution to manage the record of teachers, lecturers and tutor in a college system. We are required to create three classes Teacher, Lecturer and Tutor and each with their own attributes, constructors, and method that reflects their roles and function. With the Java Concept that we learned till now in college and the instructions given in the coursework We are to enhance our java program and make it more flexible and reusable.

Tools used for the project:

BLUE J

BlueJ is a windows based platform for Java Development Kit (JDK). It is a free Java environment started in 1999 by Michael Kolling and John Rosenberg at Monash University, Australia, as a successor to Blue. It requires to install JDK version 1.3 or more before installing BlueJ. It can be freely downloaded from its official website of BlueJ. It was developed to support learning and teaching of OOPs(object-oriented programming). The objects can be interactively created and tested. It is a simple user interface, BlueJ has a simpler interface than most professional IDEs. It offers many tools that are specific to its educational goals. There are also standard development tools available, such as an editor, compiler and runtime environment. (GeeksforGeeks, 2023)

Draw.io

Draw.io is a free, web-based diagramming tool that supports various diagram types, including UML. It integrates with various cloud storage services and can be used offline. (GeeksforGeeks, n.d.)draw.io is proprietary software for making diagrams and charts. The software lets you choose from an automatic layout function, or create a custom layout. They have a large selection of shapes and hundreds of visual elements to make your diagram or chart one-of-a-kind. (Computer Hope, 2020)

Microsoft Word

Microsoft word is a word processor software developed by Microsoft in 1983. It is the most commonly used word processor software. It is used to create professional quality documents, letters, reports, resumes, etc and also allows you to edit or modify your new or existing document. The file saved in Ms Word has .docx extension. (GeeksforGeeks, 2021)

Class Diagram

Class diagrams are a type of UML (Unified Modeling Language) diagram used in software engineering to visually represent the structure and relationships of classes in a system. UML is a standardized modeling language that helps in designing and documenting software systems. They are an integral part of the software development process, helping in both the design and documentation phases (GeeksforGeeks, 2024)

Teacher Class:

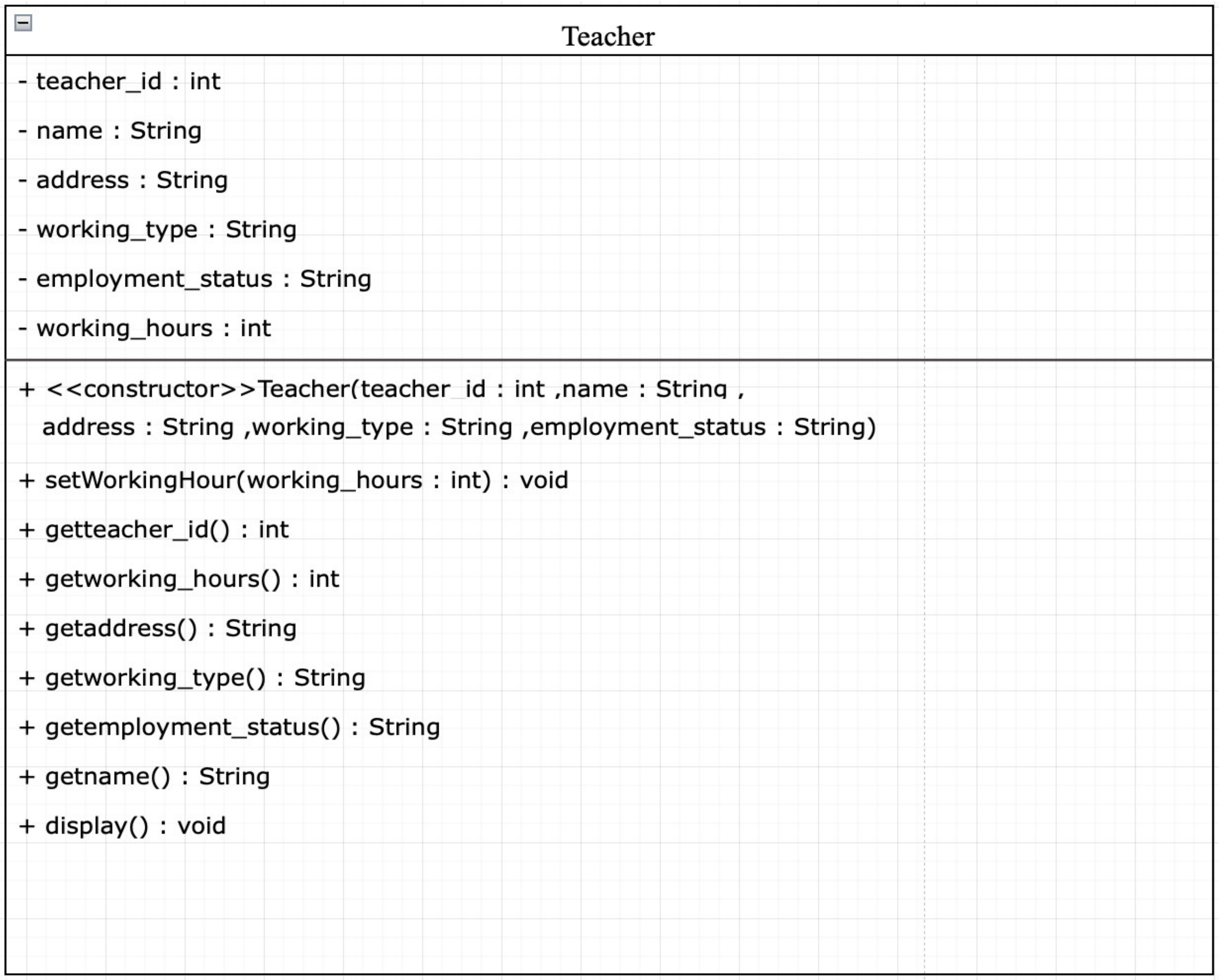

Teacher
- teacher_id : int
- name : String
- address : String
- working_type : String
- employment_status : String
- working_hours : int
+ <<constructor>>Teacher(teacher_id : int ,name : String , address : String ,working_type : String ,employment_status : String)
+ setWorkingHour(working_hours : int) : void
+ getteacher_id() : int
+ getworking_hours() : int
+ getaddress() : String
+ getworking_type() : String
+ getemployment_status() : String
+ getname() : String
+ display() : void

Figure 1:Class Diagram :Teacher Class

Lecturer Class :

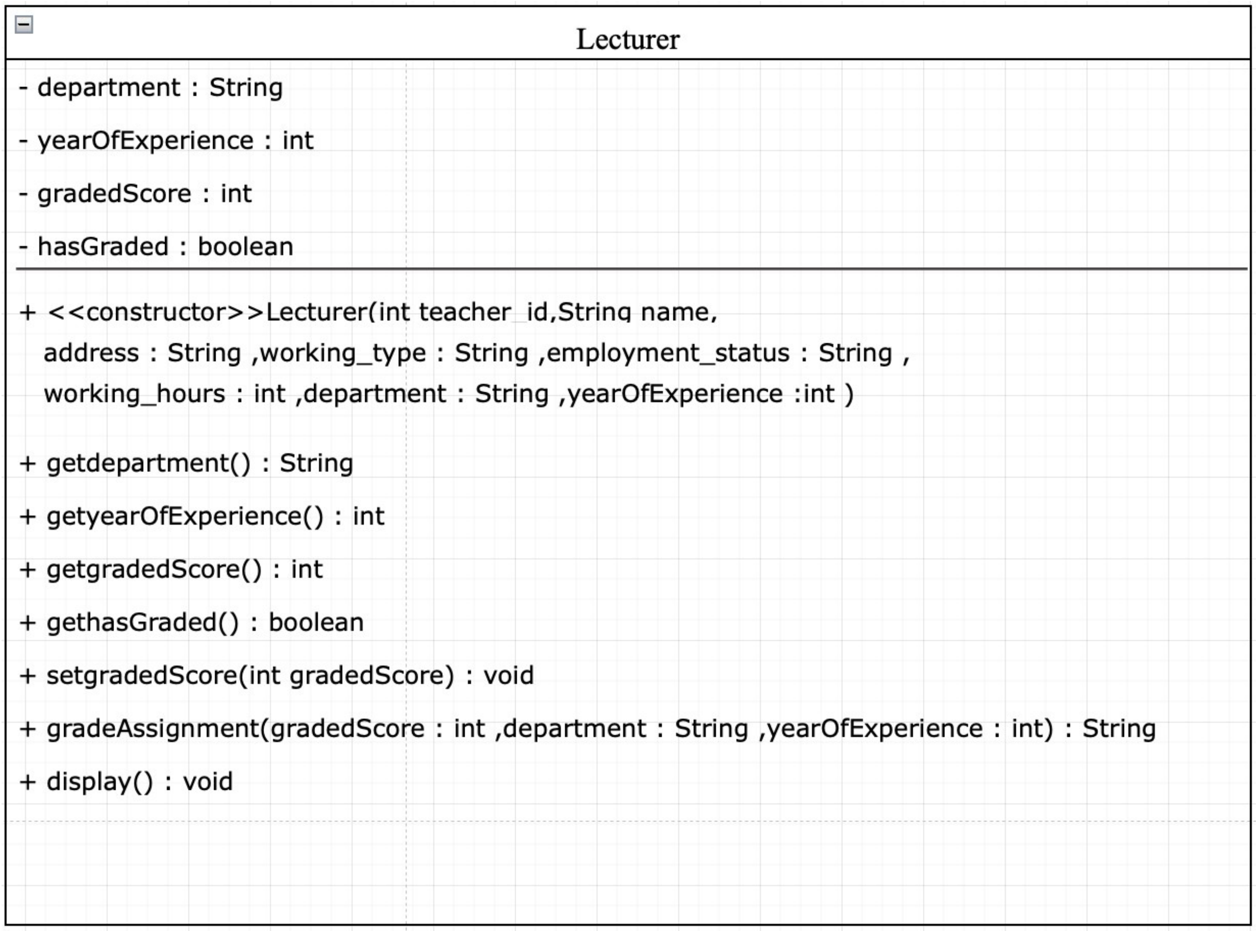

Lecturer
- department : String
- yearOfExperience : int
- gradedScore : int
- hasGraded : boolean
+ <<constructor>>Lecturer(int teacher_id,String name,
address : String ,working_type : String ,employment_status : String ,
working_hours : int ,department : String ,yearOfExperience :int)
+ getdepartment() : String
+ getyearOfExperience() : int
+ getgradedScore() : int
+ gethasGraded() : boolean
+ setgradedScore(int gradedScore) : void
+ gradeAssignment(gradedScore : int ,department : String ,yearOfExperience : int) : String
+ display() : void

Figure 2:Class Diagram :Lecturer Class

Tutor Class:

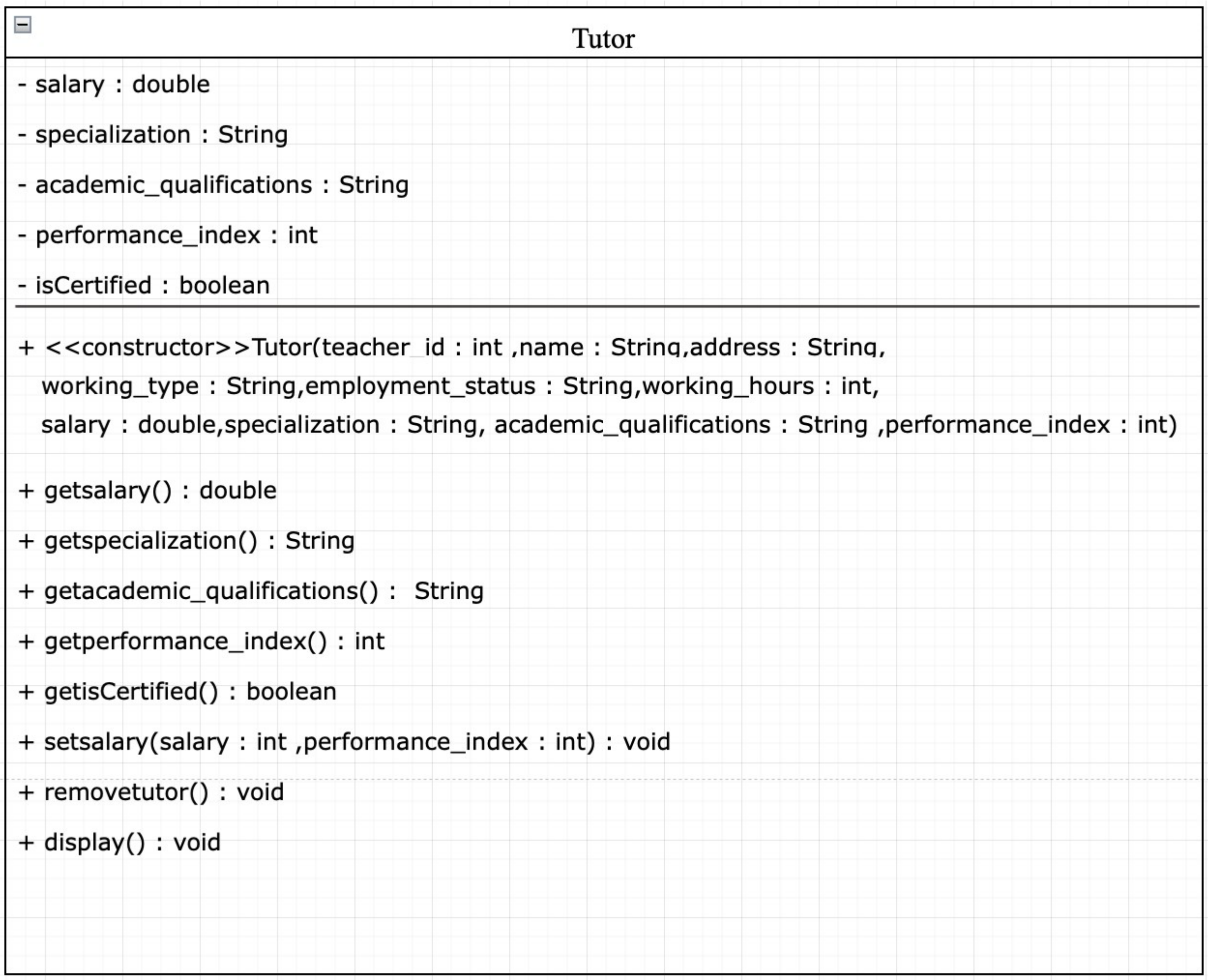

Tutor
- salary : double
- specialization : String
- academic_qualifications : String
- performance_index : int
- isCertified : boolean
+ <<constructor>>Tutor(teacher_id : int ,name : String,address : String, working_type : String,employment_status : String,working_hours : int, salary : double,specialization : String, academic_qualifications : String ,performance_index : int)
+ getsalary() : double
+ getspecialization() : String
+ getacademic_qualifications() : String
+ getperformance_index() : int
+ getisCertified() : boolean
+ setsalary(salary : int ,performance_index : int) : void
+ removetutor() : void
+ display() : void

Figure 3:Class Diagram :Tutor Class

Combined Relationship Diagram:

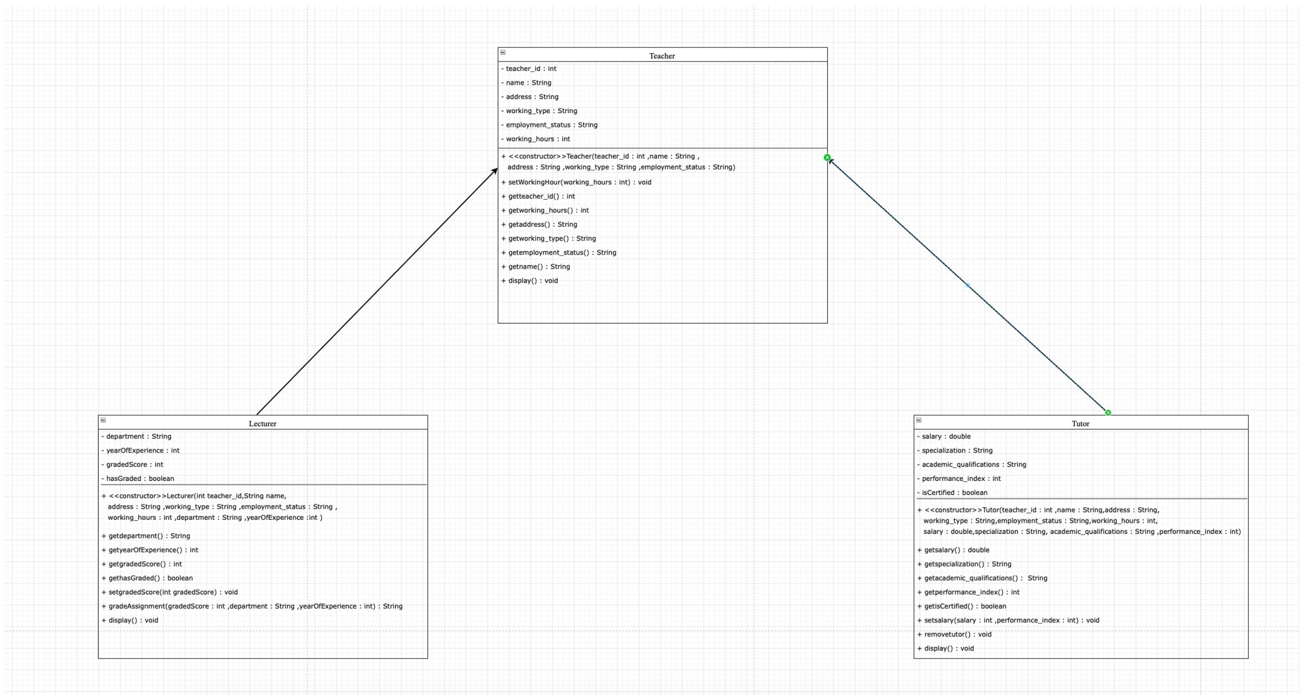

Teacher
- teacher_id : int
- name : String
- address : String
- working_type : String
- employment_status : String
- working_hours : int
+ <<constructor>>Teacher(teacher_id : int ,name : String , address : String ,working_type : String ,employment_status : String)
+ setWorkingHour(working_hours : int) : void
+ getteacher_id() : int
+ getworking_hours() : int
+ getaddress() : String
+ getworking_type() : String
+ getemployment_status() : String
+ getname() : String
+ display() : void
Lecturer
- department : String
- yearOfExperience : int
- gradedScore : int
- hasGraded : boolean
+ <<constructor>>Lecturer(int teacher_id,String name, address : String ,working_type : String ,employment_status : String , working_hours : int ,department : String ,yearOfExperience :int)
+ getdepartment() : String
+ getyearOfExperience() : int
+ getgradedScore() : int
+ gethasGraded() : boolean
+ setgradedScore(int gradedScore) : void
+ gradeAssignment(gradedScore : int ,department : String ,yearOfExperience : int) : String
+ display() : void
Tutor
- salary : double
- specialization : String
- academic_qualifications : String
- performance_index : int
- isCertified : boolean
+ <<constructor>>Tutor(teacher_id : int ,name : String,address : String, working_type : String,employment_status : String,working_hours : int, salary : double,specialization : String, academic_qualifications : String ,performance_index : int)
+ getsalary() : double
+ getspecialization() : String
+ getacademic_qualifications() : String
+ getperformance_index() : int
+ getisCertified() : boolean
+ setsalary(salary : int ,performance_index : int) : void
+ removetutor() : void
+ display() : void

Figure 4:Class Diagram :Combined Relationship Diagram (draw.io)

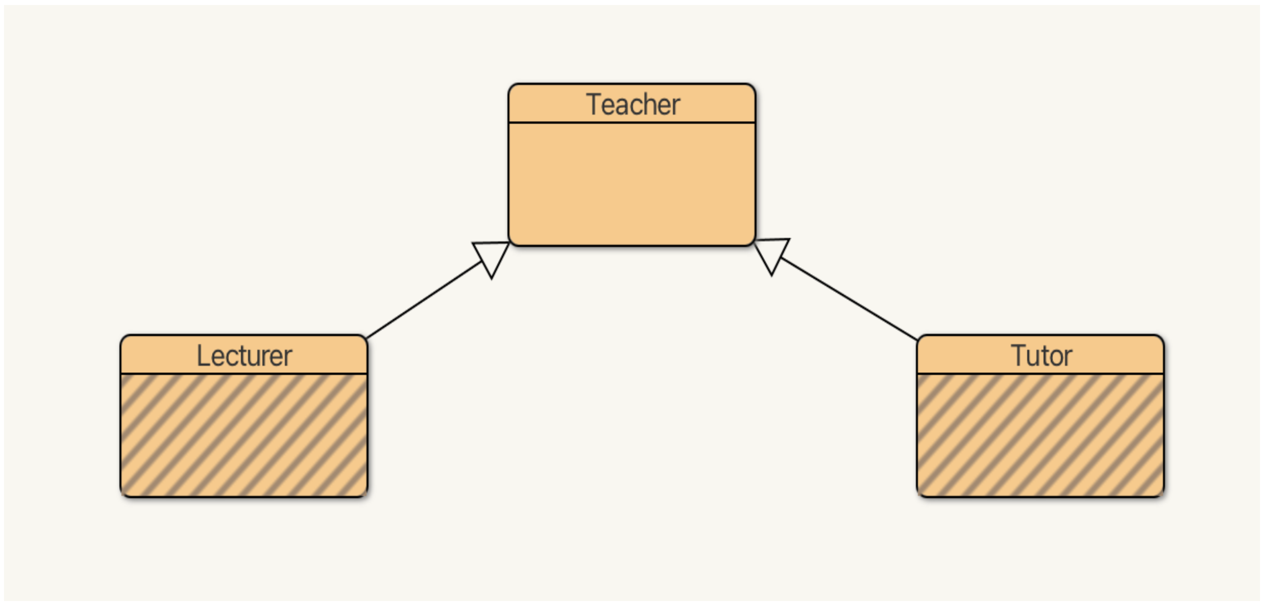


Figure 5:Class Diagram :Combined Class Diagram(BlueJ)

Pseuducode

Pseudocode is a detailed yet readable description of what computer program or algorithm should do. It is written in a formal yet readable style that uses a natural syntax and formatting so it can be easily understood by programmers and others involved in the development process. (Technopedia, 2023)

Pseudocode for Teacher

```
CREATE Parent Class Teacher
DO
DECLARE instance variable teacher_id
DECLARE instance variable name
DECLARE instance variable address
DECLARE instance variable working_type
DECLARE instance variable employment_status
DECLARE instance variable working_hours
END DO

CREATE Constructor Teacher(int teacher_id, String name, String address, String
working_type, String employment_status)
DO
    ASSIGN this.teacher_id = teacher_id
   ASSIGN this.name = name
   ASSIGN this.address = address
   ASSIGN this.working_type = working_type
```

```
    ASSIGN this.employment_status = employment_status
END DO


CREATE Getter Method getteacher_id()
DO
   RETURN this.teacher_id
END DO
CREATE Getter Method getworking_hours()
DO
   RETURN this.working_hours
END DO


CREATE Getter Method getaddress()
DO
   RETURN this.address
END DO


CREATE Getter Method getworking_type()
DO
   RETURN this.working_type
END DO
```

```
CREATE Getter Method getemployment_status()
DO
   RETURN this.employment_status
END DO

CREATE Getter Method getname()
DO
   RETURN this.name
END DO

CREATE Setter Method setWorkingHour(int working_hours)
DO
   ASSIGN this.working_hours = working_hours
END DO

CREATE Display Method display()
DO
   PRINT(“Teacher_ID = " + this.getteacher_id())
   PRINT("Teacher_Name = " + this.getname())
   PRINT("Teacher_Address = " + this.getaddress())
   PRINT("Working Type = " + this.getworking_type())
   PRINT("Employee_Status = " + this.getemployment_status())

   IF (this.getworking_hours() == 0)
DO
      PRINT("Working Hour Has Not Been Set Yet")
```

```
END DO
ELSE
    PRINT("Working_Hour = " + this.getworking_hours())
END IF
END DO
```

Pseudocode for Lecturer

```
CREATE Child Class Lecturer
DO
   DECLARE instance variable department
   DECLARE instance variable yearOfExperience
   INITIALISE instance variable gradedScore = 0
   INITIALISE instance variable  hasGraded = false
END DO

   CREATE Constructor Lecturer(int teacher_id, String name, String address, String
working_type, String employment_status, int working_hours, String department, int
yearOfExperience)
DO
   ASSIGN super(teacher_id, name, address, working_type, employment_status)
   ASSIGN super.setWorkingHour(working_hours)
   ASSIGN this.department = department
   ASSIGN this.yearOfExperience = yearOfExperience

END DO
   CREATE Getter Method getDepartment()
DO
      RETURN this.department
END DO

   CREATE Getter Method getYearOfExperience()
```

```
DO
    RETURN this.yearOfExperience
END DO


  CREATE Getter Method getGradedScore()
DO
    RETURN this.gradedScore
END DO
  CREATE Getter Method getHasGraded()
DO
    RETURN this.hasGraded
END DO
  CREATE Getter Method setGradedScore(int gradedScore)
DO
    this.gradedScore = gradedScore
END DO


CREATE Setter Method For gradedScore setgradedScore(int gradedScore)
DO
      RETURN this.gradedScore
END DO


  CREATE Grade Assignment Method gradeAssignment(int gradedScore, String department,
int yearOfExperience)
DO
    IF yearOfExperience >= 5 AND this.department.equals(department)
THEN
      IF gradedScore > 70
      ASSIGN this.gradedScore=gradedScore;
```

```
     ASSIGN this.hasGraded=true
       RETURN "Grade A"

    ELSE IF gradedScore > 60
    ASSIGN this.gradedScore=gradedScore;
     ASSIGN this.hasGraded=true
       RETURN "Grade B"

    ELSE IF gradedScore > 50
    ASSIGN this.gradedScore=gradedScore;
     ASSIGN this.hasGraded=true
   RETURN "Grade C"

    ELSE IF gradedScore > 40
    ASSIGN this.gradedScore=gradedScore;
     ASSIGN this.hasGraded=true
       RETURN "Grade D"

    ELSE IF gradedScore < 40
    ASSIGN this.gradedScore=gradedScore;
     ASSIGN this.hasGraded=true
       RETURN "Grade E"

END IF

ELSE
PRINT  ("Teacher Not Eligible For Grade Assignment")
```

```
END DO

CREATE   Display Method display()

CALL  super.display()
PRINT ("Department = " + getdepartment())
PRINT ("Year Of Experience = " + getyearOfExperience())

IF this.gradedScore==0
DO
PRINT ("Grade Score Not Assigned")
END DO
END IF
ELSE
DO
PRINT ("Grade Score = "+getgradedScore()")

END DO
```

Pseudocode for Tutor

```
CREATE Child Class Tutor

    DECLARE instance variable salary
    DECLARE instance variable specialization
    DECLARE instance variable academic_qualifications
    DECLARE instance variable performance_index
    INITIALISE instance variable isCertified = false

    CREATE CONSTRUCTOR Tutor(int teacher_id, String name, String address, String
working_type, String employment_status, int working_hours, double salary, String
specialization, String academic_qualifications, int performance_index)

        CALL super(teacher_id, name, address, working_type, employment_status)
        CALL super.setWorkingHour(working_hours)

        ASSIGN this.salary = salary
        ASSIGN this.specialization = specialization
        ASSIGN this.academic_qualifications = academic_qualifications
        ASSIGN this.performance_index = performance_index

    CREATE Getter Method getSalary()
DO
        RETURN this.salary
END DO
    CREATE Getter Method getSpecialization()
```

```
DO
    RETURN this.specialization
END DO
  CREATE Getter Method getAcademicQualifications()
DO
    RETURN this.academic_qualifications
END DO
 CREATE Getter Method getPerformanceIndex()
DO
    RETURN this.performance_index
END DO
  CREATE Getter Method getIsCertified()
DO
    RETURN this.isCertified
END DO
  CREATE Setter Method setSalary(int salary, int performance_index)
DO
    IF (this.performance_index > 5 AND this.getWorkingHours() > 20)
THEN
      IF (performance_index >= 5 AND performance_index <= 7)
        this.salary = salary + (0.05 * salary)
      ELSE IF (performance_index >= 8 AND performance_index <= 9)
        this.salary = salary + (0.10 * salary)
      ELSE IF (performance_index == 10)
        this.salary = salary + (0.20 * salary)

      ASSIGN this.isCertified = true
      END IF
ELSE
      PRINT("Salary Has Not Been Approved")
END IF
END DO



CREATE A Method removeTutor() to remove teacher who are not certified
DO
      IF (this.isCertified==false)
THEN
      ASSIGN this.salary = 0
      ASSIGN this.specialization = ""
      ASSIGN this.academic_qualifications = ""
      ASSIGN this.performance_index = 0
      ASSIGN this.isCertified = false
END IF

CREATE Display Method display() to display variable with suitable annotation
DO
   IF (this.isCertified==true)
      CALL super.display()
      PRINT("Salary = " + this.getSalary())
      PRINT("Specialization = " + this.getSpecialization())
      PRINT("Academic Qualification = " + this.getAcademicQualifications())
      PRINT("Performance Index = " + this.getPerformanceIndex())
```

```
ELSE
        CALL super.display()
      END IF
 END DO
```

Method Description

Method Description is in brief explanation of what alll the function present in the class does and what parameter it takes and how it is related to another method and variables. It is great for understanding the function and use of function present in the class with a quick glance at it .

Teacher Class

Method	Description
setWorkingHour(int working_hours)	This method is a setter method and it takes a parameter(working_hours) of type int and assigns it to instance variable working_hours.
getteacher_id()	This is an accessor method which returns the value of instance variable teacher_id. The return type of this method is int.
getworking_hours()	This is an accessor method which returns the value of instance variable working_hours. The return type of this method is int.
getaddress()	This is an accessor method which returns the value of instance variable address. The return type of this method is String.
getworking_type()	This is an accessor method which returns the value of instance variable working_type. The return type of this method is String.

getemployment_status()	This is an accessor method which returns the value of instance variable employment_status. The return type of this method is String.
getname()	This is an accessor method which returns the value of instance variable name. The return type of this method is String.
display()	This method displays all the value of instance variables with suitable annotations and it accepts zero parameter with return type void.

Table 1:Method Descriptions: Teacher Class

Lecturer Class

Method	Description
getdepartment()	This is an accessor method which returns the value of instance variable department. The return type of this method is String.
getyearOfExperience()	This is an accessor method which returns the value of instance variable yearOfExperience. The return type of this method is int.
getgradedScore()	This is an accessor method which returns the value of instance variable gradedScore. The return type of this method is int.
gethasGraded()	This method is designed to return a boolean value that indicates whether the instance variable hasGraded is true or false.
setgradedScore(int gradedScore)	This method is a mutar method and it takes a parameter(gradedScore) of type int and assigns it to instance variable gradedScore.
gradeAssignment(int gradedScore,String department,int yearOfExperience)	This method grades the assignments of students by experienced teacher of same field and its return type is String.
display()	This method displays all the value of instance variables of its own class along with Teacher class with suitable annotations and it accepts zero parameter with return type void.

Table 2:Method Descriptions: Lecturer Class

Tutor Class

Method	Description
double getsalary()	This is an accessor method which returns the value of instance variable yearOfExperience. The return type of this method is double.
getspecialization()	This is an accessor method which returns the value of instance variable specialization. The return type of this method is String.
getacademic_qualifications()	This is an accessor method which returns the value of instance variable academic_qualifications. The return type of this method is String.
getperformance_index()	This is an accessor method which returns the value of instance variable performance_index. The return type of this method is int.
getisCertified()	This method is designed to return a boolean value that indicates whether the instance variable isCertified is true or false.
setsalary(int salary,int performance_index)	This method sets salary to each tutors and if their performance_index is greater than 5 and working hour is greater than 20 then their appraisal is done and salary is given along with their appraisal amount.
removetutor()	This Methods removes the tutor and sets all of theirattributes to zero and isCertified to false
display()	This will display all the details of teacher along with the details of teacher from Teacher Class if isCertified is true but if its false it will only display details of parent class

Table 3:Method Descriptions::Tutor Class

Testing

Test 1:

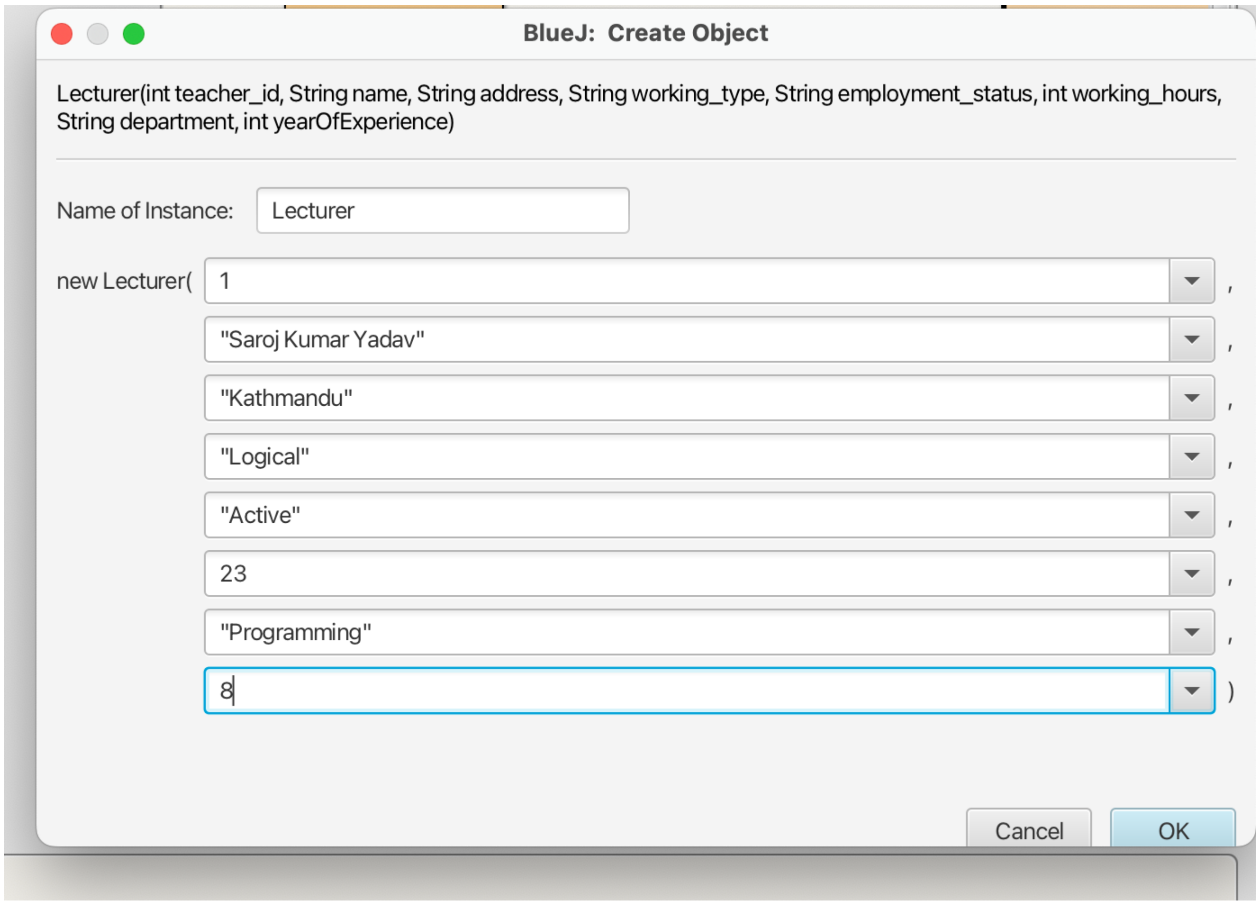
Inspect the Lecturer class, grade the Assignment and re-inspect the Lecturer Class

Test Number	1
Objective For Testing	To Inspect The Lecturer Class First Then Grading Assignment and Reinspecting Lecturer Class.
Actions Taken	1) Calling Constructor Assigning Following Values as Arguments : • teacher_id=1; • name=”Saroj Kumar Yadav”; • address=”Kathmandu”; • working_type=”Logical”; • employement_status=”Active”; • working_hours=23; • department=”Programming”; • yearOfExperience=8; 2) Inspecting The Lecturer Class 3) Assigning Grade Through gradeAssignment Method With • gradedScore = 78; • department=”Programming”; • yearOfExperience=8; 4) Re-Inspecting Lecturer Class

Anticipated Results	Assignment Should Be Graded and hasGraded should be true.
Actual Result	Assignment was Graded and hasGraded Was Set To true.
Result Conclusion	The Test Was Passed Successfully.

Table 4:Testing :Test1

Output Results

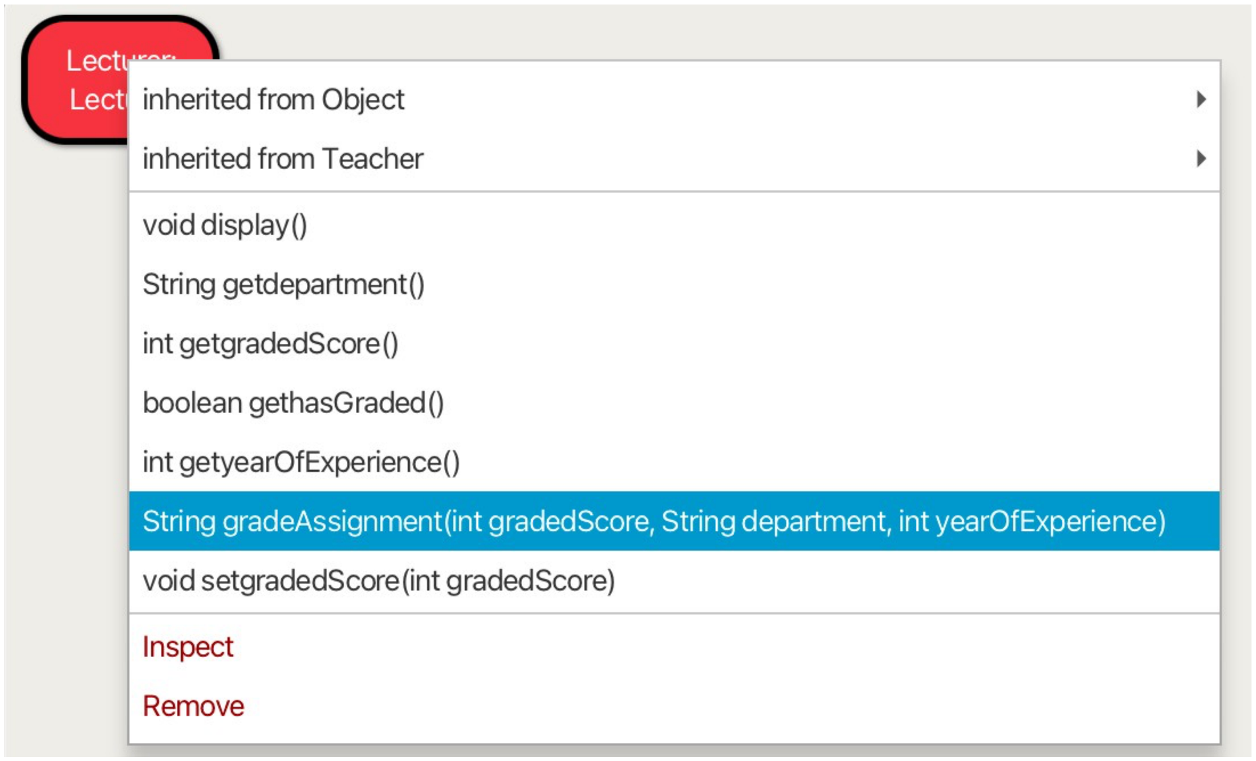


Lecturer(int teacher_id, String name, String address, String working_type, String employment_status, int working_hours, String department, int yearOfExperience)

Name of Instance: Lecturer

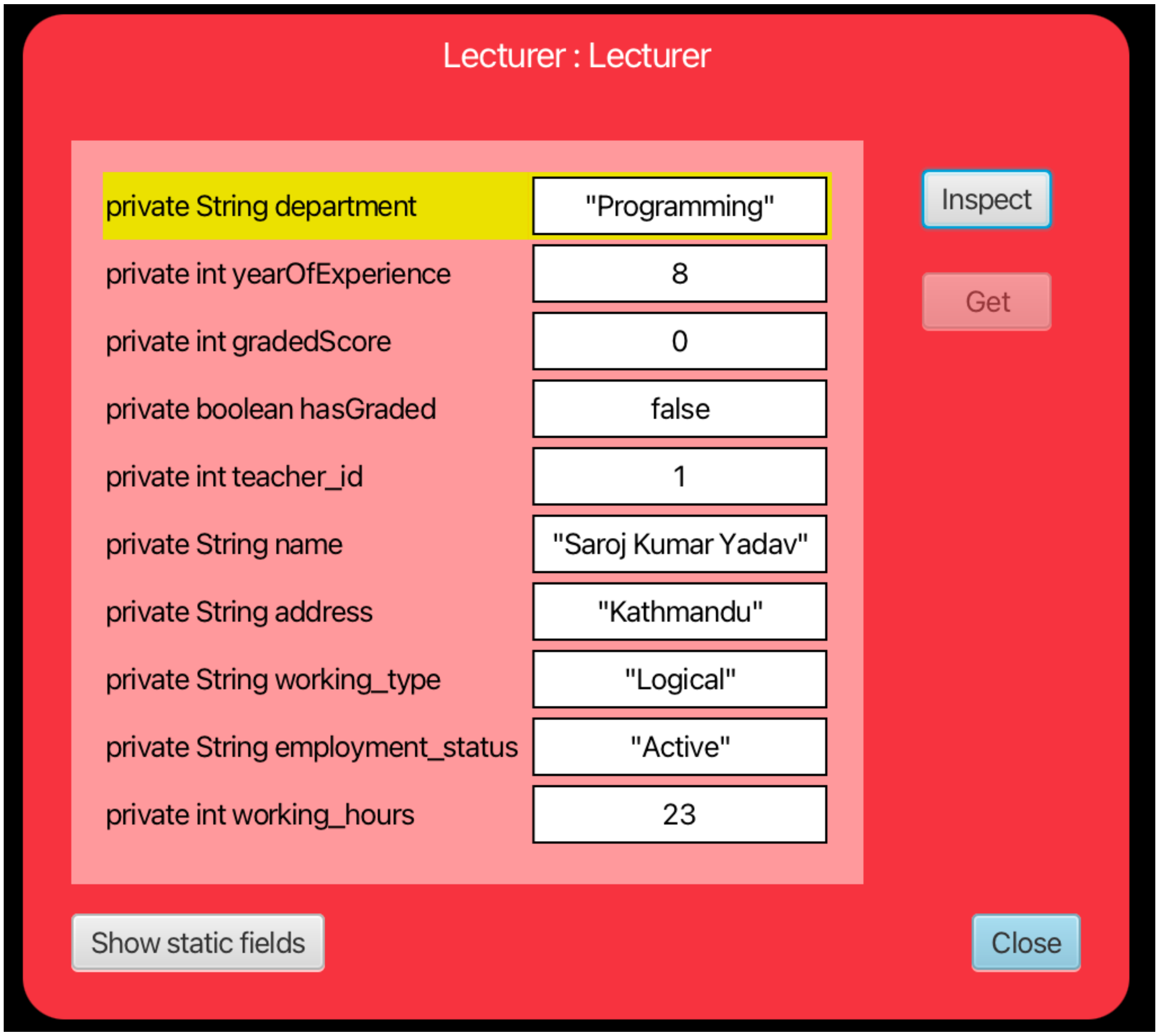
new Lecturer(1 , "Saroj Kumar Yadav" , "Kathmandu" , "Logical" , "Active" , 23 , "Programming" , 8)

Figure 6:Test1:Calling Constructor And Assigning Values



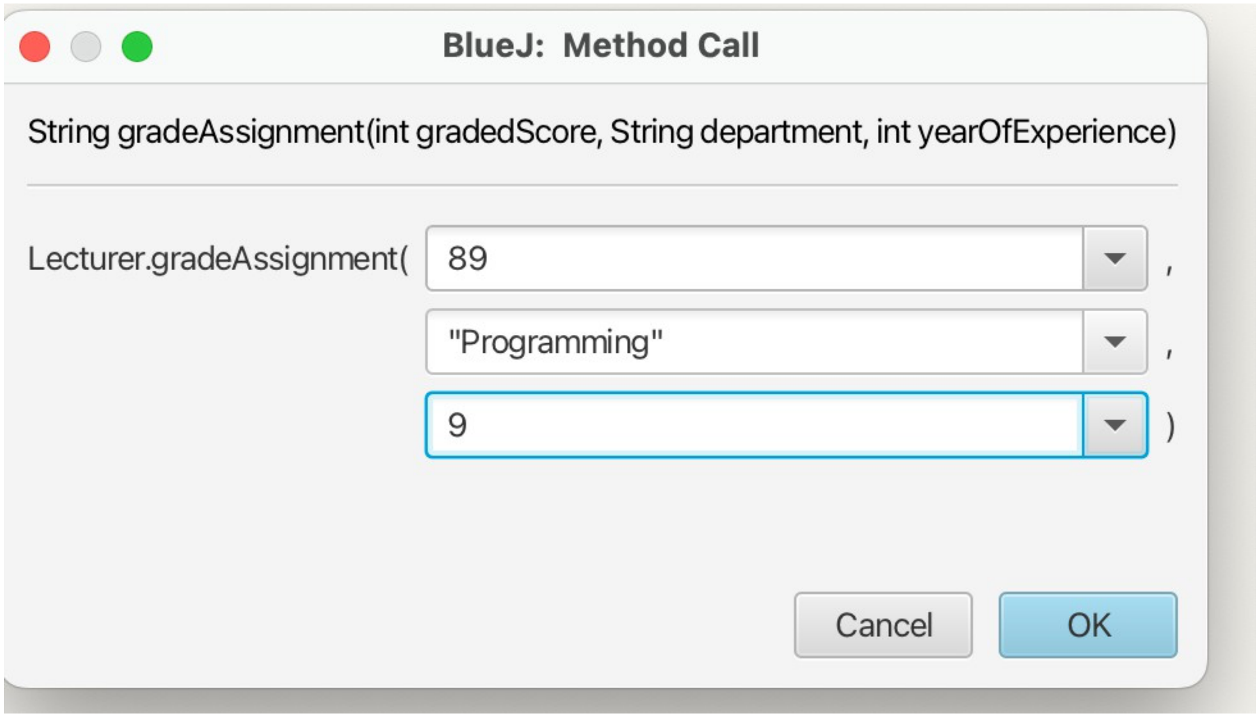
```
void display()
String getdepartment()
int getgradedScore()
boolean gethasGraded()
int getyearOfExperience()
String gradeAssignment(int gradedScore, String department, int yearOfExperience)
void setgradedScore(int gradedScore)
```

Figure 7:Test1:Showing All Methods Of Lecturer Class


Lecturer : Lecturer

private String department	"Programming"
private int yearOfExperience	8
private int gradedScore	0
private boolean hasGraded	false
private int teacher_id	1
private String name	"Saroj Kumar Yadav"
private String address	"Kathmandu"
private String working_type	"Logical"
private String employment_status	"Active"
private int working_hours	23

Figure 8:Inspection Before Grade Assignment

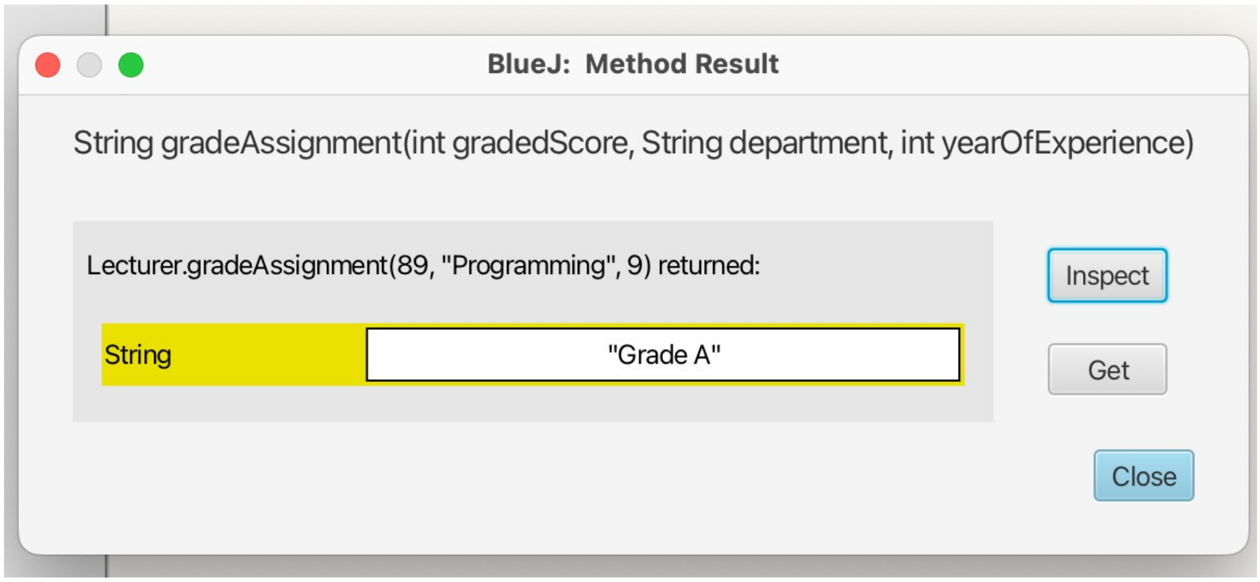


BlueJ: Method Call

String gradeAssignment(int gradedScore, String department, int yearOfExperience)

```
Lecturer.gradeAssignment( 89 ,
                          "Programming" ,
                          9 )
```

Figure 9:Test1:Calling GradeAssignment Method And Assigning Values

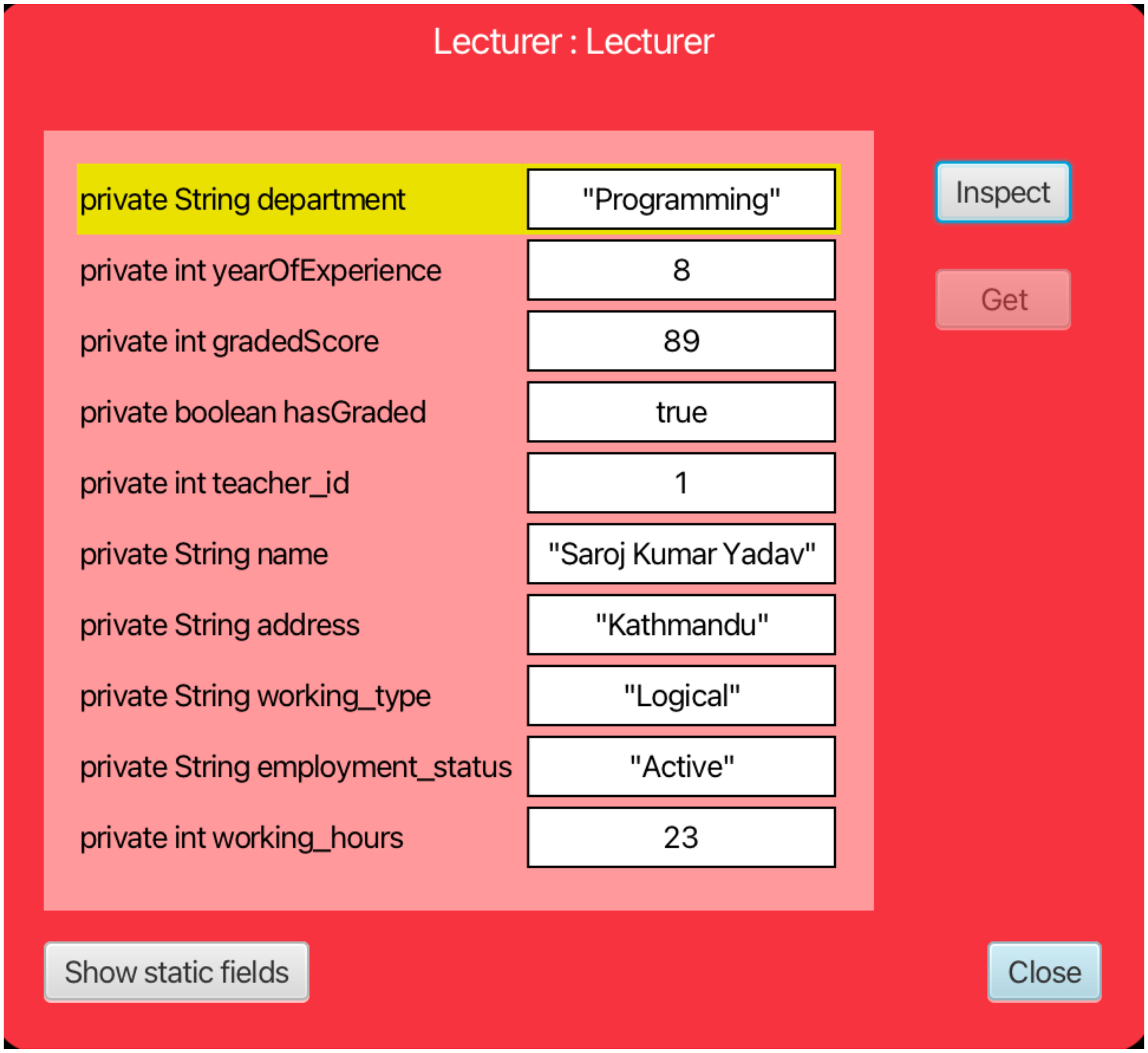


BlueJ: Method Result

String gradeAssignment(int gradedScore, String department, int yearOfExperience)

```
Lecturer.gradeAssignment(89, "Programming", 9) returned:
String "Grade A"
```

Figure 10:Test1:Output Result Of Grade Assignment



Lecturer : Lecturer

private String department	"Programming"
private int yearOfExperience	8
private int gradedScore	89
private boolean hasGraded	true
private int teacher_id	1
private String name	"Saroj Kumar Yadav"
private String address	"Kathmandu"
private String working_type	"Logical"
private String employment_status	"Active"
private int working_hours	23

Figure 11:Test1:Inspection After Grade Assignment

Test 2 :

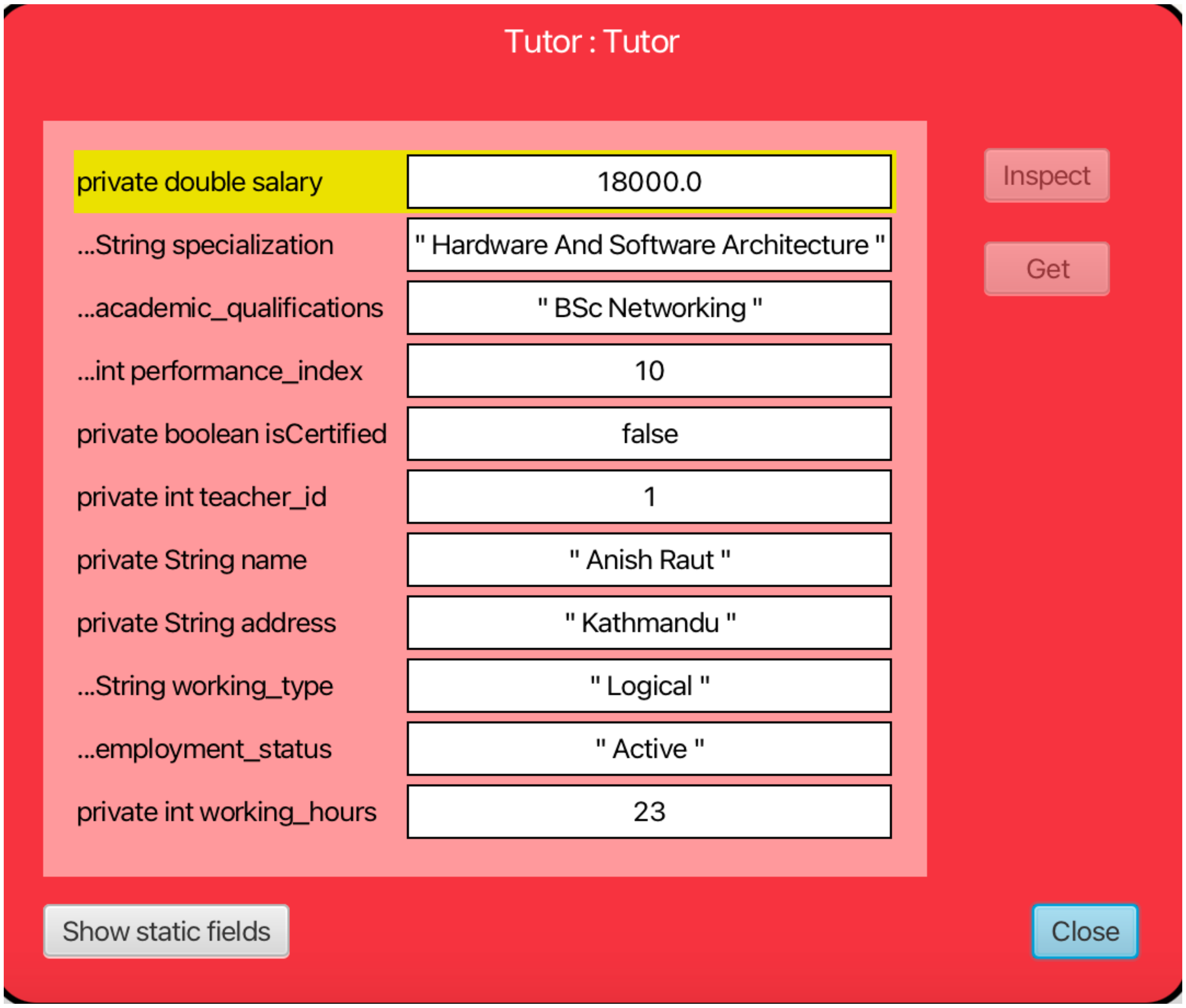
Inspect Tutor class, set salary and reinspect the Tutor class

Test Number	2
Objective For Testing	Inspect Tutor Class, Set Salary and Re-Inspect The Tutor Class
Actions Taken	1) Calling Constructor Assigning Following Values as Arguments : • teacher_id=1; • name=”Anish Raut”; • address=”Kathmandu”; • working_type=”Logical”; • employement_status=”Active”; • working_hours=23; • salary=18000; • specialization=”Hardware And Software Architecture”; • academic_qualifications=”BSc Networking”; • performance_index=10; 2) Inspecting Tutor Class 3) Calling setsalary Method Assigning Following Values as Arguments : • salary=18000; • performance_index=10; 4) Re-Inspecting The Tutor Class

Anticipated Results	Salary Should be Increased as Per Their Performance Index And Working Hour.
Actual Result	Salary Which Was Previously Set to 18000 was Changed And Set To 19800 .
Result Conclusion	The Test Passed Successfully.

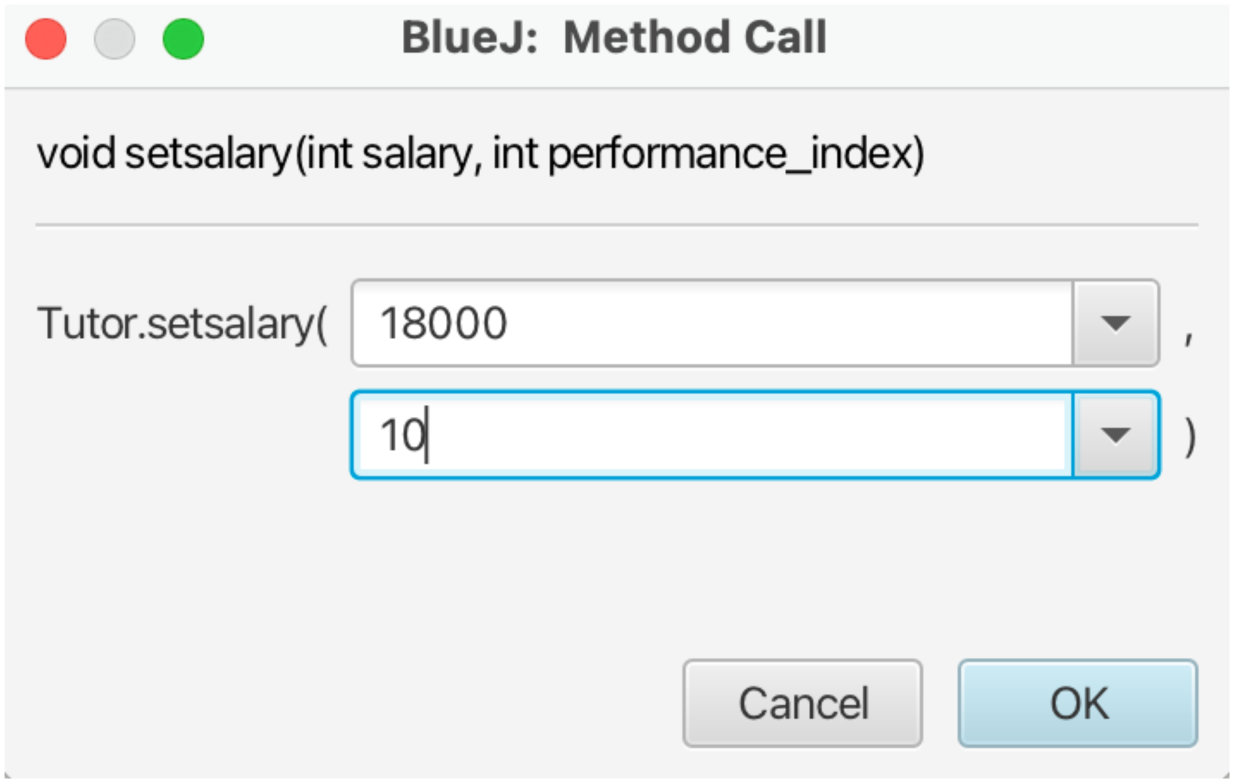
Table 5:Testing :Test2

Output Results :



private double salary	18000.0
...String specialization	" Hardware And Software Architecture "
...academic_qualifications	" BSc Networking "
...int performance_index	10
private boolean isCertified	false
private int teacher_id	1
private String name	" Anish Raut "
private String address	" Kathmandu "
...String working_type	" Logical "
...employment_status	" Active "
private int working_hours	23

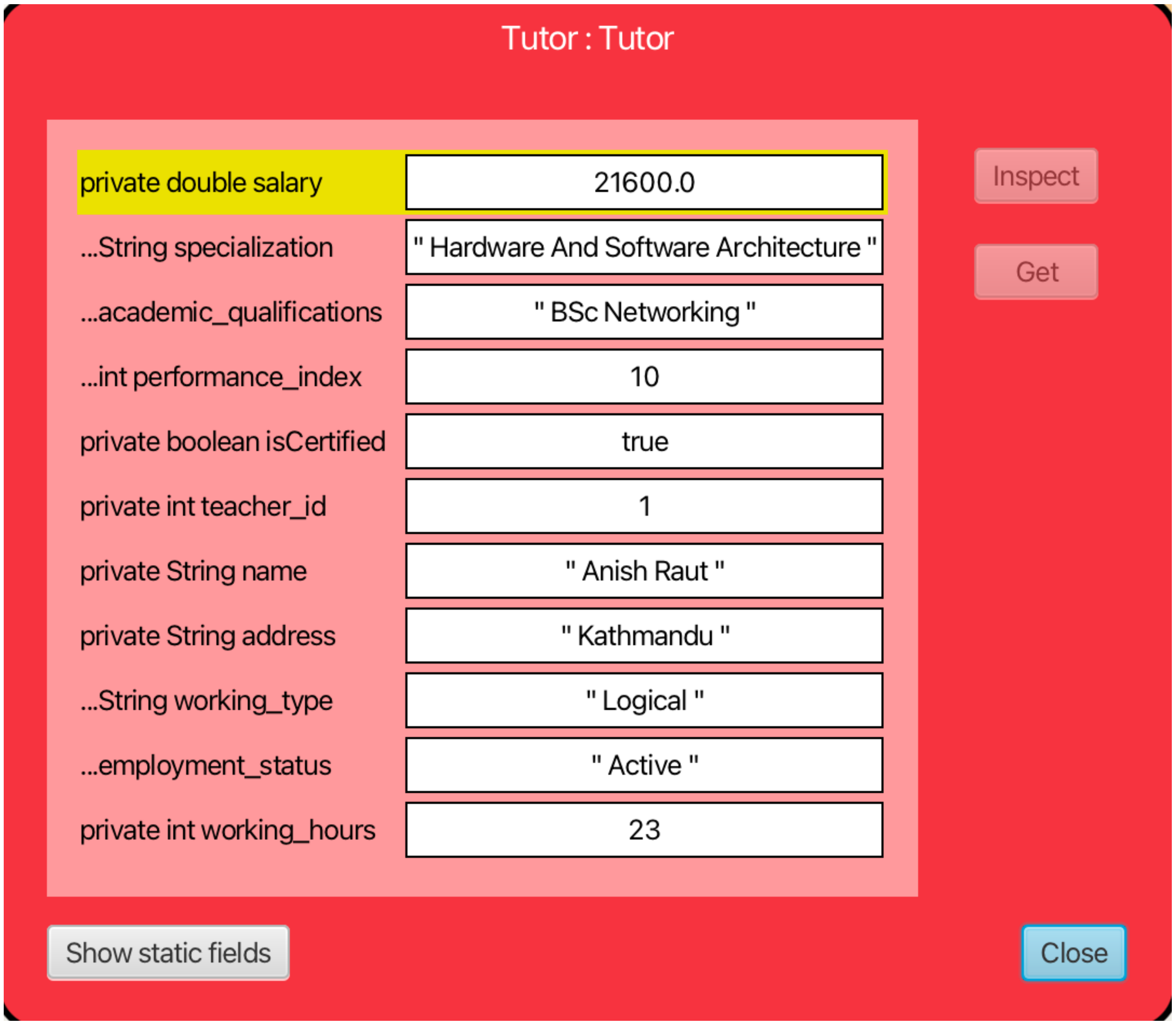
Figure 12:Test2:Inspecting Tutor Class


BlueJ: Method Call

void setsalary(int salary, int performance_index)

Tutor.setsalary(18000 , 10)

Figure 13:Test2:Setting Salary


Tutor : Tutor

private double salary	21600.0
...String specialization	" Hardware And Software Architecture "
...academic_qualifications	" BSc Networking "
...int performance_index	10
private boolean isCertified	true
private int teacher_id	1
private String name	" Anish Raut "
private String address	" Kathmandu "
...String working_type	" Logical "
...employment_status	" Active "
private int working_hours	23

Figure 14:Inspecting After Setting Salary

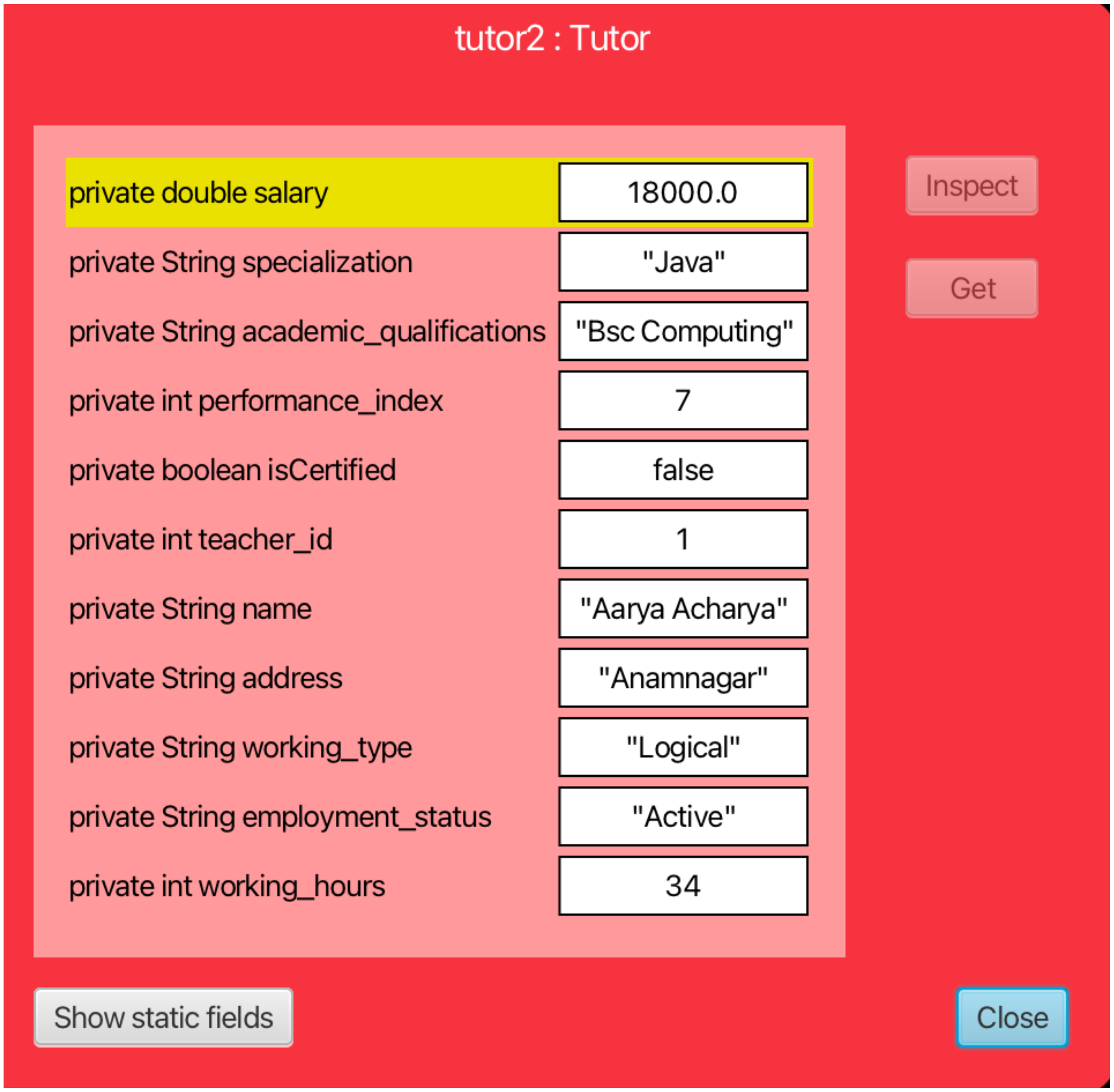
Test 3:

Inspect Tutor class again after removing the tutor.

Test Number	**3**
Objective For Testing	Inspect Tutor Class Again After Removing Tutor
Actions Taken	1) Calling Constructor Assigning Following Values as Arguments : • teacher_id=1; • name=”Aarya Acharya”; • address=”Anamnagar”; • working_type=”Logical”; • employement_status=”Active”; • working_hours=34; • salary=18000; • specialization=”Java”; • academic_qualifications=”Bsc Computing”; • performance_index=7; 2) Inspecting Tutor Class 3) Calling removetutor Function to Remove The Tutor . 4) Re-Inspecting The Tutor Class
Anticipated Results	Tutor Should Be Removed From The Inspect Field If He/She Is Not Certifed .
Actual Result	Tutor Was Successfully Removed Because Specific Tutor was Not Certified.
Result Conclusion	The Test Passed Successfully.

Table 6:Testing :Test3

<u>Output Results :</u>


tutor2 : Tutor

private double salary	18000.0
private String specialization	"Java"
private String academic_qualifications	"Bsc Computing"
private int performance_index	7
private boolean isCertified	false
private int teacher_id	1
private String name	"Aarya Acharya"
private String address	"Anamnagar"
private String working_type	"Logical"
private String employment_status	"Active"
private int working_hours	34

Figure 15:Test3:Inspecting Before Removing Tutor

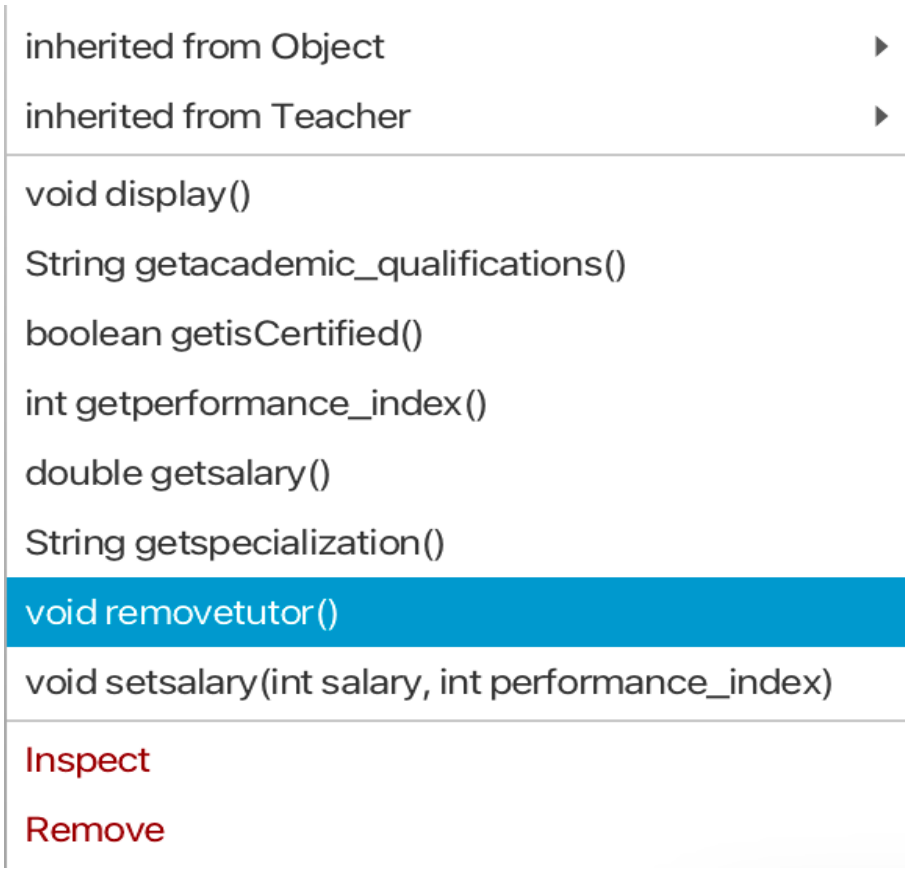
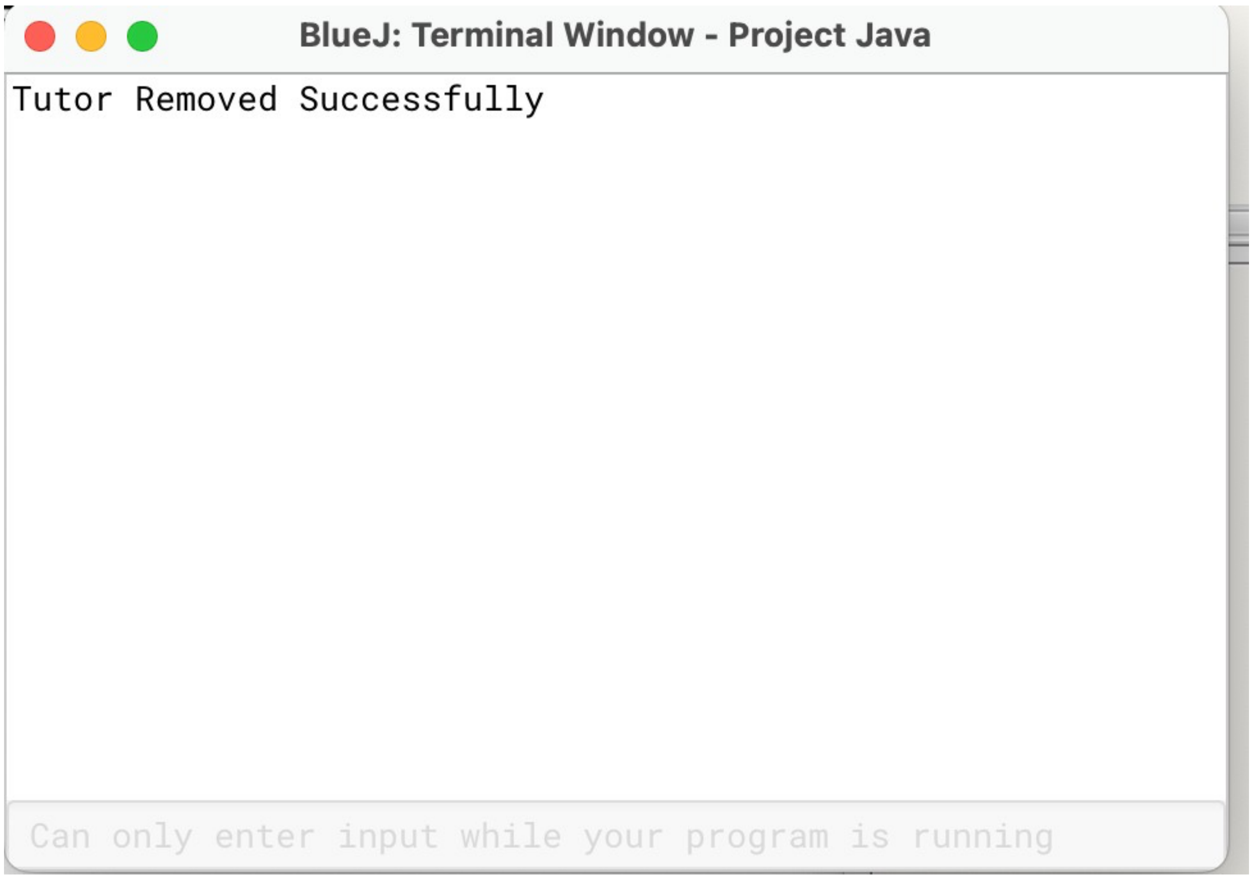

inherited from Object
inherited from Teacher
void display()
String getacademic_qualifications()
boolean getisCertified()
int getperformance_index()
double getsalary()
String getspecialization()
void removetutor()
void setsalary(int salary, int performance_index)
Inspect
Remove

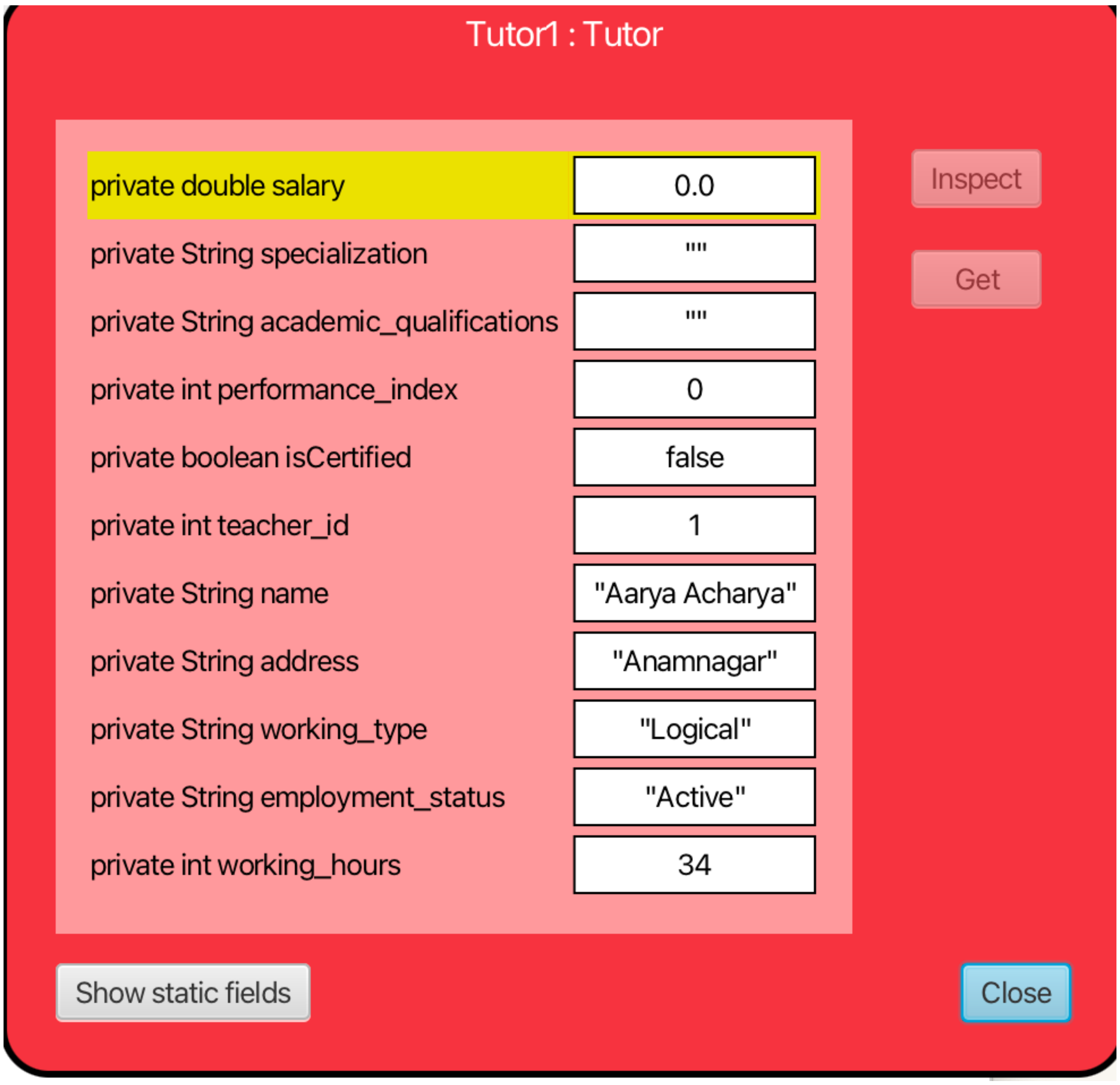
Figure 16:Test4:Showing All Methods Of Tutor Class


BlueJ: Terminal Window - Project Java

```
Tutor Removed Successfully
```

Can only enter input while your program is running

Figure 17:Test3:Output OF removetutor Method



private double salary	0.0
private String specialization	""
private String academic_qualifications	""
private int performance_index	0
private boolean isCertified	false
private int teacher_id	1
private String name	"Aarya Acharya"
private String address	"Anamnagar"
private String working_type	"Logical"
private String employment_status	"Active"
private int working_hours	34

Figure 18:Test3:Inspecting After Removing Tutor

Test 4

Display the details of Lecturer and Tutor classes.

Test Number	3
Objective For Testing	Displaying The Details Of The Lecturer And Teacher Class
Actions Taken	1) Made Objects of Both Lecturer and Tutor classes 2) Call The Display Function In Both Teacher And Lecturer Class.
Anticipated Results	The Display Method in Both The Method Should Print Detail Information Of Teacher And Lecturer Class
Actual Result	Both The Display Function Worked and Displayed All The Information Of Both The Classes.
Result Conclusion	The Test Passed Successfully.

Table 7:Testing :Test4

Output Results

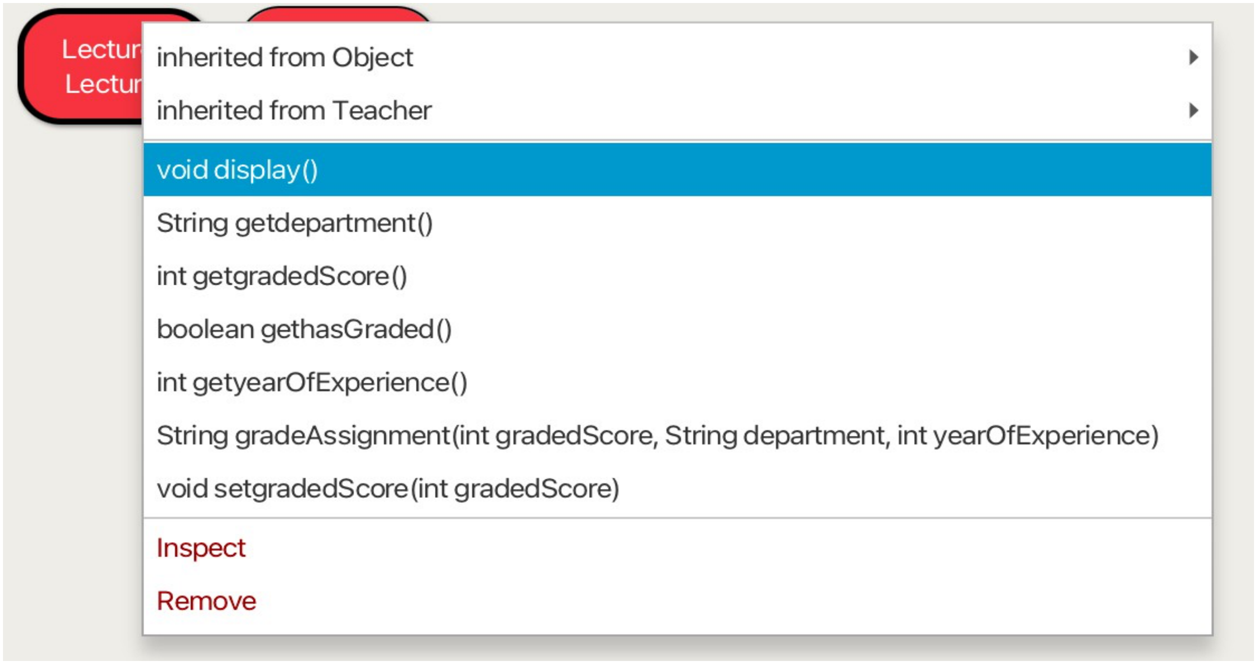
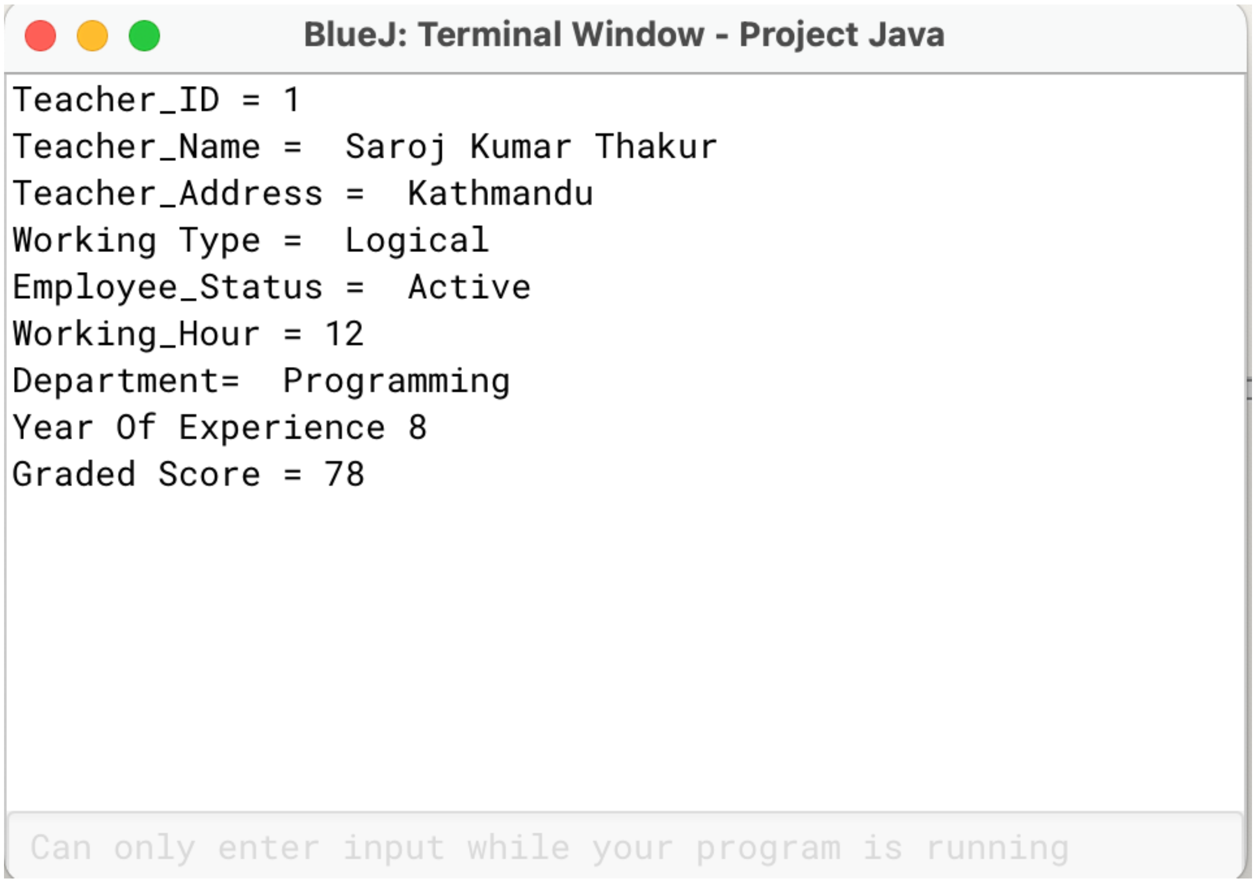


Figure 19:Test4:Showing All Methods Of Lecturer Class


BlueJ: Terminal Window - Project Java

```
Teacher_ID = 1
Teacher_Name =  Saroj Kumar Thakur
Teacher_Address =  Kathmandu
Working Type =  Logical
Employee_Status =  Active
Working_Hour = 12
Department=  Programming
Year Of Experience 8
Graded Score = 78
```

Can only enter input while your program is running

Figure 20:Test4:Output Of Display Method Of Lecturer Class

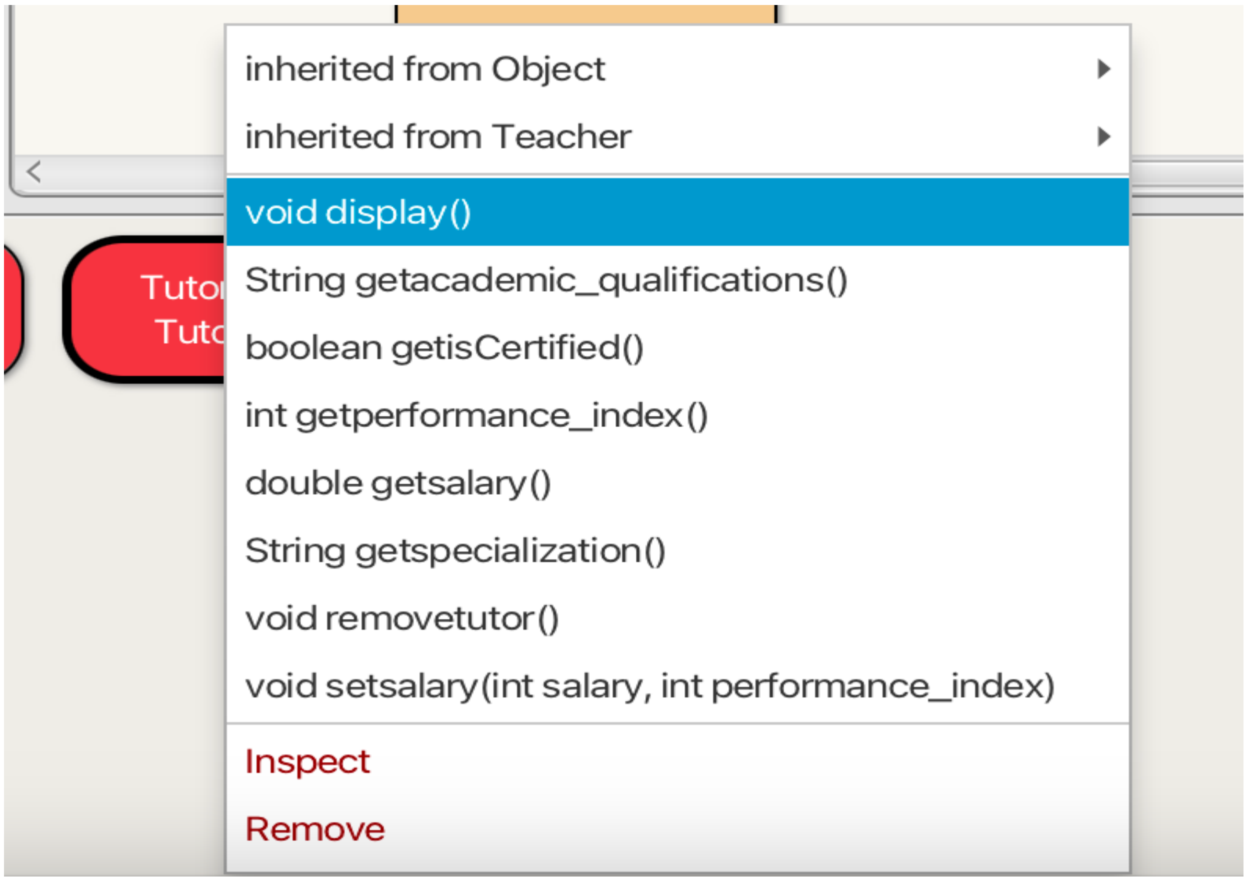
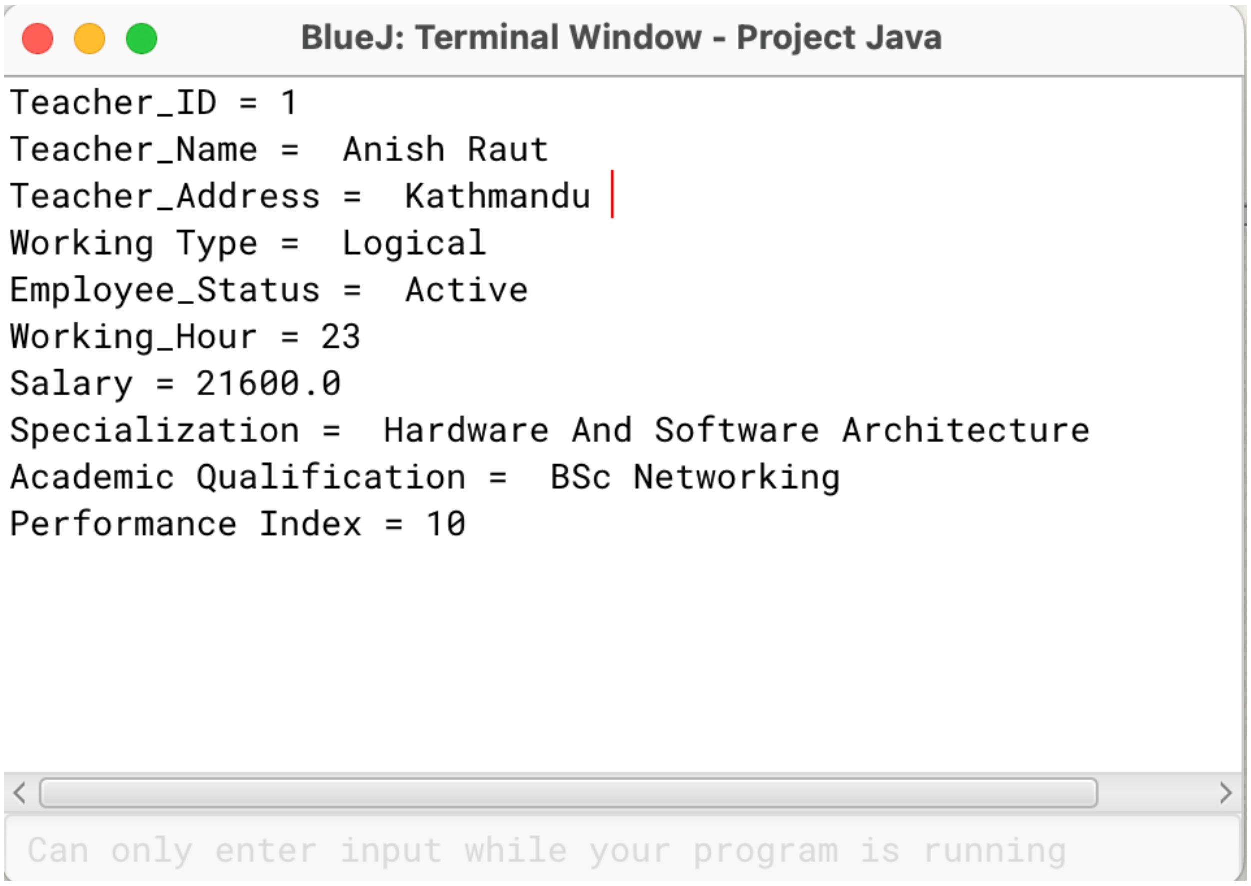


Figure 21:Showing All Method Of Tutor Class


BlueJ: Terminal Window - Project Java

```
Teacher_ID = 1
Teacher_Name =  Anish Raut
Teacher_Address =  Kathmandu
Working Type =  Logical
Employee_Status =  Active
Working_Hour = 23
Salary = 21600.0
Specialization =  Hardware And Software Architecture
Academic Qualification =  BSc Networking
Performance Index = 10
```

Can only enter input while your program is running

Figure 22:Test4:Output Of Display Method Of Tutor Class

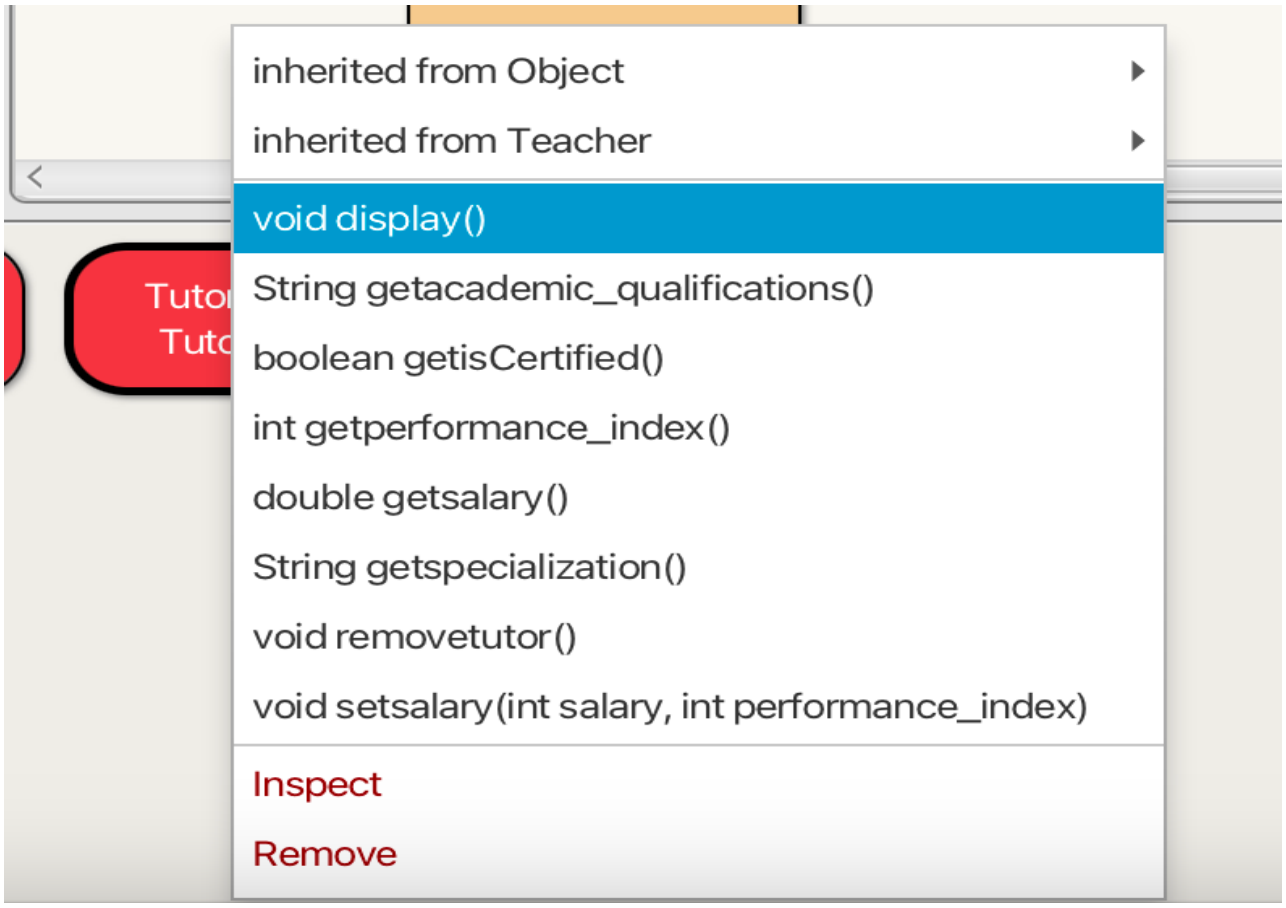


Figure 23:Test4:Showing All Methods Of Tutor Class

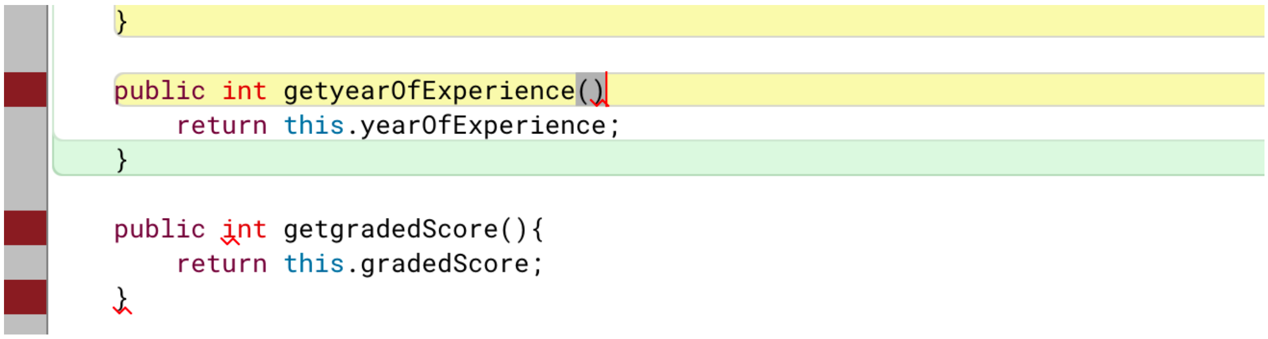
Error Detection and Error Removal

Syntax Error:

A syntax error in computer science is an error in the syntax of a coding or programming language, entered by a programmer. Syntax errors are caught by a software program called a compiler, and the programmer must fix them before the program is compiled and then run. (Technopedia, 2023)

Error Detection :

The syntax error which i encountered in Teacher Class while doing the project was because i didnt put an open paranthesis ({)in getyearofExperience() method.

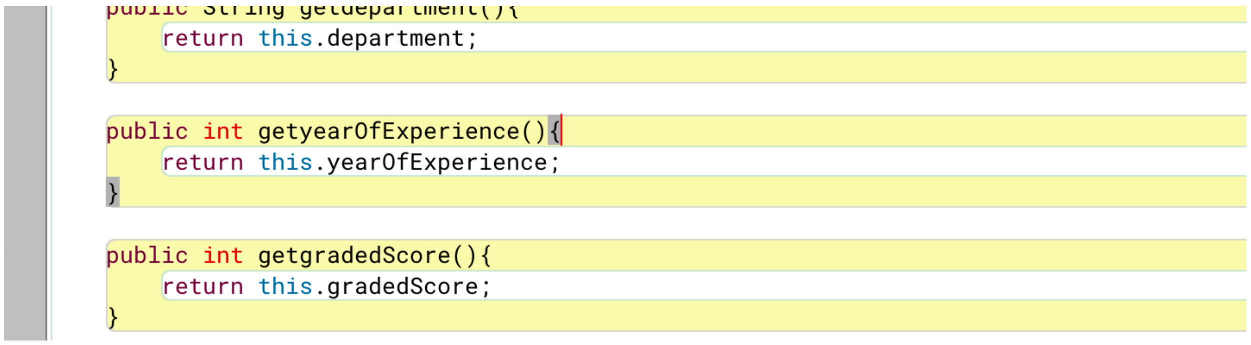


```
}

public int getyearOfExperience()
    return this.yearOfExperience;
}

public int getgradedScore(){
    return this.gradedScore;
}
```

Figure 24:Error Detection and Error Removal:Error1

Error Removal :

The error was then solved after adding the closing parenthesis as shown in the screenshot below:



```
    return this.department;
}

public int getyearOfExperience(){
    return this.yearOfExperience;
}

public int getgradedScore(){
    return this.gradedScore;
}
```

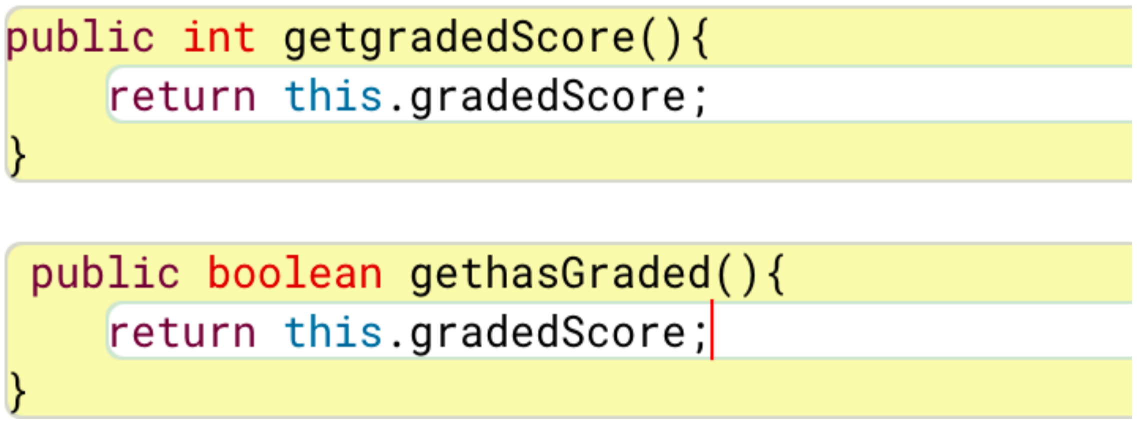
Figure 25:Error Detection and Error Removal:Solution1

Sementic Error:

A semantic error is a problem in our code that prevents the interpreter from understanding it. There may be nothing wrong the logic of our code, but it will cause the program to crash the way we have written.

Error Detection :

The second error that I encountered was sementic error in the Lecturer Class inside gethasGraded() method .Instead of "this.hasGraded" I wrote "this.gradedScore" as shown in the picture :

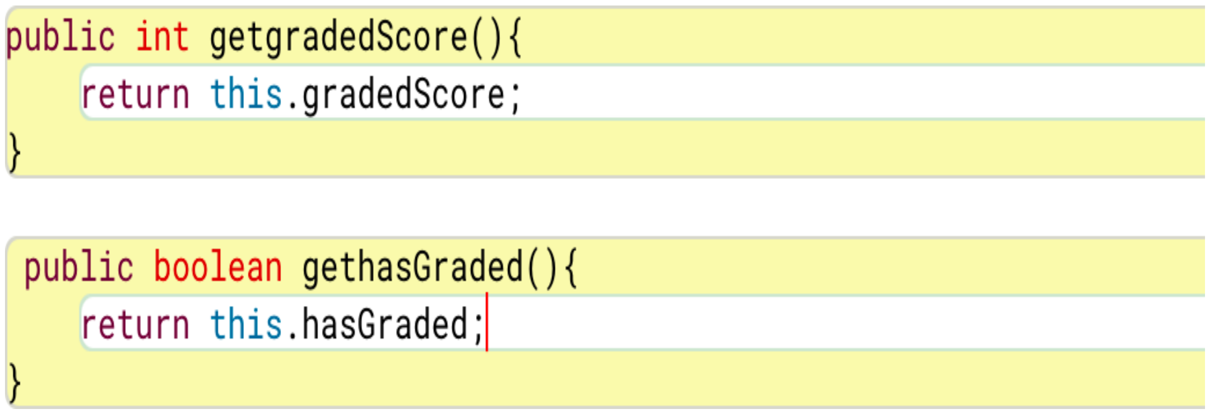


```
public int getgradedScore(){
    return this.gradedScore;
}

 public boolean gethasGraded(){
    return this.gradedScore;
}
```

Figure 26:Error Detection and Error Removal:Error2

Error Removal :

The error was then solved after writing “this.hasGraded” which was previously written as “this.gradedScore”.



```
public int getgradedScore(){
    return this.gradedScore;
}

public boolean gethasGraded(){
    return this.hasGraded;
}
```

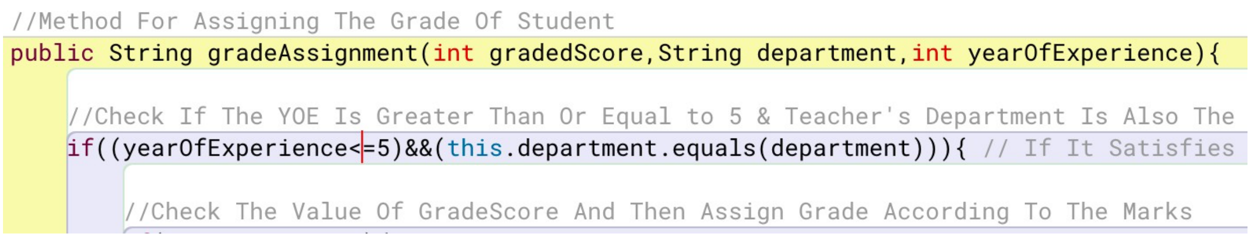
Figure 27:Error Detection and Error Removal:Solution2

Logical Error

A **logic error** is a condition encountered by a computer program where a result is not logically correct, but is not reported as an error. A program experiencing a logic error is especially dangerous, because it may report erroneous results as true. A **logic error** will not cause the program to crash, so the error is difficult for its developers to identify and resolve. (Computer Hope, 2018)

Error Detection :

The logical error that I encountered was in the gradeAssignment method. Instead of increasing writing greater than or equal to (>) I had written less than or equal to (<) which was creating an issue in the output.

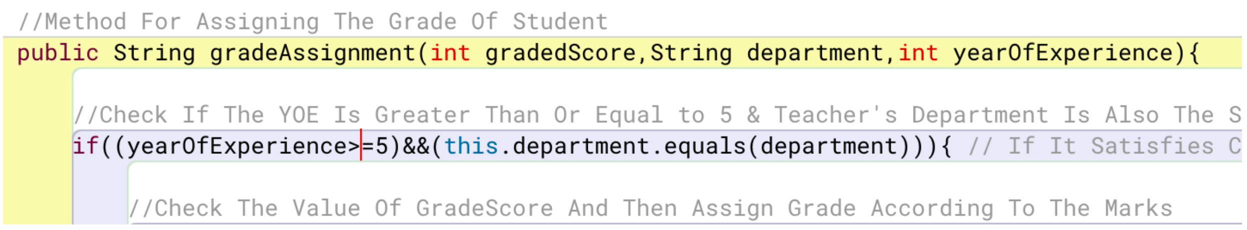


```
//Method For Assigning The Grade Of Student
public String gradeAssignment(int gradedScore,String department,int yearOfExperience){

    //Check If The YOE Is Greater Than Or Equal to 5 & Teacher's Department Is Also The
    if((yearOfExperience<=5)&&(this.department.equals(department))){ // If It Satisfies

        //Check The Value Of GradeScore And Then Assign Grade According To The Marks
```

Figure 28:Error Detection and Error Removal:Error3

Error Removal :

The error then was fixed when I Replaced (<) with (>) as shown in the screenshot :



```
//Method For Assigning The Grade Of Student
public String gradeAssignment(int gradedScore,String department,int yearOfExperience){

    //Check If The YOE Is Greater Than Or Equal to 5 & Teacher's Department Is Also The S
    if((yearOfExperience>=5)&&(this.department.equals(department))){ // If It Satisfies C

        //Check The Value Of GradeScore And Then Assign Grade According To The Marks
```

Figure 29:Error Detection and Error Removal:Solution3

Conclusion

In this coursework, We were centred with creating different classes and assigning suitable following the OOPs concept. We explored the OOPs concept of hierarchy which was a quite new and interesting experience .We had to build three classes i.e. Teacher ,Lecturer and Tutor which followed the hierarchy concept of OOPs in which the teacher was the parent class while rest of the classes were of child class.

We applied the OOPs concept to create a sound system that can manage the records of the teacher, lecturer and tutor records in a college system. We defined the attribute and methods for each classes which really helped a lot to get familiar with different thing like creating a constructor, getter method and setter method ,different datatypes and variables and many more.

I encountered various difficulties while creating object as the result didn't came out as I expected it be but however after hours of understanding the logic behind all the methods I have created in my project , I figured out the solution and was able to solve it.

This Coursework helped me to think in a more rigorous manner which really helped me to develop problem solving attitude that was new to me.This will really benefit me in my future programming journey .It also introduced me to the OOPs concept for the first time in my life .The best thing however I learned was to plan before starting coding. This helped me to make few errors and enjoy coding more as I faced fewer errors and frustrations as coding is one of the very first thing to get demotivated after making errors.

I really appreciate our teacher Mr. Saroj Kumar Yadav and Mr Pramodh Tuladhar for helping me and guiding me in this course work and also helping me debug some of my problem in my coding section which I would be thankful for my entire life.

Completing the project boosted my confidence in myself. It was not really a complete real life project but was a pretty huge step to some small real life implementation of what we have learned , but it was a valuable learning experience that demonstrated me the potential of this understanding in future .

Bibliography

GeeksforGeeks, 2022. *Introduction To Java-GeeksforGeeks.* [Online]
Available at: https://www.geeksforgeeks.org/introduction-to-java/
[Accessed 25 1 2024].
GeeksforGeeks, 2023. *Introduction to BlueJ-GeeksforGeeks.* [Online]
Available at: https://www.geeksforgeeks.org/introduction-of-bluej/
[Accessed 25 1 2024].
Computer Hope, 2020. *What is draw.io?.* [Online]
Available at: https://www.computerhope.com/jargon/d/drawio.htm
[Accessed 25 1 2024].
GeeksforGeeks, 2021. *Introduction to Microsoft - GeeksforGeeks.* [Online]
Available at: https://www.geeksforgeeks.org/introduction-to-microsoft-word/
[Accessed 25 1 2024].
GeeksforGeeks, 2024. *Class Diagram | Unified Modeling Language (UML) - GeeksforGeeks.* [Online]
Available at: https://www.geeksforgeeks.org/unified-modeling-language-uml-class-diagrams/
[Accessed 25 1 2024].
Technopedia, 2023. *What is pseudocode? | Definition from TechTarget.* [Online]
Available at:
https://www.techtarget.com/whatis/definition/pseudocode#:~:text=Pseudocode%2520is%252
[Accessed 25 1 2024].
Technopedia, 2023. *What is a Syntax Error ? - Definition from Technopedia.* [Online]
Available at: https://www.techopedia.com/definition/13391/syntax-error
[Accessed 25 1 2024].
Computer Hope, 2018. *What is a logic error?.* [Online]
Available at: https://www.computerhope.com/jargon/l/logierro.htm
[Accessed 25 1 2024].

Appendix:

Source Code For Teacher

```
public class Teacher

{

    //Decleration Of Instance Variables

    private int teacher_id;

    private String name;

    private String address;

    private String working_type;

    private String employment_status;

    private int working_hours;


    //Constructor Creation For Assigning The Values Of Instance Variable

    public Teacher(int teacher_id,String name,String address,String working_type,String
employment_status)

    {


        this.teacher_id=teacher_id;

        this.name=name;

        this.address=address;
        this.working_type=working_type;

        this.employment_status=employment_status;


    }




    //Getter Method to Access All The Instance Variable in Child Classes

    public int getteacher_id()

    {

        return this.teacher_id;

    }


    public int getworking_hours()

    {

        return this.working_hours;

    }


    public String getaddress()

    {

        return this.address;

    }


    public String getworking_type()
    {
        return this.working_type;
    }


    public String getemployment_status()
    {
        return this.employment_status;
    }


    public String getname()
    {
        return this.name;
    }
    //Setter Method For Setting The Value Of Instance Variable i.e. (working_hours)
    public void setWorkingHour(int working_hours)
    {
        this.working_hours=working_hours;
    }



    //Display Method To Display All The Values Of Instance Variables With Suitable
Annotation
    public void display()
```

```
   {


      System.out.println("Teacher_ID = "+this.getteacher_id());

      System.out.println("Teacher_Name = "+this.getname());

      System.out.println("Teacher_Address = "+this.getaddress());

      System.out.println("Working Type = "+this.getworking_type());

      System.out.println("Employee_Status = "+this.getemployment_status());


   //Conditions For Cheaking If Working Hours Has Been Set Or Not

   if(working_hours==0) //If Working Hour Has Been Set Then Print Out The Set
Working Hour With Suitable Annotation

   {

      System.out.println("Working Hour Has Not Been Set Yet");


   }

   else

   {  //If Working Hour Has Been Set Then Print Out The Set Working Hour With
Suitable Annotation

      System.out.println("Working_Hour = " +this.getworking_hours());

   }

  }

}
```

Source code for Lecturer Class :

```
public class Lecturer extends Teacher

{

    //Declaring Instance Variables

    private String department;

    private int yearOfExperience;

    private int gradedScore=0;

    private boolean hasGraded=false;


    // Constructor Creation For Assigning The Values Of Instance Variable

    public Lecturer(int teacher_id,String name,String address,String working_type,String
employment_status,int working_hours,String department, int yearOfExperience)

{


        super(teacher_id,name,address,working_type,employment_status);

        super.setWorkingHour(working_hours);

        this.department=department;

        this.yearOfExperience=yearOfExperience;


}
```

```
//Getter Method to Access All The Instance Variable
public String getdepartment()
{
    return this.department;
}

public int getyearOfExperience()
{
    return this.yearOfExperience;
}

public int getgradedScore()
{
    return this.gradedScore;
}

 public boolean gethasGraded()
{
    return this.hasGraded;
}



//Setter Method For Setting The Value Of Instance Variable i.e. (gradedScore)
```

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



    //Method For Assigning The Grade Of Student
    public String gradeAssignment(int gradedScore,String department,int
yearOfExperience)
    {

        //Check If The YOE Is Greater Than Or Equal to 5 & Teacher's Department Is Also
The Same
        if((yearOfExperience>=5)&&(this.department.equals(department))) // If It Satisfies
Condition Then Do
        {

            //Check The Value Of GradeScore And Then Assign Grade According To The
Marks
            if(gradedScore>70)
            {
                this.gradedScore=gradedScore;
                this.hasGraded=true;//Assign true To hasGraded After Grade Assignment
               return "Grade A";
```

```
}
else if(gradedScore >60)
{
     this.gradedScore=gradedScore;
     this.hasGraded=true;//Assign true To hasGraded After Grade Assignment
    return "Grade B";
}
else if(gradedScore >50)
{
     this.gradedScore=gradedScore;
     this.hasGraded=true;//Assign true To hasGraded After Grade Assignment
    return "Grade C";
}
else if(gradedScore >40)
{
     this.gradedScore=gradedScore;
     this.hasGraded=true;//Assign true To hasGraded After Grade Assignment
    return "Grade D";
}
else if(gradedScore<40)
{
     this.gradedScore=gradedScore;
     this.hasGraded=true;//Assign true To hasGraded After Grade Assignment
```

```
            return  "Grade E";
         }

      }



      //If YOE And Teacher's Assignment Doesn't Match
      return "Teacher Not Eligible For Grade Assignment";

}



      //Display Method To Display The Details Of The Lecturer
      public void display()
      {
      super.display();//Calling Method from Parent to Child

      System.out.println("Department = "+getdepartment());
      System.out.println("Year Of Experience = "+getyearOfExperience());


      //Checking If Grading Score Has Been Set Or Not
      if(this.gradedScore==0) //If Its Not Set Display A Suitable Message
```

```
        {
            System.out.println("Graded Score Not Assigned  ");
        }
        else//If Its Set Already Just Print The Output With Suitable Notation
        {
        System.out.println("Graded Score = "+getgradedScore());
    }
}
}
```

Source code for Tutor Class :

```
public class Tutor extends Teacher
{
    //Declaring Instance Variables
    private double salary;
    private String specialization;
    private String academic_qualifications;
    private int performance_index;
    private boolean isCertified=false;

     // Constructor Creation For Assigning The Values Of Instance Variable
    public Tutor(int teacher_id, String name, String address, String working_type, String
employment_status,int working_hours, double salary, String specialization, String
academic_qualifications,int performance_index)
    {
        super(teacher_id, name, address, working_type, employment_status);
        super.setWorkingHour(working_hours);
        this.salary = salary;
        this.specialization = specialization;
        this.academic_qualifications = academic_qualifications;
        this.performance_index = performance_index;


    }

    //Getter Method to Access All The Instance Variable
     public double getsalary()
```

```
{
     return this.salary;
}
public String getspecialization()
{
    return this.specialization;
}
public String getacademic_qualifications()
{
    return this.academic_qualifications;
}
public int getperformance_index()
{
    return this.performance_index;
}
public boolean getisCertified()
{
    return this.isCertified;
}



//Setter Method For Setting The Value Of Instance Variable i.e. (salary) After Giving
Appraisal As Per Their Work
public void setsalary(int salary,int performance_index)
{

    if((this.performance_index>5)&&(this.getworking_hours()>20))
    {
        if((performance_index>=5)&&(performance_index<=7))
        {
            this.salary=salary+((double)5/100)*salary;
        }
        else if((performance_index>=8)&&(performance_index<=9))
        {
            this.salary=salary+((double)10/100)*salary;
        }
        else if (performance_index==10)
        {
            this.salary=salary+((double)20/100)*salary;
        }
        this.isCertified=true;

    }
    else
    {
        System.out.println("Salary Has Not Been Approved");
    }



}

//Method For RemovingTutor If Tutor Has Not Been Certified Yet
public void removetutor()
{

if(isCertified==false)  //Check The Value Of (isCertified) If Its False Then Assign
Values To The Tutor
{
//Assigning the Values Of The Instance Variable To None And Zero
 this.salary=0d;
 this.specialization="";
 this.academic_qualifications="";
 this.performance_index=0;

 this.isCertified=false; //Assigning The Value Of isCertified To False
    System.out.println("Tutor Removed Successfully"); // Displaying A Suitable
Message If Tutor Is Removed
}
else
{

    System.out.println("Tutor Was Not Removed");//Displaying A Suitable Message To
Convey That Tutor Was Not Removed
}
}
//Method Overwriting To Display With Suitable Annotation
public void display()
{
    if(isCertified)
    {
        //Calling Display Method Of Teacher To Display The Information Of Teacher
        super.display();

        //Display Additional Information
        System.out.println("Salary = "+this.getsalary());
        System.out.println("Specialization = "+this.getspecialization());
```

```
            System.out.println("Academic Qualification = 
"+this.getacademic_qualifications());
            System.out.println("Performance Index = "+this.getperformance_index());



        }
        else
        {
            super.display(); //Display Information Of Teacher
        }
    }

}
```