CS4001NI Programming

30% Individual Coursework

2023-24 Autumn

Student Name: Ishpa Maharjan

London Met ID: 23047616

College ID: NP01CP4A230045

Group: C2

Assignment Due Date: Friday, January 26, 2024

Assignment Submission Date: Friday, January 26, 2024

Table of contents

Contents

Tables of figures

Table of tables

1.Introduction

1.1 About the coursework

JAVA

Java is a multi-platform, object oriented, case sensitive programming language used for coding web applications. It is widely used by millions of developers because it is free to use and versatile programming language. Java is mostly used for game development, cloud computing, big data, artificial importance, IOT. Developers choose Java out of other programming languages because it includes; High quality learning resources, inbuilt functions of libraries, Active community support, High quality development tools, Platform independent, Security and many more (aws, 2023).

1.2 Tools used in the coursework

Some of the tools used are:

1. BLUEJ:

BlueJ is an integrated development languages for the Java programming language which is mostly used for the educational purposes. It is easier to work for the small-scale software development that runs with the help of java development kit (JDK). BlueJ is used for the learning and teaching of object-oriented programming languages. In BlueJ we can see the main screen which shows class structure. It includes object-oriented concepts such as class and objects communicate through method calls (Wales, 2023).

Figure 1

2. Ms-Word:

Ms-Word is one of the widely used programs of Microsoft offices, it is a word processor developed by Microsoft. It is used to make professional quality documents, letters, reports, etc. Ms-Word allows us to format, edit our files and documents in the best possible ways. It is used in education, workplace, for authors, etc (BYJU'S, 2024).

Figure 2

2. Class Diagram

2.1 Teacher class

Table 1

Teacher
+teacherID: int +teachername: String +address: String +workingtype: String +emploument_status: String +working_hours: int
+<<constructor>>Teacher(teacherID: int, teachername: String, address: String, workingtype: String, emploument_status: String, working_hours: int) +getteacherID(): int +getteachername(): String +getaddress(): String +getworkingtype(): String +getemploument_status(): String +getworking_hours(): int +setworking_hours(working_hours: int): void +display: void

2.2 Lecturer class

Table 2

Lecturer
-Department: String -YearsOfExperience: int -gradedScore: int -hasGraded: boolean
+<<constructor>>Lecturer(teacherID: int, teachername: String, address: String, workingtype: String, working_hours: String, employment_status: String, Department: String, YearsOfExperience: int) +getDepartment(): String +getYearsOfExperience(): int +getGradedScore(): int +hasGraded(): boolean +setGradedScore(gradedscore: int): void +display: void

2.3 Tutor class

Table 3

Tutor
-salary: double
-specialization: String
-academic_qualifications: String
-performanceIndex: int
-isCertified: boolean
+<<constructor>>Tutor(teacherID: int, teachername: String, address: String, workingtype: String, employment_status: String, working_hours: int, salary: double, specialization: String, academic_qualifications: String, performanceIndex: int)
+getSalary(): double
+getSpecialization(): String
+getAcademicQualifications(): String
+getPerformanceIndex(): String
+isCertified(): boolean
+setSalary(salary: double): void
+setperformnceIndex(performanceIndex: int): void
+removeTutor(): void
+display: void

2.4 Combined class diagram

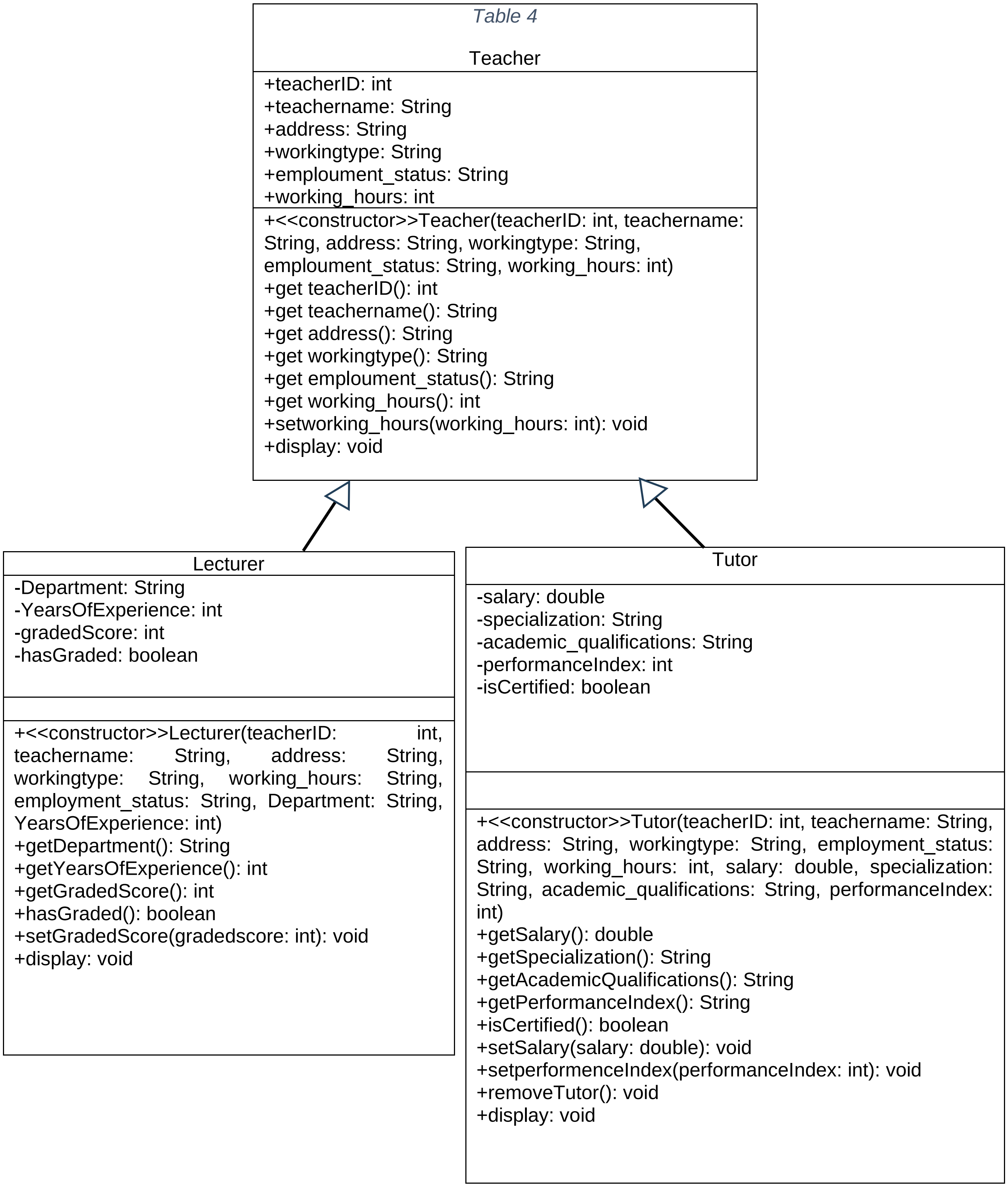


Table 4
Teacher
+teacherID: int
+teachername: String
+address: String
+workingtype: String
+emploument_status: String
+working_hours: int
+<<constructor>>Teacher(teacherID: int, teachername: String, address: String, workingtype: String, emploument_status: String, working_hours: int)
+get teacherID(): int
+get teachername(): String
+get address(): String
+get workingtype(): String
+get emploument_status(): String
+get working_hours(): int
+setworking_hours(working_hours: int): void
+display: void
Lecturer
-Department: String
-YearsOfExperience: int
-gradedScore: int
-hasGraded: boolean
+<<constructor>>Lecturer(teacherID: int, teachername: String, address: String, workingtype: String, working_hours: String, employment_status: String, Department: String, YearsOfExperience: int)
+getDepartment(): String
+getYearsOfExperience(): int
+getGradedScore(): int
+hasGraded(): boolean
+setGradedScore(gradedscore: int): void
+display: void
Tutor
-salary: double
-specialization: String
-academic_qualifications: String
-performanceIndex: int
-isCertified: boolean
+<<constructor>>Tutor(teacherID: int, teachername: String, address: String, workingtype: String, employment_status: String, working_hours: int, salary: double, specialization: String, academic_qualifications: String, performanceIndex: int)
+getSalary(): double
+getSpecialization(): String
+getAcademicQualifications(): String
+getPerformanceIndex(): String
+isCertified(): boolean
+setSalary(salary: double): void
+setperformenceIndex(performanceIndex: int): void
+removeTutor(): void
+display: void

3. Pseudocode

3.1 Teacher class

```
CREATE a parent class Teacher
DO
        DECLARE instance variable teacherID as int using private specifier
        DECLARE instance variable teachername as int using private specifier
        DECLARE instance variable address as int using private specifier
        DECLARE instance variable workingtype as int using private specifier
        DECLARE instance variable employment_status as int using private specifier
        DECLARE instance variable working_hours as int using private specifier


        CREATE a constructor of Teacher(int teacherID, String teachername, String
        address, String workingtype, String employment_status)
         DO
                ASSIGN this keyword to store local variable teacherID in instance variable
                teacherID
                ASSIGN this keyword to store local variable teachername in instance
                variable teachername
                ASSIGN this keyword to store local variable address in instance variable
                address
                ASSIGN this keyword to store local variable workingtype in instance
                variable workingtype
```

```
          ASSIGN this keyword to store local variable employment_status in instance
          variable employment_status
          ASSIGN this keyword to store local variable working_hours in instance
          variable working_hours
      END DO


CREATE an accessor method teacherID() with return type int
  Do
    RETURN teacherID
  END DO
CREATE an accessor method teachername() with return type String
  DO
    RETURN teachername
  END DO
CREATE an accessor method address() with return type String
  DO
    RETURN address;
  END DO
CREATE an accessor method workingtype() with return type String
  DO
    RETURN workingtype;
  END DO
```

```
CREATE an accessor method employment_status() with return type String
  DO
      RETURN employment_status;
   END DO
CREATE an accessor method working_hours() with return type int
  DO
      RETURN working_hours;
   END DO



CREATE an mutator method working_hours(working_hours) with non-return type int
   DO
        ASSIGN this keyword to set working_hours
   END DO



   CREATE a method to displayTeacher()
   DO
      DISPLAY(The teacher ID)
      DISPLAY(The teacher name)
      DISPLAY(The address)
      DISPLAY(The workingtype)
   DISPLAY(The Employment Status)




    IF working_hours  is greater than 0
        DO
             DISPLAY ( Working Hours)
        END DO
    ELSE
        DO
             DISPLAY (Working Hours is Not assigned)
        END DO
    IF END

END DO
```

3.2 Lecturer class

```
CREATE a child class teacher
DO
    DECLARE instance variable Department as String using private specifier
    DECLARE instance variable YearsOfExperience as int using private specifier
    DECLARE instance variable gradedscore as int using private specifier
    DECLARE instance variable hasgraded as boolean using private specifier


    CREATE a constructor of lecturer(int teacherID, String teachername, String
    address, String workingtype, int working_hours, String employment_status,
    String Department, int YearsOfExperience)
    DO
        ASSIGN super(teacherID, teachername, address, workingtype,
        employment_status) keyword to call super class constructor
        SET working_hours(working_hours) as a setter method from parent class
        ASSIGN this keyword to store local variable Department in instance
        variable Department
        ASSIGN this keyword to store local variable YearsOfExperience in
        instance variable YearsOfExperience
        ASSIGN gradedscore as zero by passing default value
        ASSIGN hasgraded as false by passing default value
 END DO
```

```
CREATE an accessor method Department() with return type String
   DO
      RETURN Department
   END DO

 CREATE an accessor method YearsOfExperience() with return type int
   DO
      RETURN YearsOfExperience
   END DO

CREATE an accessor method GradedScore() with return type int
   DO
      RETURN gradedscore
   END DO

CREATE an accessor method hasgraded() with return type boolean
   DO
      RETURN hasGraded;
   END DO

   CREATE an mutator method GradedScore(gradedscore) with non-return type int
```

```
  DO
       ASSIGN this keyword to set gradedscore
  END DO



CREATE a method to grade assignment(int GradedScore, String Department, int
YearsOfExperience)   with non-return type
  DO
       IF hasGraded is false
       IF YearsOfExperience is greater than or equal to 5 and department is also
       relevant to the department with same area of interest
       DO
               IF gradedScore is greater than or equal to 70
                       DO
                               DISPLAY (The grade is A)
                       END DO
               ELSE IF gradedScore is greater than or equal to 60
                       DO
                               DISPLAY (The grade is B)
                       END DO
               ELSE IF gradedScore is greater than or equal to 50
                       DO
                               DISPLAY (The grade is C)
```

```
                END DO
            ELSE IF gradedScore is greater than or equal to 40
                DO
                    DISPLAY (The grade is D)
                END DO
            ELSE
                DO
                    DISPLAY (The grade is E)
                END DO
            hasGraded=true
        END DO
            ELSE
                DO
                    DISPLAY (The assignment is already graded.)
                 END DO
            END IF
            END IF
    END DO
```

```
CREATE a method to displayLecturer()
   DO
      ASSIGN super keyword to displayTeacher() from the parent class
      DISPLAY(The Department)
      DISPLAY (The Years Of Experience)
      IF hasgraded
        DO
                DISPLAY(The Graded Score)
         END DO
      ELSE
        DO
                 DISPLAY(The Score has not been Graded)
        END DO
        END IF
    END DO

END DO
```

3.3 Tutor class

```
CREATE a child class Tutor
DO
    DECLARE instance variable salary as double using private specifier
    DECLARE instance variable Specialization as String using private specifier
    DECLARE instance variable salary as double using private specifier
    DECLARE instance variable academic_qualifications as String using private
    specifier
    DECLARE instance variable performanceIndex as int using private specifier
    DECLARE instance variable isCertified as boolean using private specifier


    CREATE a constructor Tutor(int teacherID, String teachername, String address,
    String workingtype, String employment_status, int working_hours, double salary,
    String specialization, String academic_qualifications, int performanceIndex)
    DO
        ASSIGN super(teacherID, teachername, address, workingtype,
        employment_status) keyword to call super class constructor
        SET working_hours(working_hours) as a setter method from parent class
        ASSIGN this keyword to store local variable salary in instance variable
        salary
        ASSIGN this keyword to store local variable specialization in instance
        variable specialization
        ASSIGN this keyword to store local variable academic_qualifications in
        instance variable academic_qualifications
        ASSIGN this keyword to store local variable performanceIndex in instance
        variable performanceIndex
        ASSIGN this keyword to store false as a default value in instance variable
        isCertified
    END DO

CREATE an accessor method salary() with return type double
    DO
        RETURN salary
    END DO

CREATE an accessor method specialization() with return type String
    DO
        RETURN specialization
    END DO

CREATE an accessor method academic_qualifications() with return type String
    DO
        RETURN academic_qualifications
    END DO
```

```
CREATE an accessor method performanceIndex() with return type int
    DO
        RETURN performanceIndex
    END DO

CREATE an accessor method isCertified() with return type boolean
    DO
        RETURN isCertified
    END DO

CREATE a mutator method salary(salary) with non-return type double
    DO
        ASSIGN this keyword to set salary
    END DO

CREATE a mutator method PerformanceIndex(PerformanceIndex) with non-return type
double
    DO
        ASSIGN this keyword to set PerformanceIndex
    END DO
```

```
CREATE a method to set salary(double newSalary, int newPerformanceIndex)
DO
    IF (isCertified is false)
    IF(performanceIndex is greater than 5 and getworking_hours() is greater than 20)
    DO
        double appraisalPercentage
            IF (performanceIndex is greater than or equal to 5 and
            performanceIndex is less than or equal to 7)
            DO
                appraisalPercentage is 0.05
            END DO
            ELSE IF (performanceIndex is greater than or equal to 8 and
            performanceIndex is less than or equal to 9)
            DO
                appraisalPercentage is 0.10
            END DO
            ELSE IF (performanceIndex is equal to 10)
            DO
                appraisalPercentage is 0.20
            END DO
            ELSE
            DO
                appraisalPercentage is 0.0
            END DO
            END IF




        double appraisal is equal to the product of salary and appraisalPercentage
        ASSIGN this keyword and store in instance variable of salary increased by
        certain percentage
        ASSIGN this keyword and store in instance variable of performanceIndex
        in newPercentage
        isCertified is true
            DISPLAY(salary is successfully approved)
    END DO

        ELSE
        DO
            DISPLAY(Salary cannot be approved for the tutor)
        END DO
    END IF
```

```
        ELSE
        DO
            DISPLAY(Salary cannot be changed for a certified tutor)
        END DO
END DO
END IF



CREATE a method removeTutor() to remove the method
DO
    IF isCertified is false
    DO
        ASSIGN this keyword to set salary as zero
        ASSIGN this keyword to set specialization as zero
        ASSIGN this keyword to set academic_qualifications as zero
        ASSIGN this keyword to set performanceIndex as zero
        ASSIGN this keyword to set isCertified as false
        DISPLAY(The Tutor is successfully removed)
    END DO
    ELSE
    DO
        DISPLAY(Certified Tutor cannot be removed)
    END DO
    END IF
END DO



CREATE a method display with non-return type
DO
    IF isCertified
    DO
        DISPLAY(The salary of a tutor)
        DISPLAY(The specialization of a tutor)
        DISPLAY(The academic qualifications of a tutor)
        DISPLAY(The performance Index of a tutor)
    END DO
    ELSE
    DO
        ASSIGN super keyword to display displayTeacher() from parent class
    END DO
    END IF
END DO
END DO
```

4. Description of Methods

4.1 Teacher class

- Accessor(getter) method

Table 5

Accessor method	What does it do?
get teacherID()	This method returns the attribute teacherID.
get teachername()	This method returns the attribute teachername.
get address()	This method returns the attribute address.
get workingtype()	This method returns the attribute workingtype.
get employment_status()	This method returns the attribute employment_status.
get working_hours()	This method returns the attribute working_hours.

- Mutator(setter) method

Table 6

Mutator method	What does it do?
setworking_hours(int working_hours)	This method is a non-return type method that sets the working hours in int.

- Display method

This method displays all the attributes of Teacher class with non-return type displayTeacher() which gives teacherID, teachername, address, workingtype, employment_status, working_hours.

4.2 Lecturer class

- Accessor(getter) method

Table 7

Accessor method	What does it do?
get Department()	This method returns the attribute Department.
get YearsOfExperience()	This method returns the attribute YearsOfExperience.
get GradedScore()	This method returns the attribute GradedScore.
get hasGraded()	This method returns the attribute hasGraded.

- Mutator(setter) method

Table 8

Mutator method	What does it do?
set GradedScore(int gradedScore)	This method is a non-return type method that sets the Graded Score in int.

- Method to grade assignment

This method is a non-return type method that grades the assignment according to their graded score. In this method we have if else statement for years of experience which must be greater than or equal to five and department is also relevant to the department with same area of interest to grade the assignment. In this method hasGraded is set to false for assigning grades but if hasGraded is true it displays the assignment is already graded.

- Display method

This method displays all the attributes of Lecturer class with non-return type displayLecturer() which displays Department, Years of experience and also calls the display method from teacher class by super keyword. It also includes if else statement in this method to Grade Score.

4.3 Tutor class

Table 9

- Accessor(getter) method

Accessor method	What does it do?
get Salary()	This method returns the attribute Department.
get Specialization()	This method returns the attribute YearsOfExperience.
get AcademicQualifications()	This method returns the attribute GradedScore.
get PerformanceIndex()	This method returns the attribute hasGraded.
get isCertified()	This method returns the attribute isCertified.

- Mutator(setter) method

Table 10

Mutator method	What does it do?
set Salary(double salary)	This method is a non-return type method that sets the Salary in double.
set PerformanceIndex(int performanceIndex)	This method is a non-return type method that sets the performanceIndex in int.

- Method to set Salary

This method is a non-return type method that sets the Salary according to their performance index and working hours. In this method we have if else statement for performanceIndex that must be greater than five and working_hours that must be greater than 20 to set the salary. In this method isCertified is set to false to set the salary that displays the salary is successfully approved else the salary cannot be approved for the tutor but if isCertified is true it displays the salary cannot be changed for a certified tutor.

- Method to remove tutor

This method is a non-return type method for removing the tutor. It has if else statement for isCertified that displays Tutor is removed if it is false else it displays the tutor cannot be removed.

- Display method

This method displays all the attributes of Tutor class with non-return type displayTutor() which displays Salary, Specialization, Academic Qualifications, Performance Index and also calls the display method from teacher class by super keyword.

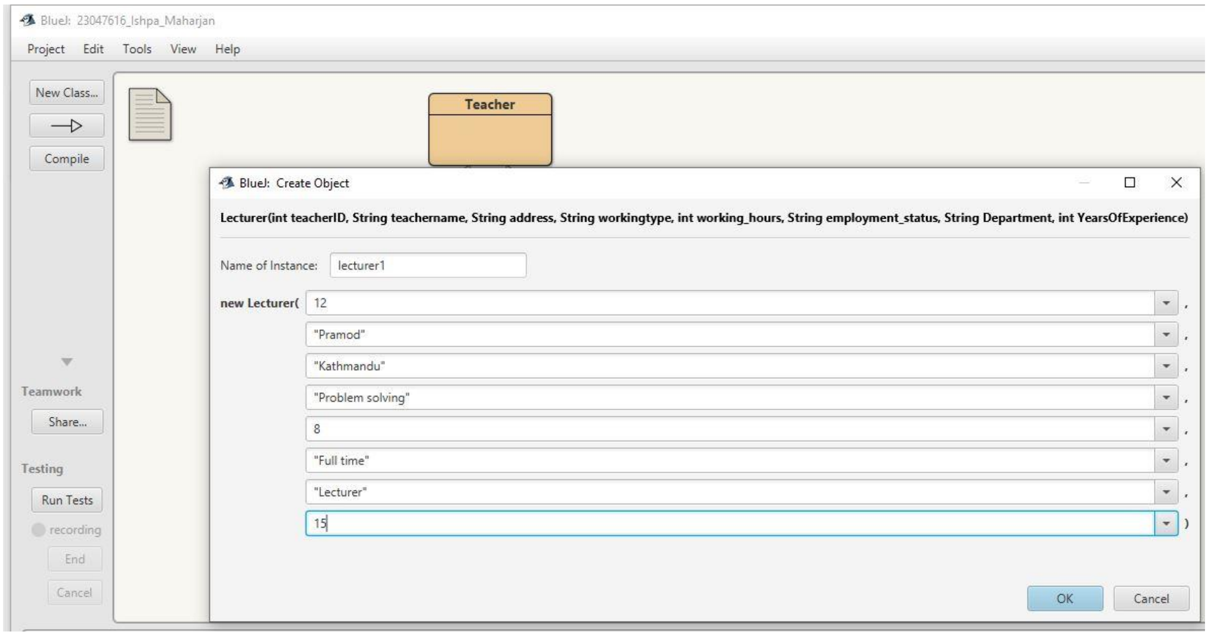
5.Testing

5.1 For Lecturer class

Table 11

Test	1
Objective	Inspect the Lecturer class, grade the assignment, and re-inspect the Lecturer Class
Action	➢ The lecturer is called with the following arguments: Lecturer ID: 12 Lecturer name: “Pramod” Address: “Kathmandu” Working type: ‘Problem solving” Working hours: 8 Employment status: “Full time” Department: “Lecture” Years of experience: 15 ➢ Inspection of the Lecturer class. ➢ void appointLecturer is called with the following arguments. Department: “Lecture” Years of experience: 15 gradedScore: 0 hasGraded: false teachername: “Pramod” address: “Kathmandu” workingtype: “Problem solving” employment_status: “Full time” working_hours: 8 ➢ Re-inspection of the Lecturer class Department: “Lecture” Years of experience: 15 gradedScore: 75 hasGraded: false teachername: “Pramod” address: “Kathmandu” workingtype: “Problem solving” employment_status: “Full time” working_hours: 8
Expected Result	The assignment would be graded successfully.
Actual Result	The assignment was graded.
Conclusion	The test is successful.

Output Result:

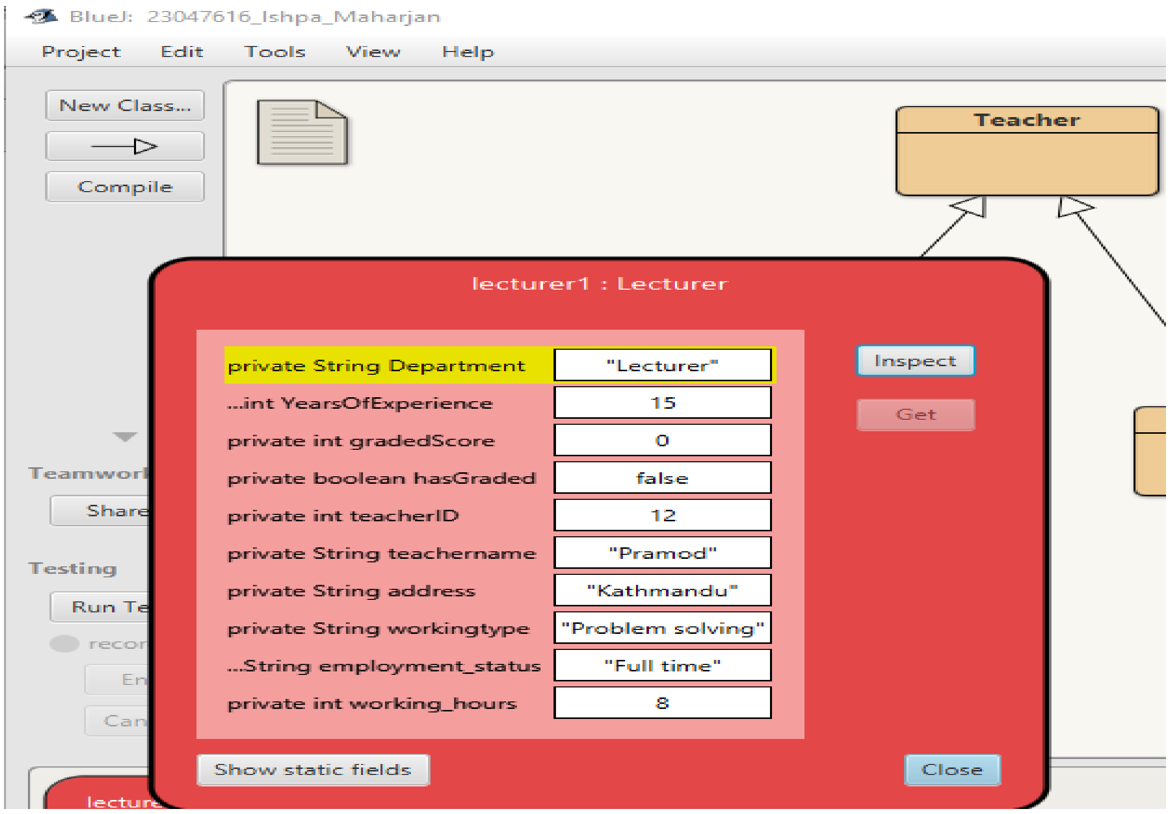


Lecturer(int teacherID, String teachername, String address, String workingtype, int working_hours, String employment_status, String Department, int YearsOfExperience)

Name of Instance: lecturer1

new Lecturer(12, "Pramod", "Kathmandu", "Problem solving", 8, "Full time", "Lecturer", 15)

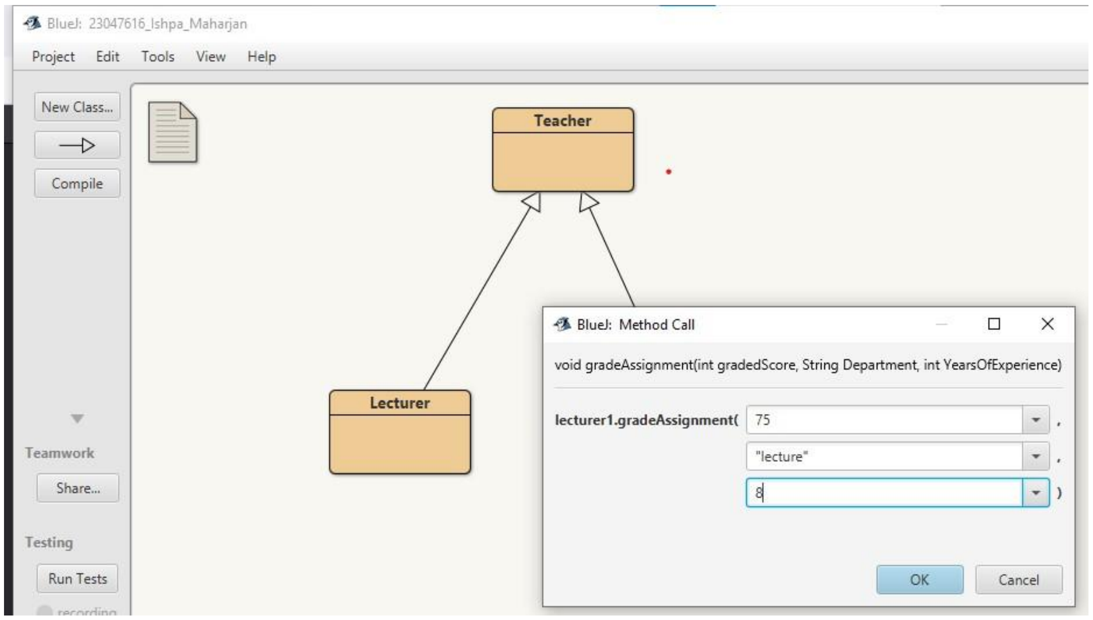
Figure 3: screenshot of assigning data in lecturer class



lecturer1 : Lecturer

private String Department	"Lecturer"
...int YearsOfExperience	15
private int gradedScore	0
private boolean hasGraded	false
private int teacherID	12
private String teachername	"Pramod"
private String address	"Kathmandu"
private String workingtype	"Problem solving"
...String employment_status	"Full time"
private int working_hours	8

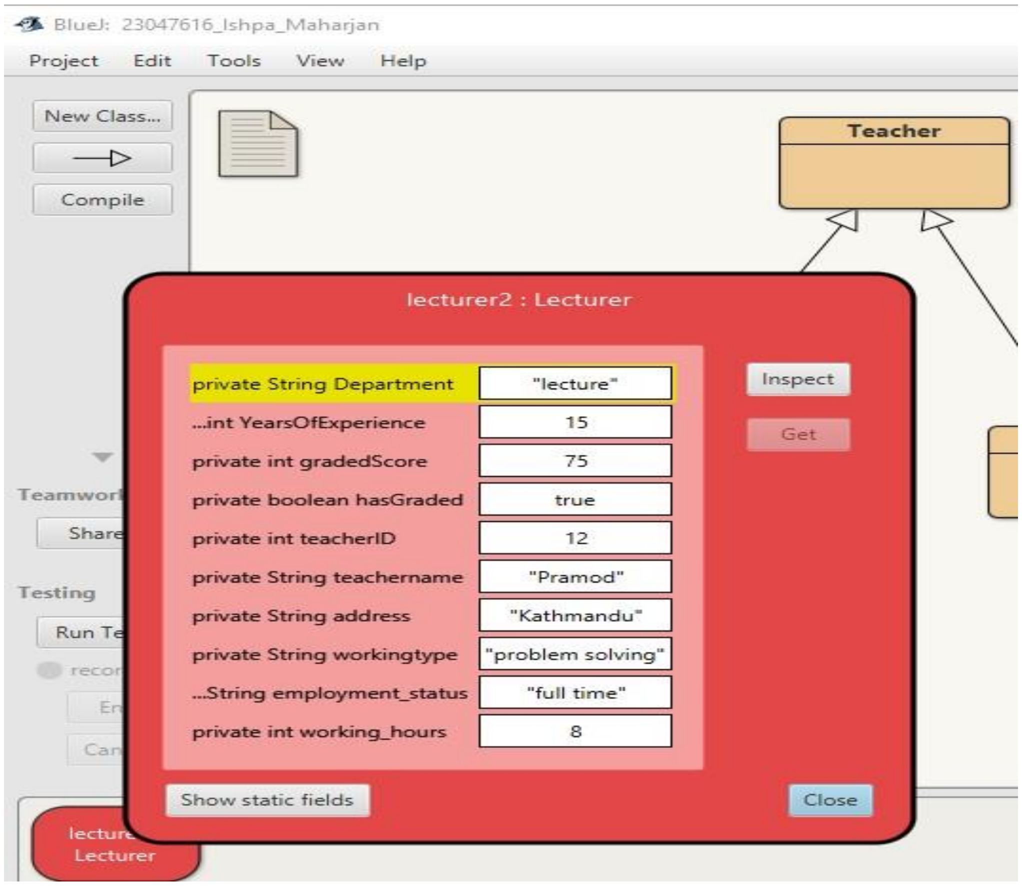
Figure 4: screenshot for the inspection of lecturer class



void gradeAssignment(int gradedScore, String Department, int YearsOfExperience)

lecturer1.gradeAssignment(75 , "lecture" , 8)

Figure 5: screenshot to set the graded score

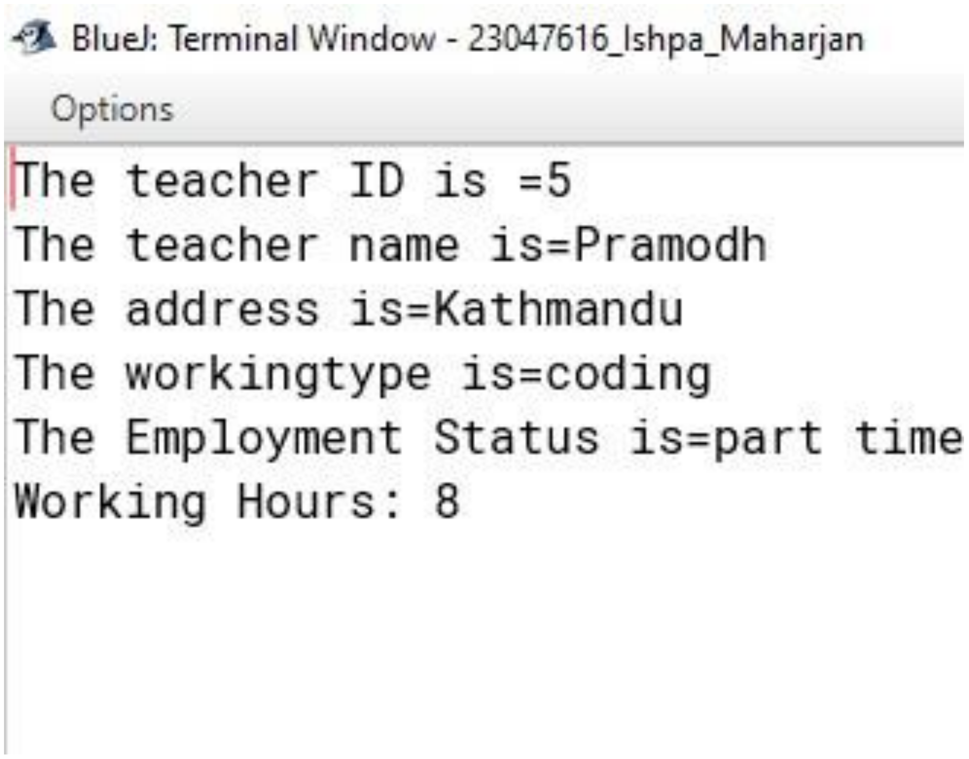


lecturer2 : Lecturer

private String Department	"lecture"
...int YearsOfExperience	15
private int gradedScore	75
private boolean hasGraded	true
private int teacherID	12
private String teachername	"Pramod"
private String address	"Kathmandu"
private String workingtype	"problem solving"
...String employment_status	"full time"
private int working_hours	8

Figure 6: screenshot for re-inspection of lecturer class

Output Result for Teacher class:



```
The teacher ID is =5
The teacher name is=Pramodh
The address is=Kathmandu
The workingtype is=coding
The Employment Status is=part time
Working Hours: 8
```

Figure 7: Output Result of Teacher class

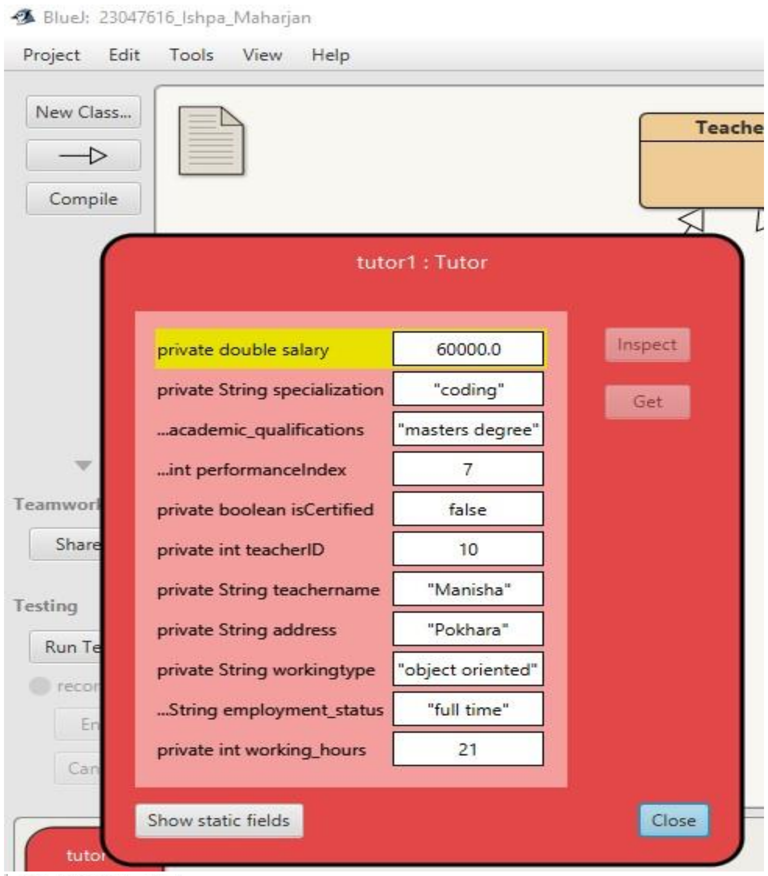
5.2 For Tutor class

Table 12

Test	2
Objective	Inspect the Tutor class, set the salary, and re-inspect the Tutor Class
Action	➢ The Tutor is called with the following arguments: Lecturer ID: 10 Lecturer name: “Manisha” Address: “Pokhara” Working type: “Object oriented” Working hours: 21 Salary: 60000 Department: “coding” Academic qualifications: “master’s degree” Performance Index: 7 ➢ Inspection of the Tutor class. ➢ void appointTutor is called with the following arguments: salary: 60000 specialization: “coding” academic qualifications: “master’s degree” performanceIndex: 7 isCertified: false teacher ID: 10 teachername: “Manisha” address: “Pokhara” working type: “object oriented” employment_status: “full time” working hours: 21 ➢ Re-inspection of the Tutor class salary: 83000 specialization: “coding” academic qualifications: “master’s degree” performanceIndex: 9 isCertified: true teacher ID: 10 teachername: “Manisha” address: “Pokhara” working type: “object oriented” employment_status: “full time” working hours: 21
Expected Result	The salary would be successfully approved.
Actual Result	The salary was approved.
Conclusion	The test is successful.

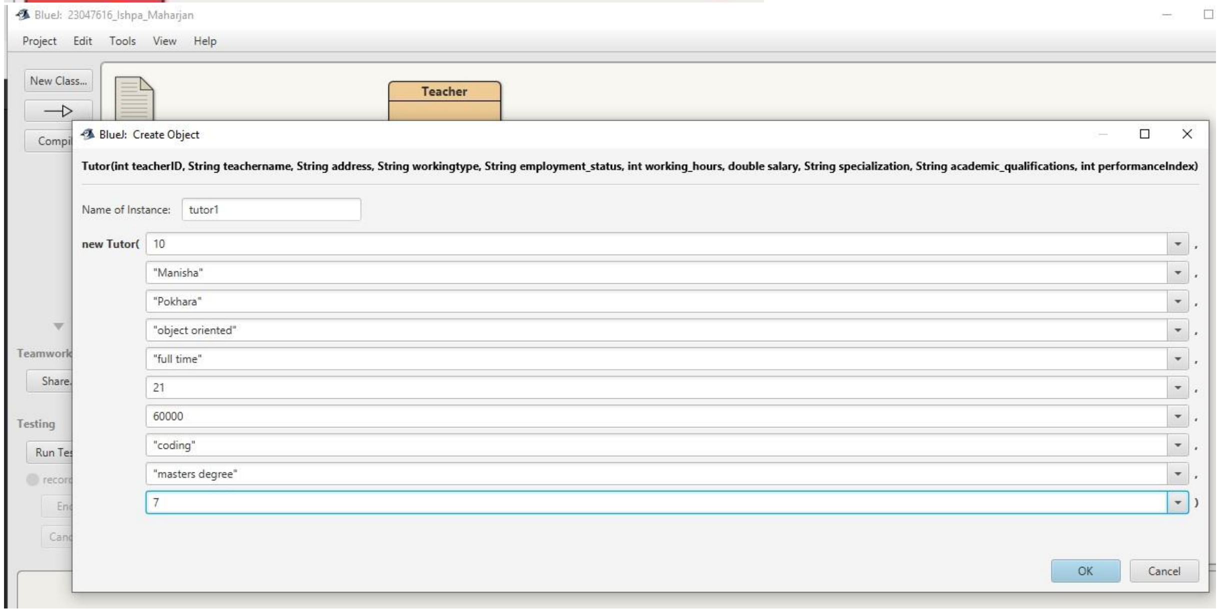
Output Result:

Figure 8: screenshot of assigning data in Tutor class



tutor1 : Tutor

private double salary	60000.0
private String specialization	"coding"
...academic_qualifications	"masters degree"
...int performanceIndex	7
private boolean isCertified	false
private int teacherID	10
private String teachername	"Manisha"
private String address	"Pokhara"
private String workingtype	"object oriented"
...String employment_status	"full time"
private int working_hours	21

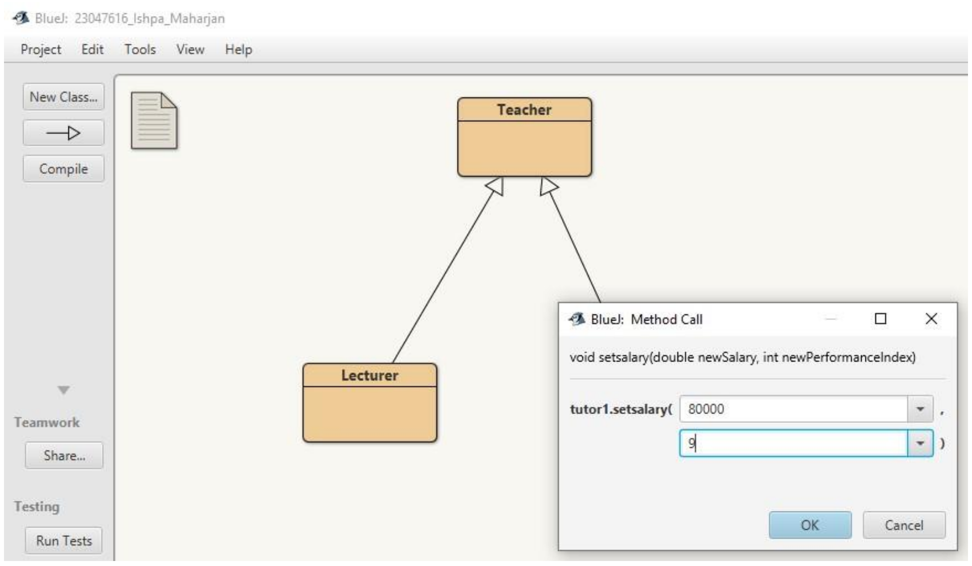


Tutor(int teacherID, String teachername, String address, String workingtype, String employment_status, int working_hours, double salary, String specialization, String academic_qualifications, int performanceIndex)

Name of Instance: tutor1

new Tutor(10 ,
"Manisha" ,
"Pokhara" ,
"object oriented" ,
"full time" ,
21 ,
60000 ,
"coding" ,
"masters degree" ,
7)

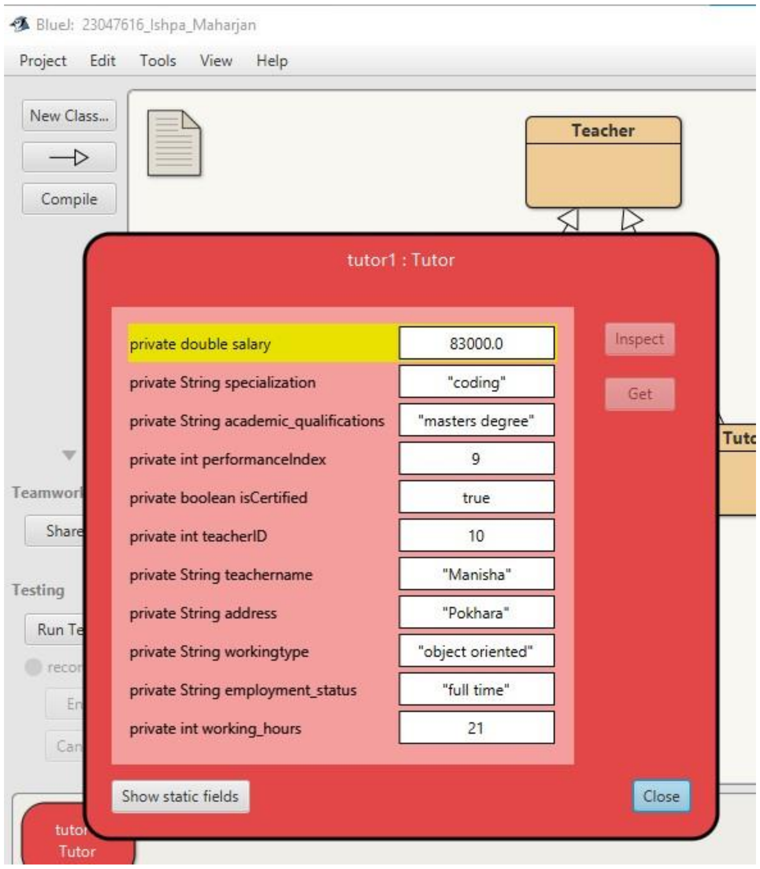
Figure 9: screenshot for the inspection of Tutor class



void setsalary(double newSalary, int newPerformanceIndex)

tutor1.setsalary(80000 , 9)

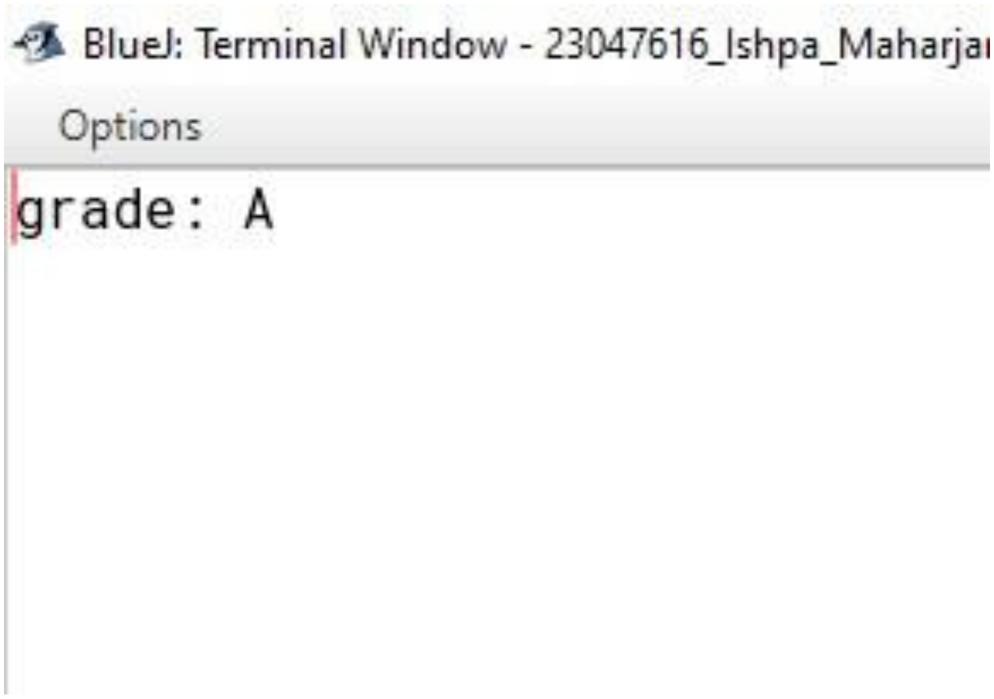
Figure 10: screenshot to set the salary and performance Index



private double salary	83000.0
private String specialization	"coding"
private String academic_qualifications	"masters degree"
private int performanceIndex	9
private boolean isCertified	true
private int teacherID	10
private String teachername	"Manisha"
private String address	"Pokhara"
private String workingtype	"object oriented"
private String employment_status	"full time"
private int working_hours	21

Figure 11: screenshot for re-inspection of Tutor class

Output Result of Lecturer class: grading score is set as 75



```
grade: A
```

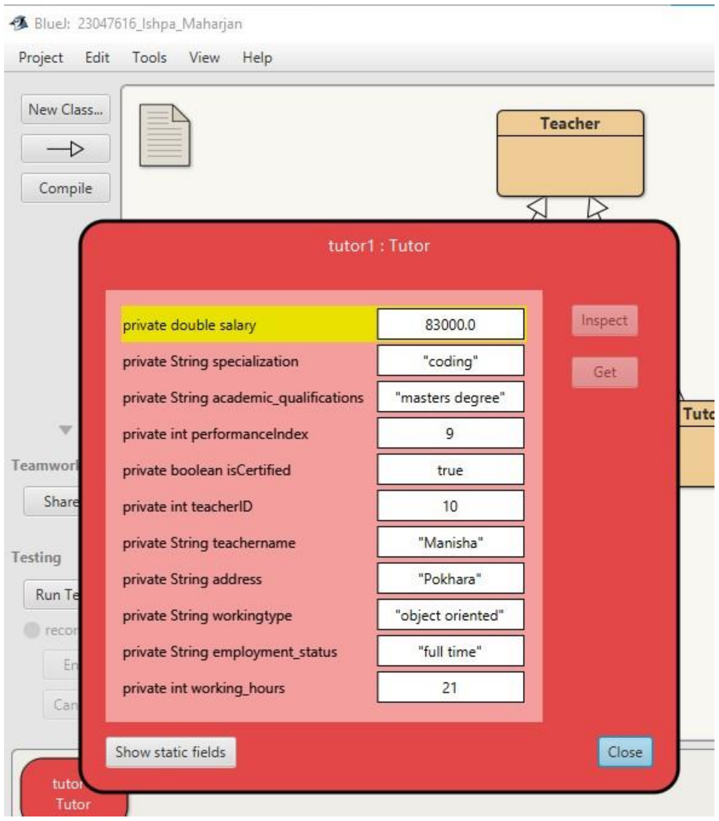
Figure 12: Output Result of Lecturer class

5.3 For Tutor class after removing tutor

Table 13

Test	3
Objective	Inspect the Tutor class, inspect the Tutor Class after removing tutor
Action	➢ The Tutor is called with the following arguments: Lecturer ID: 10 Lecturer name: “Manisha” Address: “Pokhara” Working type: “Object oriented” Working hours: 21 Salary: 60000 Department: “coding” Academic qualifications: “master’s degree” Performance Index: 7 ➢ Inspection of the Tutor class after removing tutor salary: 0.0 specialization: “” academic qualifications: “” performanceIndex: 0 isCertified: false teacher ID: 10 teachername: “Manisha” address: “Pokhara” working type: “object oriented” employment_status: “full time” working hours: 21
Expected Result	The tutor class would be successfully removed.
Actual Result	The Tutor class was removed.
Conclusion	The test is successful.

Output Result:



private double salary	83000.0
private String specialization	"coding"
private String academic_qualifications	"masters degree"
private int performanceIndex	9
private boolean isCertified	true
private int teacherID	10
private String teachername	"Manisha"
private String address	"Pokhara"
private String workingtype	"object oriented"
private String employment_status	"full time"
private int working_hours	21

Figure 13: screenshot of tutor class before removing tutor

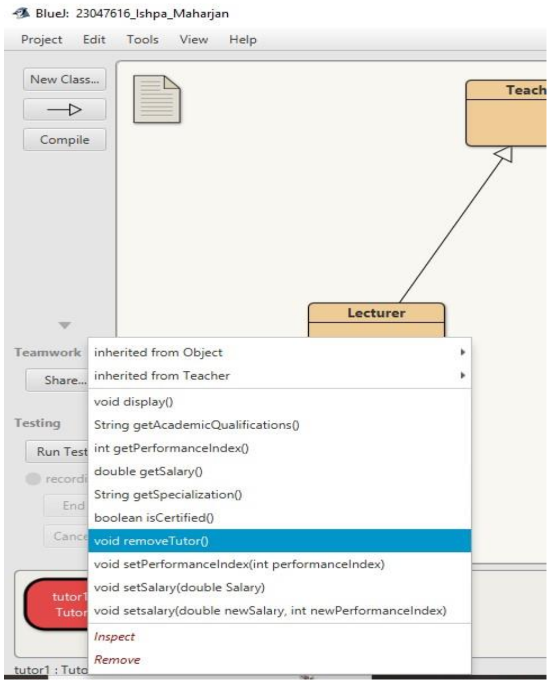
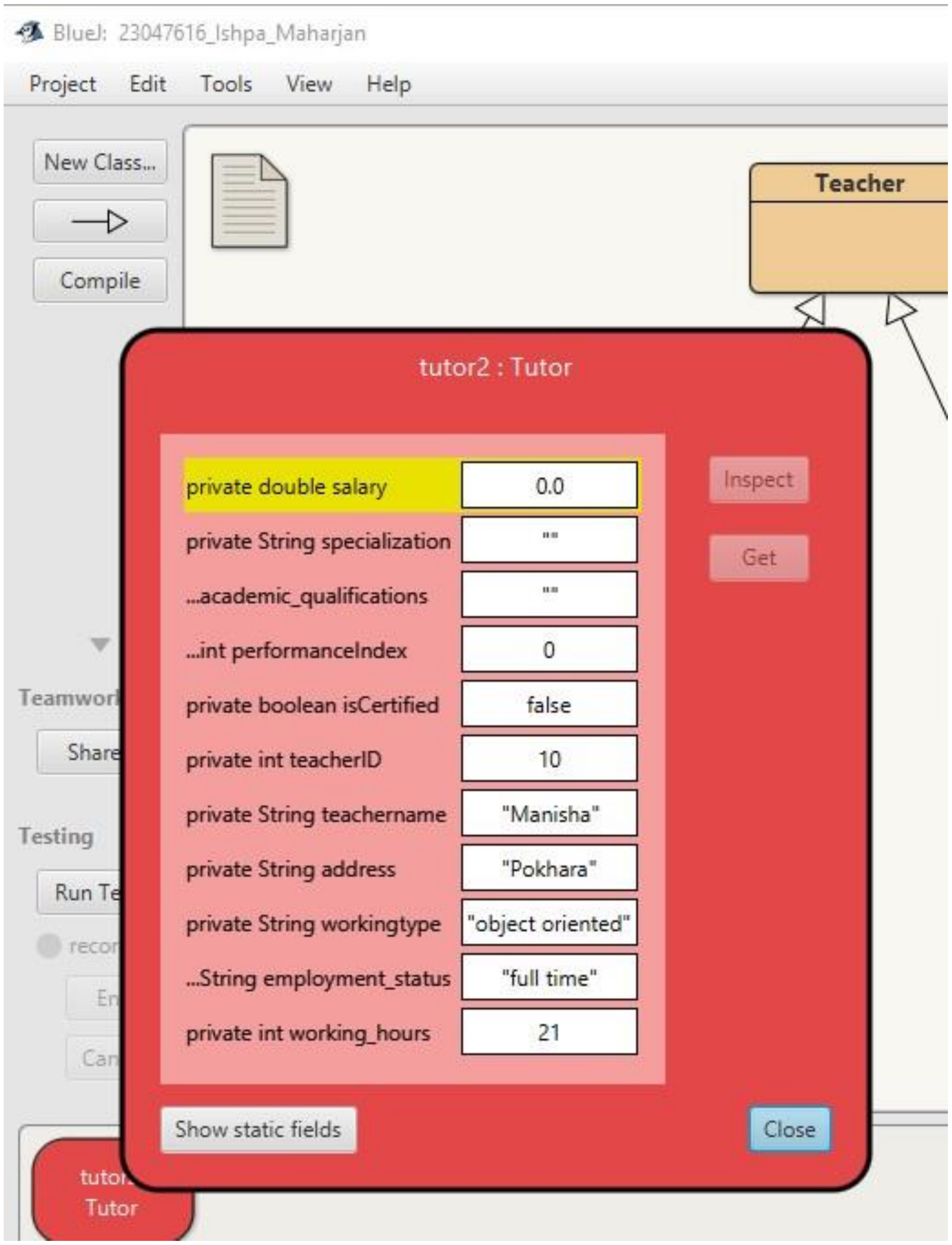

BlueJ: 23047616_Ishpa_Maharjan
Project Edit Tools View Help
New Class...
Compile
Teach
Lecturer
Teamwork
Share...
Testing
Run Test
tutor1
Tutor
inherited from Object
inherited from Teacher
void display()
String getAcademicQualifications()
int getPerformanceIndex()
double getSalary()
String getSpecialization()
boolean isCertified()
void removeTutor()
void setPerformanceIndex(int performanceIndex)
void setSalary(double Salary)
void setsalary(double newSalary, int newPerformanceIndex)
Inspect
Remove

Figure 14: removing Tutor()

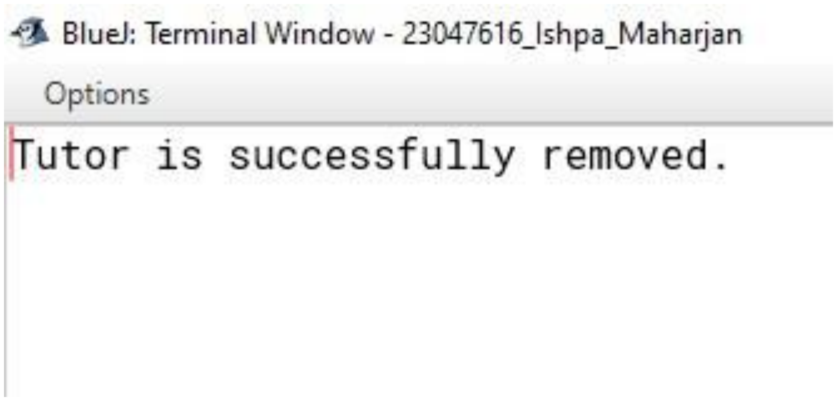


tutor2 : Tutor

private double salary	0.0
private String specialization	""
...academic_qualifications	""
...int performanceIndex	0
private boolean isCertified	false
private int teacherID	10
private String teachername	"Manisha"
private String address	"Pokhara"
private String workingtype	"object oriented"
...String employment_status	"full time"
private int working_hours	21

Figure 15: screenshot of Tutor class after removing Tutor

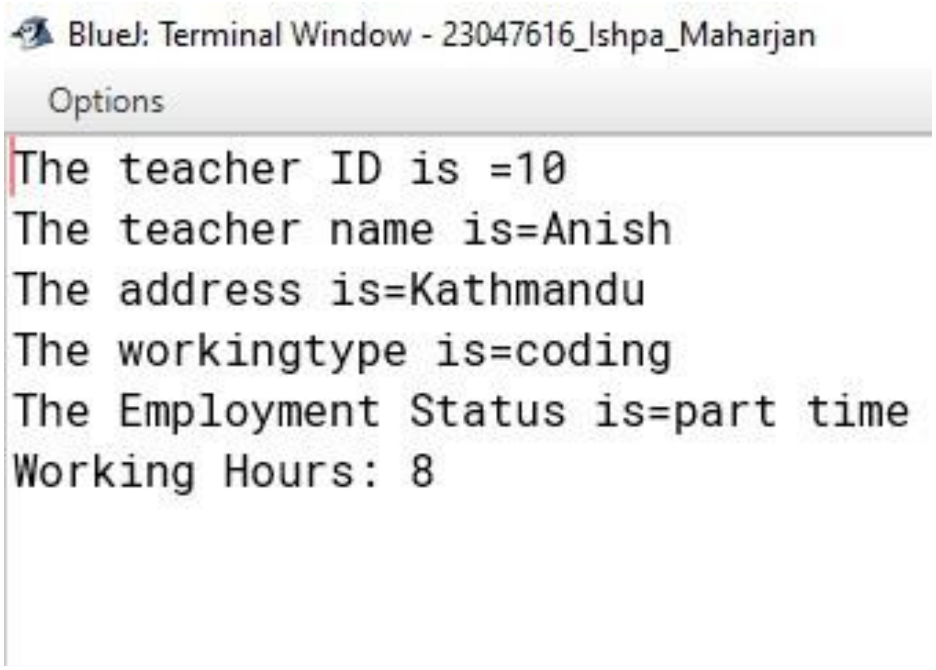
Output Result of Tutor class: removing Tutor



```
Tutor is successfully removed.
```

Figure 16: Output Result of Tutor class

Output Result of Tutor class



```
The teacher ID is =10
The teacher name is=Anish
The address is=Kathmandu
The workingtype is=coding
The Employment Status is=part time
Working Hours: 8
```

Figure 17: Output Result of Tutor class

5.4 For displaying lecturer and tutor classes

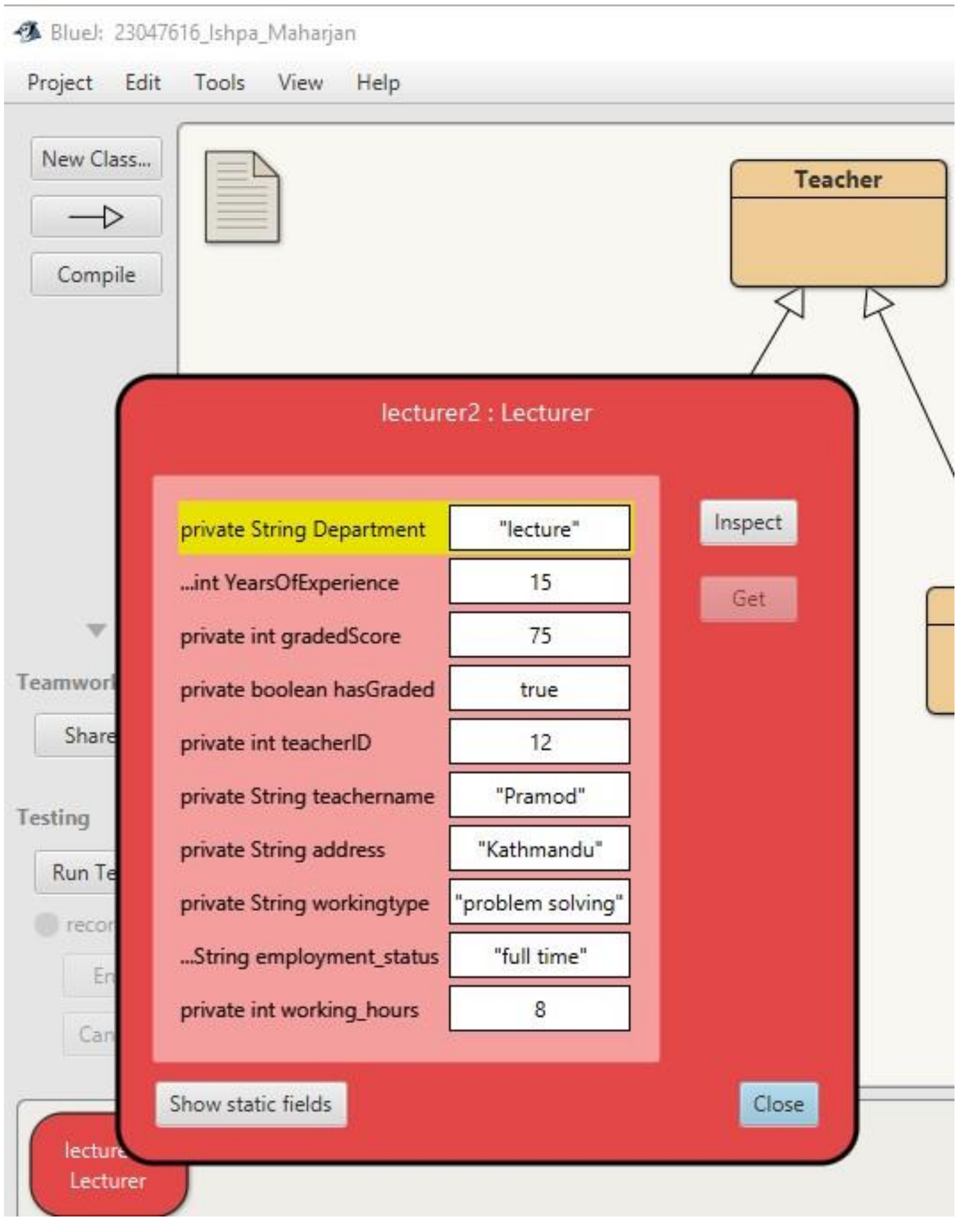
Table 14

Test	4
Objective	Display the details of Lecturer and Tutor classes.
Action of Lecturer class	➢ The lecturer is called with the following arguments: Lecturer ID: 12 Lecturer name: “Pramod” Address: “Kathmandu” Working type: ‘Problem solving” Working hours: 8 Employment status: “Full time” Department: “Lecture” Years of experience: 15 ➢ Inspection of the Lecturer class. ➢ void appointLecturer is called with the following arguments. Department: “Lecture” Years of experience: 15 gradedScore: 0 hasGraded: false teachername: “Pramod” address: “Kathmandu” workingtype: “Problem solving” employment_status: “Full time” working_hours: 8 ➢ Re-inspection of the Lecturer class Department: “Lecture” Years of experience: 15 gradedScore: 75 hasGraded: false teachername: “Pramod” address: “Kathmandu” workingtype: “Problem solving” employment_status: “Full time” working_hours: 8

Action of Tutor class	➢ The Tutor is called with the following arguments: Lecturer ID: 10 Lecturer name: “Manisha” Address: “Pokhara” Working type: “Object oriented” Working hours: 21 Salary: 60000 Department: “coding” Academic qualifications: “master’s degree” Performance Index: 7 ➢ Inspection of the Tutor class. ➢ void appointTutor is called with the following arguments: salary: 60000 specialization: “coding” academic qualifications: “master’s degree” performanceIndex: 7 isCertified: false teacher ID: 10 teachername: “Manisha” address: “Pokhara” working type: “object oriented” employment_status: “full time” working hours: 21 ➢ Re-inspection of the Tutor class salary: 83000 specialization: “coding” academic qualifications: “master’s degree” performanceIndex: 9 isCertified: true teacher ID: 10 teachername: “Manisha” address: “Pokhara” working type: “object oriented” employment_status: “full time” working hours: 21

Action of tutor class to remove tutor	➢ Inspection of the Tutor class after removing tutor salary: 0.0 specialization: “” academic qualifications: “” performanceIndex: 0 isCertified: false teacher ID: 10 teachername: “Manisha” address: “Pokhara” working type: “object oriented” employment_status: “full time” ➢ working hours: 21
Expected Result	The Lecturer and Tutor class would be successfully displaying the condition it satisfies.
Actual Result	The Lecturer and Tutor class successfully displayed the condition it satisfies.
Conclusion	The test is successful.

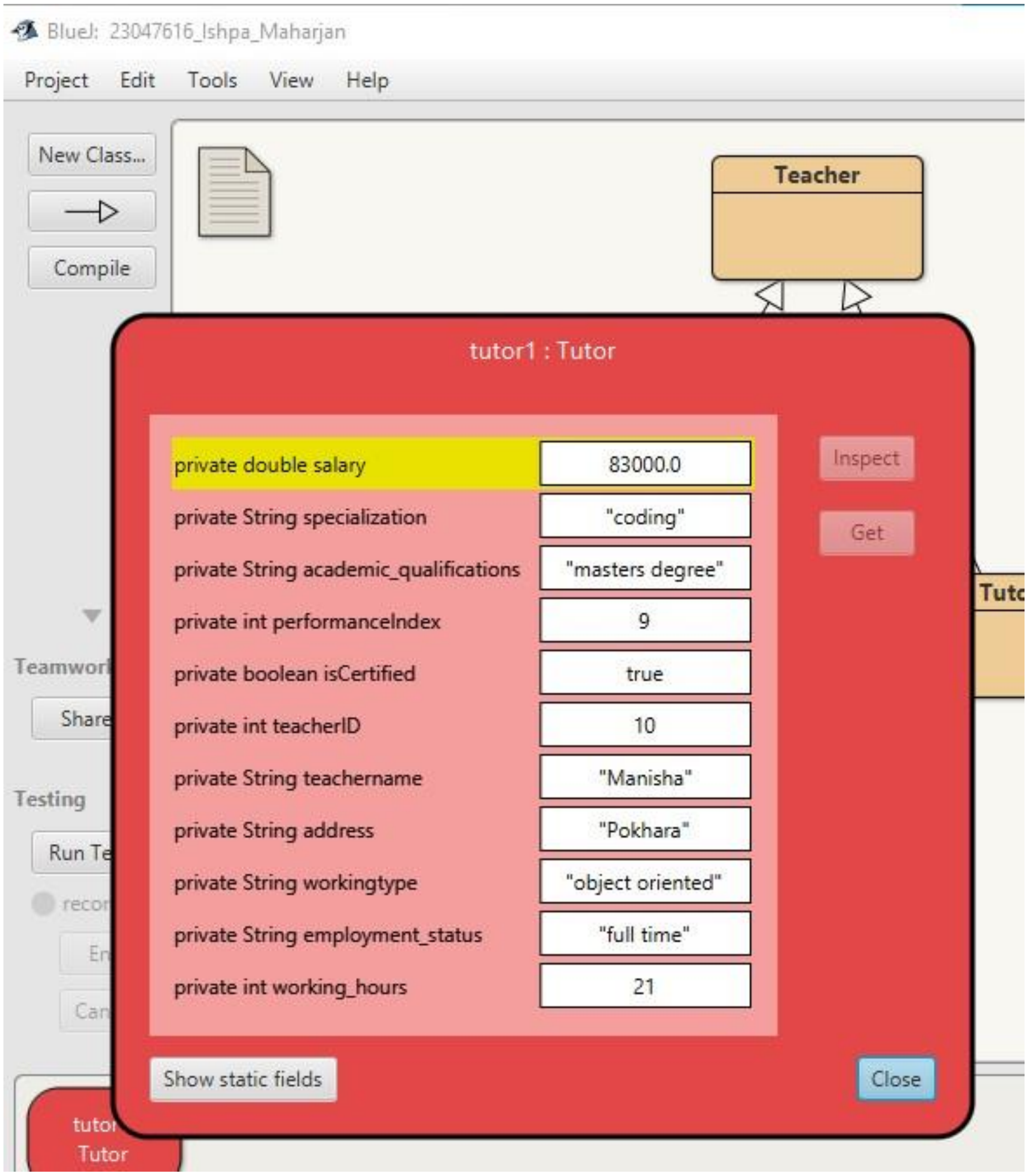
5.5 Display all lecturer and tutor result:



lecturer2 : Lecturer

private String Department	"lecture"
...int YearsOfExperience	15
private int gradedScore	75
private boolean hasGraded	true
private int teacherID	12
private String teachername	"Pramod"
private String address	"Kathmandu"
private String workingtype	"problem solving"
...String employment_status	"full time"
private int working_hours	8

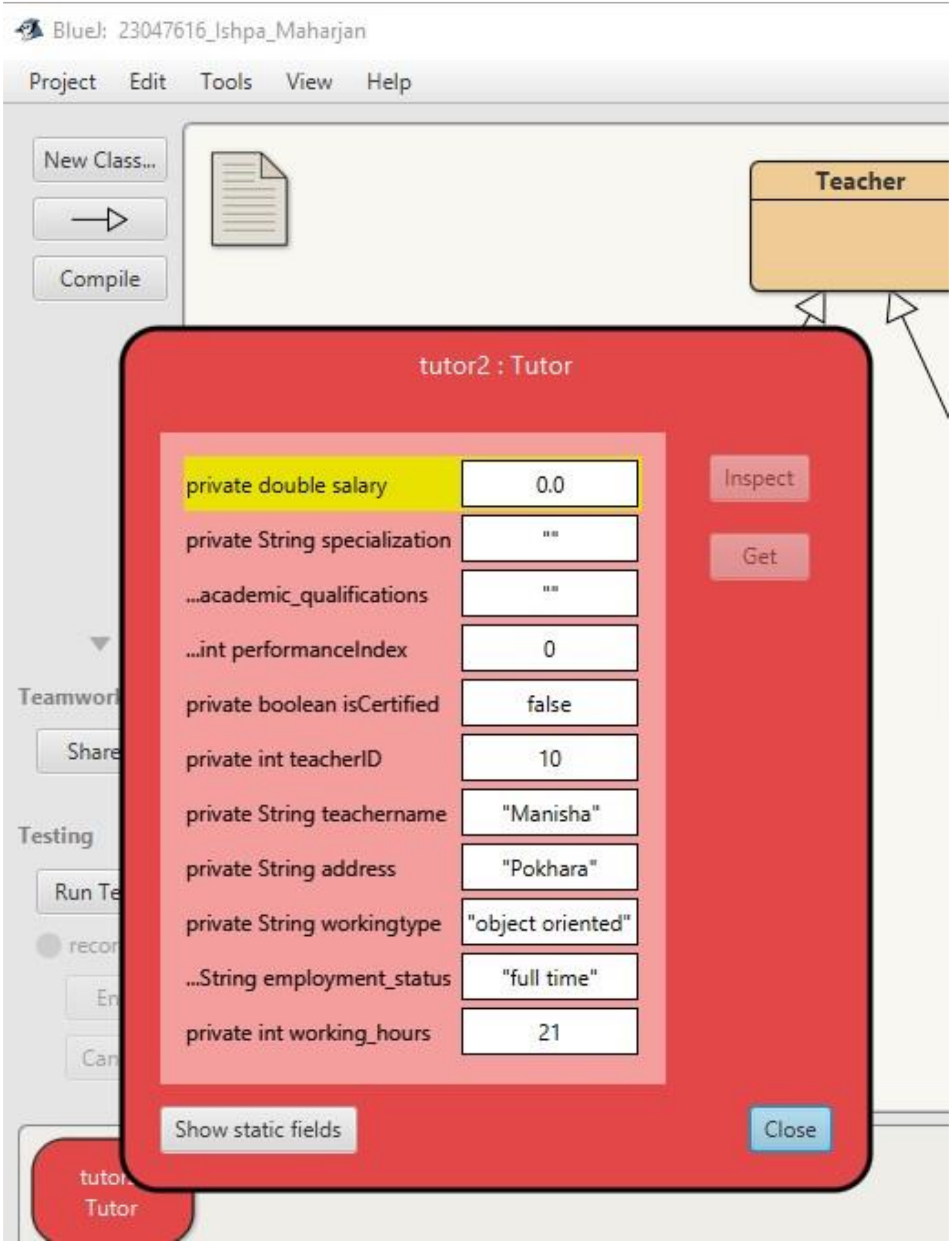
Figure 18: screenshot of displaying lecturer class



tutor1 : Tutor

private double salary	83000.0
private String specialization	"coding"
private String academic_qualifications	"masters degree"
private int performanceIndex	9
private boolean isCertified	true
private int teacherID	10
private String teachername	"Manisha"
private String address	"Pokhara"
private String workingtype	"object oriented"
private String employment_status	"full time"
private int working_hours	21

Figure 19: screenshot of displaying Tutor class



tutor2 : Tutor

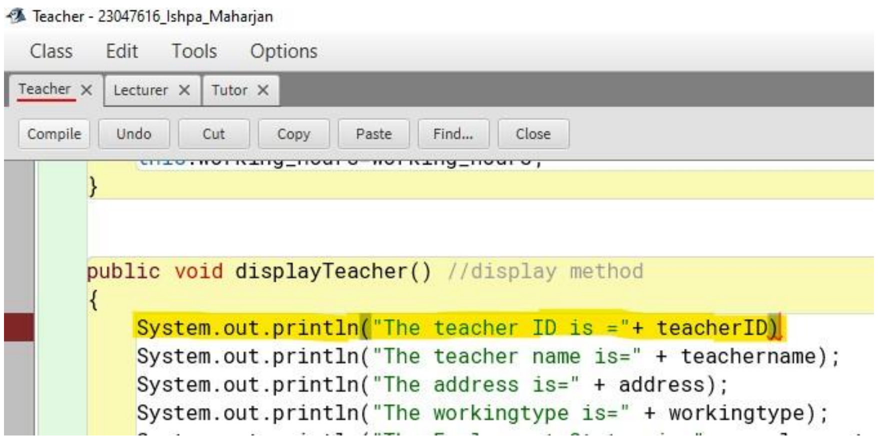
private double salary	0.0
private String specialization	""
...academic_qualifications	""
...int performanceIndex	0
private boolean isCertified	false
private int teacherID	10
private String teachername	"Manisha"
private String address	"Pokhara"
private String workingtype	"object oriented"
...String employment_status	"full time"
private int working_hours	21

Figure 20: screenshot of removing and displaying tutor class

6.Error Detection and Correction

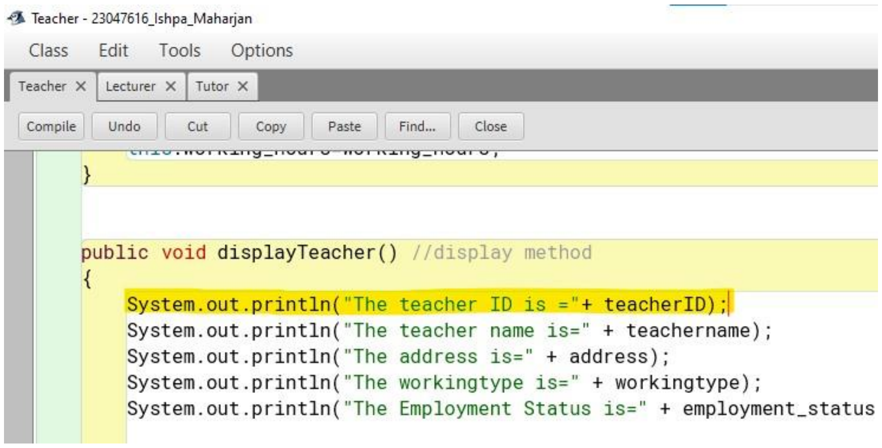
6.1 Syntax error:

A syntax error is the error in the syntax of coding or programming languages which is entered by a programmer or a coder. Syntax errors are detected by a software program called compiler. A programmer must correct it before compilation and run the program (Rouse, 2023).



```
}

public void displayTeacher() //display method
{
    System.out.println("The teacher ID is ="+ teacherID)
    System.out.println("The teacher name is=" + teachername);
    System.out.println("The address is=" + address);
    System.out.println("The workingtype is=" + workingtype);
```

Figure 21: Screenshot of a syntax error

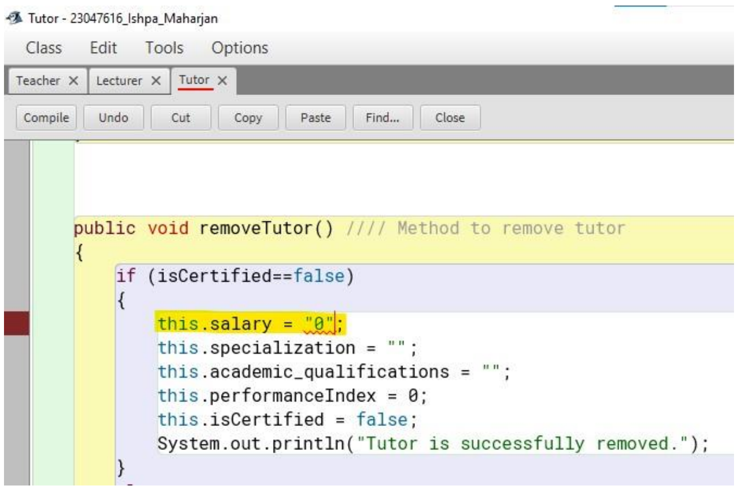


```
}

public void displayTeacher() //display method
{
    System.out.println("The teacher ID is ="+ teacherID);
    System.out.println("The teacher name is=" + teachername);
    System.out.println("The address is=" + address);
    System.out.println("The workingtype is=" + workingtype);
    System.out.println("The Employment Status is=" + employment_status
```

Figure 22: Screenshot of a syntax error corrected

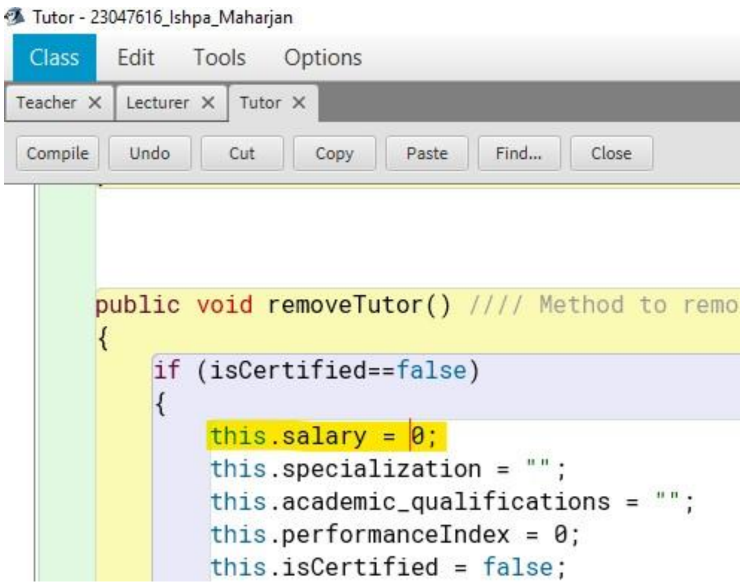
6.2 Semantics error:

A semantic error is an error which is grammatically correct but it doesn't make any sense in terms of programming. The semantic error can arise using the wrong variable or using wrong operator or doing operation in wrong order. For an example int x= 2.5 where integer value doesn't take decimal value (Jaiswal, 2021).



```
public void removeTutor() //// Method to remove tutor
{
    if (isCertified==false)
    {
        this.salary = "0";
        this.specialization = "";
        this.academic_qualifications = "";
        this.performanceIndex = 0;
        this.isCertified = false;
        System.out.println("Tutor is successfully removed.");
    }
```

Figure 23: Screenshot of semantics error



```
public void removeTutor() //// Method to remo
{
    if (isCertified==false)
    {
        this.salary = 0;
        this.specialization = "";
        this.academic_qualifications = "";
        this.performanceIndex = 0;
        this.isCertified = false;
```

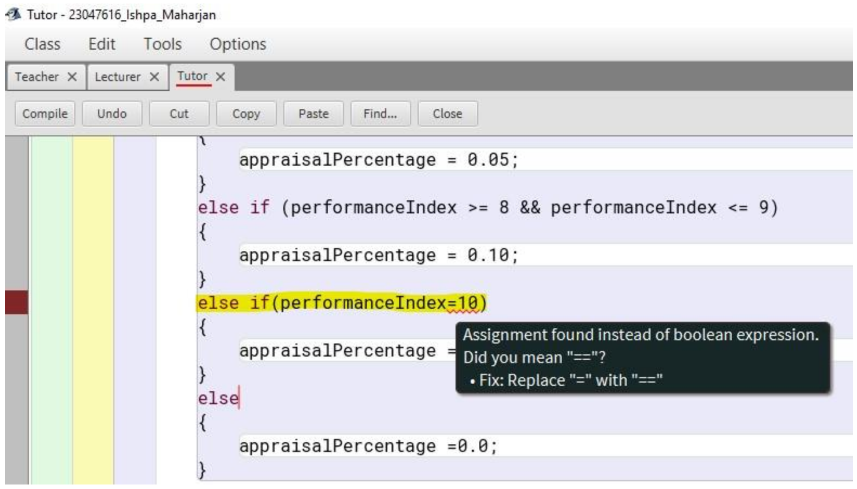
Figure 24: Screenshot of correction of semantics error

6.3 Logical error:

A logical error is a mistake in the implementation of logic within a program which leads to unexpected error and incorrect results during the run time. The programmer has to find the error himself (KOCHITTY, 2020).

For example:

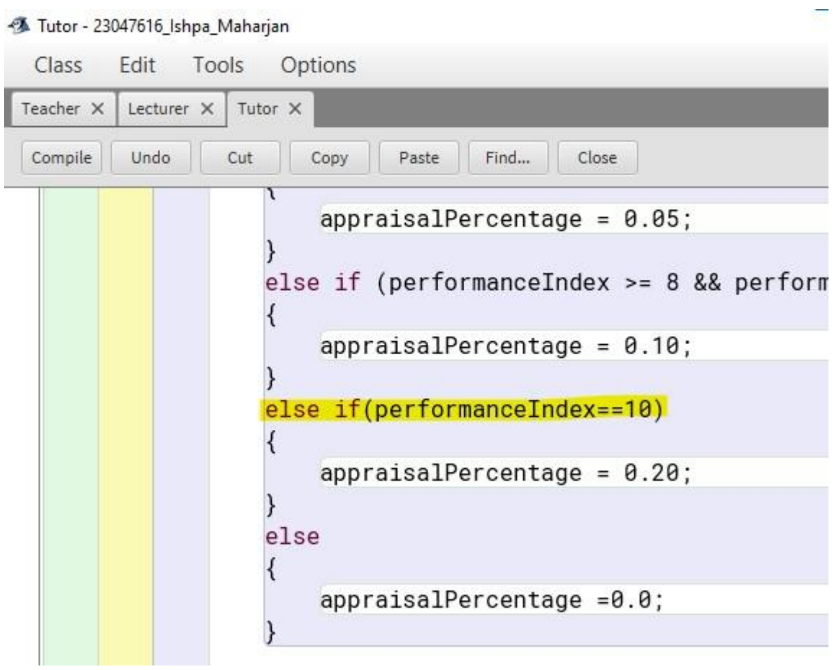
```
int a = 10;

if(a%10=0)

{

        System.out.print("Divisible by 10");

}
```


Tutor - 23047616_Ishpa_Maharjan
Class Edit Tools Options
Teacher Lecturer Tutor
Compile Undo Cut Copy Paste Find... Close

```
{
    appraisalPercentage = 0.05;
}
else if (performanceIndex >= 8 && performanceIndex <= 9)
{
    appraisalPercentage = 0.10;
}
else if(performanceIndex=10)
{
    appraisalPercentage =
}
else
{
    appraisalPercentage =0.0;
}
```

Assignment found instead of boolean expression.
Did you mean "=="?
• Fix: Replace "=" with "=="

Figure 25: Screenshot of logical error



```
    appraisalPercentage = 0.05;
}
else if (performanceIndex >= 8 && perform
{
    appraisalPercentage = 0.10;
}
else if(performanceIndex==10)
{
    appraisalPercentage = 0.20;
}
else
{
    appraisalPercentage =0.0;
}
```

Figure 26: Screenshot of logical error corrected

7. Conclusion

In conclusion this documentation has provided a comprehensive understanding of this coursework which let me gain good knowledge about the java programming which is also an Object-Oriented Programming language.

The major things I learned is:

- Creating a class.
- Creating inheritance which means creating new classes based on existing ones.
- Method Overriding
- Implementing different keyword in inheritance
- Encapsulation where accessor or getter method reads the value of a variable while mutator or setter method updates or sets the value of a variable.
- Creating a constructor
- Method to display the classes with non-return type

There are some problems and their solutions that I faced during this coursework such as:

- I had to face some of the logical error in case of if else statement in Tutor class where I solved with the help of pdf and slides provided in My second Teacher.
- Also, I had some technical error or some codes were missing which I solved researching a lot and tried hard to understand the questions provided.

The most enjoying part of this documentation for me was testing. I was enjoying and learning by assigning the value and taking screenshots of the result which was testing successfully while on the other hand pseudocode consumed a bit more time to complete although I was learning.

As I conclude this coursework, the journey through this documentation has been one of the discovery and empowerment and I am confident enough to achieve knowledge regarding this programming and documentation.

8.References

References

aws, 2023. *aws.* [Online]
Available at: https://aws.amazon.com/what-is/java/

BYJU'S, 2024. *BYJU'S exam prep.* [Online]
Available at: https://byjus.com/govt-exams/microsoft-word/

Jaiswal, S., 2021. *JavaTpoint.* [Online]
Available at: https://www.javatpoint.com/semantic-error

KOCHITTY, J., 2020. *toppr.* [Online]
Available at: https://www.toppr.com/guides/computer-science/programming-methodology/general-concepts-of-programming-methodology/logical-errors/

Rouse, M., 2023. *TechDictionary.* [Online]
Available at: https://www.techopedia.com/definition/13391/syntax-error

Wales, J., 2023. *wikipedia.* [Online]
Available at: https://en.wikipedia.org/wiki/BlueJ

9.Appendix

9.1Teacher Class

```
public class Teacher

{

    private int teacherID;

    private String teachername;

    private String address;

    private String workingtype;

    private String employment_status;

    private int working_hours;


    public Teacher(int teacherID, String teachername, String address, String
    workingtype, String employment_status)

    {

        this.teacherID=teacherID;

        this.teachername=teachername;

        this.address=address;
        this.workingtype=workingtype;

        this.employment_status=employment_status;

     }




    public int getteacherID()

    {

        return this.teacherID;

    }

    public String getteachername()

    {

        return this.teachername;

    }

    public String getaddress()

    {

        return this.address;

    }

    public String getworkingtype()
    {
        return this.workingtype;
    }
    public String getemployment_status()
    {
        return this.employment_status;
    }
    public int getworking_hours()
    {
        return this.working_hours;
    }



    public void setworking_hours(int working_hours)
    {
        this.working_hours=working_hours;
    }
```

}

```
public void displayTeacher()

{

    System.out.println("The teacher ID is ="+ teacherID);

    System.out.println("The teacher name is=" + teachername);

    System.out.println("The address is=" + address);

    System.out.println("The workingtype is=" + workingtype);

    System.out.println("The Employment Status is=" + employment_status);




    if(working_hours > 0)

        {

            System.out.println("Working Hours: " + working_hours);

        }

    else

        {

            System.out.println("Working Hours: Not assigned");

        }

}
```

```
}
```

9.2 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, int working_hours, String employment_status, String Department, int YearsOfExperience)
  {
      super(teacherID,teachername,address,workingtype,employment_status);
      setworking_hours(working_hours);
      this.Department=Department;
      this.YearsOfExperience=YearsOfExperience;
```

```
     this.gradedScore=0;  //passing default value
     this.hasGraded=false;   //default value
   }

   public String getDepartment()
   {
      return this.Department;
   }

   public int getYearsOfExperience()
   {
      return this.YearsOfExperience;
   }

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

```
public boolean hasGraded()

{

    return this.hasGraded;

}




public void setGradedScore(int gradedScore)

{

    this.gradedScore = gradedScore;

}




public void gradeAssignment(int gradedScore, String Department, int
YearsOfExperience)

{
```

```
if(hasGraded==false)

if (YearsOfExperience >= 5 && Department.equals(Department))

{

        if (gradedScore >= 70)

        {

            System.out.print("grade: A");

        }

        else if (gradedScore >= 60)

        {

            System.out.print("grade: B");

        }

        else if (gradedScore >= 50)

        {

            System.out.print("grade:C");

        }

        else if (gradedScore >= 40)

        {

           System.out.print("grade:D");
```

```
                }
                else
                {
                    System.out.print("grade:E");
                }
                hasGraded=true;
            }
            else
            {
                System.out.print("The assignment is already graded.");
            }
        }




        public void display()
        {
            super.displayTeacher();

            System.out.print("The Department is=" +Department);

            System.out.print("The Years Of Experience is=" +YearsOfExperience);

            if(hasGraded)

            {

                System.out.print("The Graded Score is=" +gradedScore);

            }

            else

            {

                System.out.print("The Score has not been Graded");

            }

        }




    }
```

9.3 Tutor class

```
public class Tutor extends Teacher

{

    private double salary;

    private String specialization;

    private String academic_qualifications;

    private int performanceIndex;

    private boolean isCertified;



     public Tutor(int teacherID, String teachername, String address, String
     workingtype, String employment_status, int working_hours, double salary, String
     specialization, String academic_qualifications, int performanceIndex)

     {

             super(teacherID,teachername,address,workingtype,employment_status);

             setworking_hours(working_hours);

             this.salary= salary; //local variable is stored in instance variable

             this.specialization= specialization;
```

```
            this.academic_qualifications=academic_qualifications;

            this.performanceIndex=performanceIndex;

            this.isCertified=false;  //default values

   }




    public double getSalary()

    {

       return this.salary;

    }


    public String getSpecialization()

    {

       return this.specialization;

    }


    public String getAcademicQualifications()

    {
        return this.academic_qualifications;

    }



    public int getPerformanceIndex()

    {

        return this.performanceIndex;

    }



    public boolean isCertified()

    {

        return this.isCertified;

    }





    public void setSalary(double Salary)

    {

        this.salary=salary;
    }



    public void setPerformanceIndex(int performanceIndex)

    {

        this.performanceIndex = performanceIndex;

    }




    public void setsalary(double newSalary, int newPerformanceIndex)

    {

        if(isCertified==false)

        if (performanceIndex > 5 && getworking_hours()>20)

        {

            double appraisalPercentage;

                if (performanceIndex >= 5 && performanceIndex <= 7)

                {

                    appraisalPercentage = 0.05;

                }
```

```
else if (performanceIndex >= 8 && performanceIndex <= 9)

{

    appraisalPercentage = 0.10;

}

else if(performanceIndex==10)

{

    appraisalPercentage = 0.20;

}

else

{

    appraisalPercentage =0.0;

}




double appraisal = salary * appraisalPercentage;

this.salary = newSalary+appraisal;

this.performanceIndex = newPerformanceIndex;

isCertified = true;
```

```
                System.out.println("Salary is successfully approved.");

            }

                else

                {

                    System.out.println("Salary cannot be approved for the tutor.");

                }


            else

            {

                System.out.println("Salary cannot be changed for a certified tutor.");

            }

        }




        public void removeTutor()

        {

            if (isCertified==false)
            {
                this.salary = 0;
                this.specialization = "";
                this.academic_qualifications = "";
                this.performanceIndex = 0;
                this.isCertified = false;
                System.out.println("Tutor is successfully removed.");
            }
            else
            {
                System.out.println("Certified tutor cannot be removed");
            }
        }



        public void display()
        {
            if (isCertified)

            {

                System.out.println("The Salary of a turor+" + salary);

                System.out.println("The Specialization of a tutor=" + specialization);

                System.out.println("The Academic Qualifications:of a tutor=" +
                academic_qualifications);

                System.out.println("The Performance Index of a tutor=" +
                performanceIndex);

            }

            else

            {

              super.displayTeacher();

            }

        }

    }
```