Module Code & Module Title:

CS4001NT Programming

Assessment Weightage & Type:

30% Individual Coursework

Year and Semester:

2023 Autumn

Student Name: Bidur Siwakoti
London Met ID:BIS0388
College ID: NP05CP4A230013.
Assignment Due Date: 26 Jan 2024
Word Count:5039

Table of Contents

Table of Images:

Table of Tables:

1 INTRODUCTION

Figure 1:James Gosling

Java is a general-purpose, object-oriented, and high-level programming language. It is used to develop different applications such as desktop applications, network applications, web applications, mobile applications, distributed applications, and embedded and smart system applications. Java was developed in the 1990s by James Gosling in the early 1990s by Sun Microsystems in the USA. (GeeksForGeeks, 03 April 2023) As Java is an object-oriented language, it uses the concept of class and object. In Java, a class serves as a blueprint or template that defines the attributes and behaviors of a certain class where objects are the instances of this class.

1.1 About The Coursework

In this project, three distinct classes Teacher, Lecturer, and tutor have to be designed, following the principles of object-oriented programming in Java. In this project, The Teacher class serves as the superclass or the foundational blueprint, consisting of six essential attributes: Teacher ID, Teacher Name, Address, working type, employment status, and working hours where Teacher ID and working hours accepted integer data type and rest of other accepted string data type. The Teacher class established the relation between the lecturer class and the tutor class. The lecture class has attributes like department, year of experience, graded score, and Boolean indicator for grading status denoted as ‘hasGraded’. Lastly, the Tutor class, another subclass of teachers contains attributes such as salary, specialization, academic qualification, performance index, and Boolean indicator for certification status denoted as ‘isCertified’. This program offers an organized method for storing and handling information related to teachers, lecturers, and tutors. It can be applicable in an educational institution, specifically in college and university, where the systematic organization of personal data is very important for efficient management.

1.2 Tools used in this coursework

- **BlueJ**

  BlueJ is a user-friendly Integrated Development Environment (IDE) used for developing Java programs. It is renowned for its simplicity, and user-friendly interface which make it an excellent option for beginners and educators also it was originally developed for educational purposes. It also supports object-oriented principles. BlueJ comes with an integrated debugger that allows user to visualize their code, inspect variables, and understand the flow of program execution. Therefore, I use BlueJ as a code editor for my assessment.

Figure 2:BlueJ

- **Microsoft word**

  Microsoft Word is a user-friendly word-processing tool used for creating various types of documents such as letters, agreements, reports, etc. Its widespread use across every sector displays its versatility and adaptability. I use Microsoft Word to compose this coursework as it provides an easy and efficient way of creating well-structured content.

Figure 3:Ms-Word

2 CLASS DIAGRAM

A class diagram is a static diagram that represents the static view of an application. It is used not just to create executable code for software applications but also to visualize, describe, and document various parts of a system. It describes the attributes and operation of a class. Class diagrams are Significantly used in the designing of object-oriented languages such as Java, and C++. The class diagram has three sections. The upper section has the name of the class, the middle section has all the attributes, and the lower section has methods and operations.

The attributes are written along with its visibility factor,

(+) indicates public

(-) indicates private and

(#) indicates protected. (GeeksForGeeks, 18 jan 2024)

- The class diagram for the Teacher class is given below:

Table 1:class diagram of teacher

Teacher
-teacherId:int -teacherName:string -address:string -workingType:string -employmentstatus:string -workingHours:int
<<constructor>> Teacher(teacherID:int, techerName:string, address:string, workingType:string, employmentStatus:string) + getTeacherID:int +getTeacherName:String +getAddress:string +getworkingType:string +getEmploymentStatus:string +getworkingHours:int +setworkingHours:int +displayTeacherInfo:void

- The class diagram for the lecturer class:

Table 2:class diagram of Lecturer class

Lecturer
-Department:String -yearsofExperiences:int -gradedscore:int -hasGraded:boolean
<<constructor>> lecturer(teacherID:int, techerName:string, address:string, workingType:string, department:string, yearofExperiences:int, workingHours:int) + getdepartment:string + getYearofExperiences:int + getGradedScore:int + getHasGraded:Boolean +setGradedScore:void +GradeAssignment:void + DisplayTeacherInfo:void

- The class diagram for the tutor class:

Table 3:class diagram of Tutor class

Tutor
-salary:double -specialization:string -academicQualifiacations:string -performanceIndex:int -isCertified:boolean
+ <<constructor>> Tutor(teacherID:int, teacherName:string, address:string, workingType:String, employmentStatus:Strings, workingHours:int, Salary:int, Specialization:string, academicQualification:strings, performanceIndex:int”) +getsalary:double +getSpecialization:string +getacademicQualification:string +getPerformanceIndex:int +setSalary:void +removeTutor:void +displayTeacherInfo:void

- The class diagram showing hierarchical inheritance:

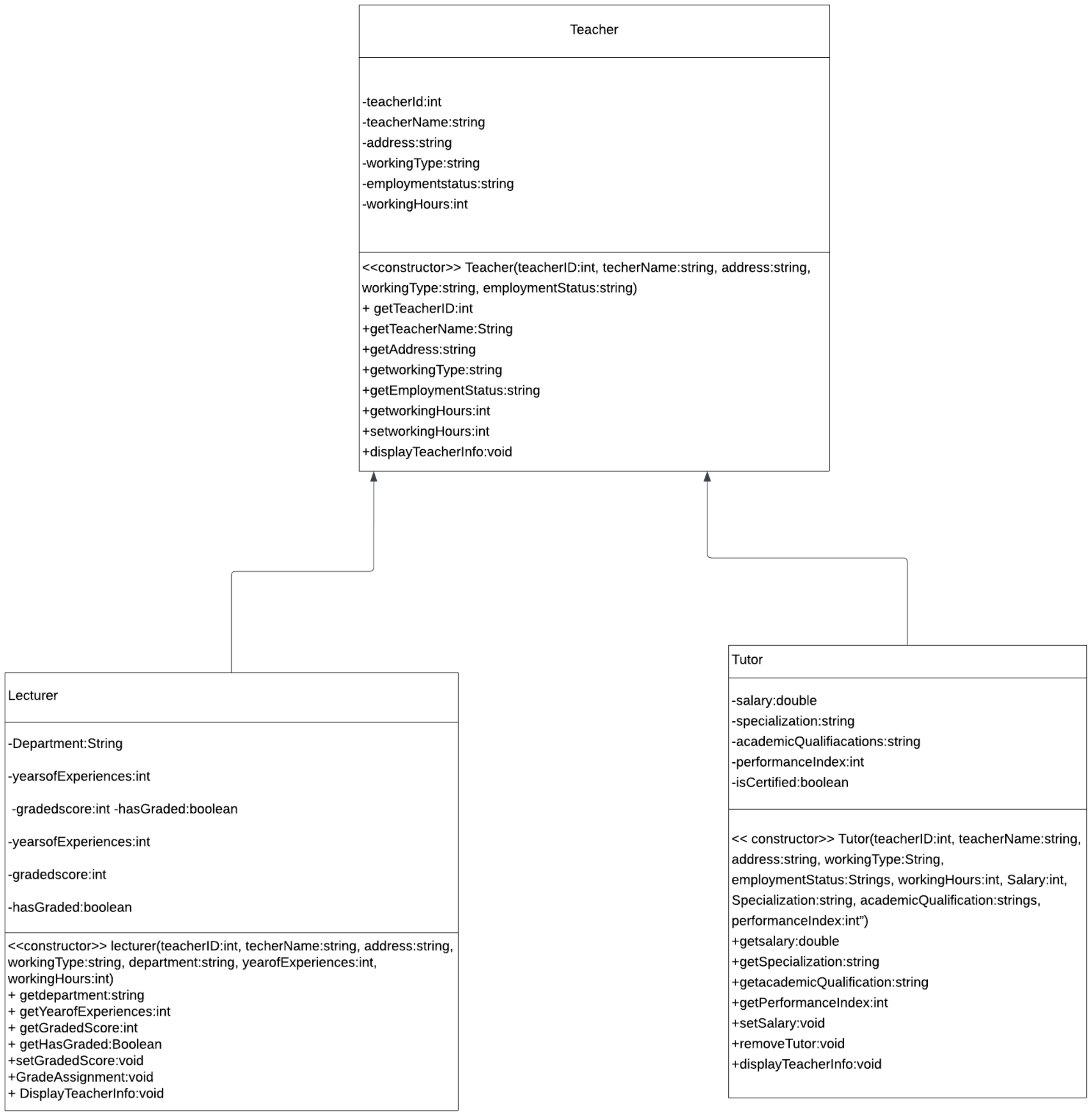

Teacher
-teacherId:int
-teacherName:string
-address:string
-workingType:string
-employmentstatus:string
-workingHours:int
<<constructor>> Teacher(teacherID:int, techerName:string, address:string, workingType:string, employmentStatus:string)
+ getTeacherID:int
+getTeacherName:String
+getAddress:string
+getworkingType:string
+getEmploymentStatus:string
+getworkingHours:int
+setworkingHours:int
+displayTeacherInfo:void
Lecturer
-Department:String
-yearsofExperiences:int
-gradedscore:int -hasGraded:boolean
-yearsofExperiences:int
-gradedscore:int
-hasGraded:boolean
<<constructor>> lecturer(teacherID:int, techerName:string, address:string, workingType:string, department:string, yearofExperiences:int, workingHours:int)
+ getdepartment:string
+ getYearofExperiences:int
+ getGradedScore:int
+ getHasGraded:Boolean
+setGradedScore:void
+GradeAssignment:void
+ DisplayTeacherInfo:void
Tutor
-salary:double
-specialization:string
-academicQualifiacations:string
-performanceIndex:int
-isCertified:boolean
<< constructor>> Tutor(teacherID:int, teacherName:string, address:string, workingType:String, employmentStatus:Strings, workingHours:int, Salary:int, Specialization:string, academicQualification:strings, performanceIndex:int")
+getsalary:double
+getSpecialization:string
+getacademicQualification:string
+getPerformanceIndex:int
+setSalary:void
+removeTutor:void
+displayTeacherInfo:void

Figure 4:Inheritance diagram

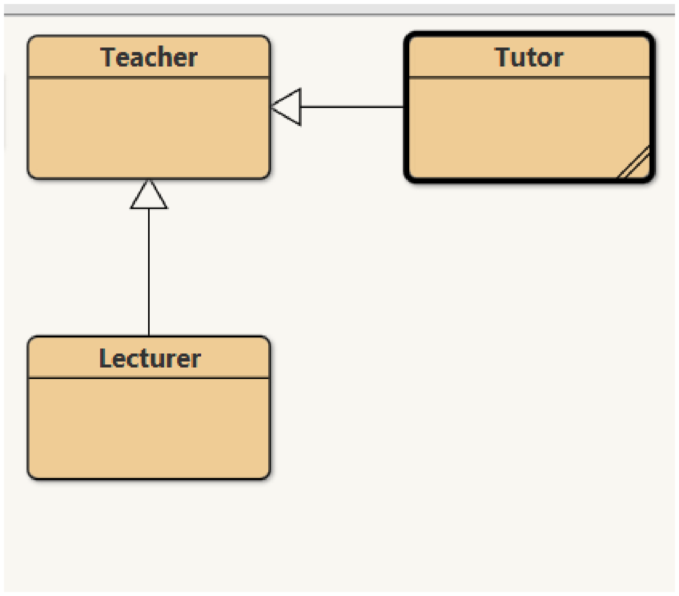


Figure 5:Class Diagram (BlueJ)

3 PSEUDOCODE

Pseudocode is defined as a method of describing a process or writing programming code and algorithm using natural language such as English. It is not a code itself, but rather a description of what the code should do. (Study.com, n.d.) Pseudocode is not complied by computer, its main purpose is to read and understood by human. It is also referred as “False code” or “representation of code”. (Anon., n.d.) In this coursework I have to create the pseudocode of all class i.e. Teacher class, Lecturer class, and Tutor class.

The pseudocode for the Teacher class is given below:

```
CREATE a parent class Teacher

DO

DECLARE instance variable Teacherid as  private string

DECLARE instance variable Teachername as  private string

DECLARE instance variable Addresh as private string

DECLARE instance variable Workingtype as  private string

DECLARE instance variable Employementstatus as  string

DECLARE instance variable Workinghours as integer

END DO


CREATE a constructor as Teacher(set Teacherid as integer , set Teachername as String, set Addresh as String , set Workingtype as String, set Employementstatus as String)

DO

        SET Teacherid to assign Teacherid

        SET Teachername to assign Teachername

        SET Addresh to assign Addresh

        SET Workingtype to assign Workingtype
```

```
    SET Employementstatus to assign Employementstatus
END DO
CREATE a integer getter method for Teacherid()
DO
    Return the value of Teacherid
END DO
CREATE a String getter method for Teachername()
DO
    Return the value of Teachername
ENDDO
CREATE a String getter method for Addresh()
DO
    Return the value of Address
END DO
CREATE a String getter method for Workingtype()
DO
    Return the value for Workingtype
END DO
CREATE a string getter method for Employementstatus()
DO
    Return the value for Employementststatus
END DO
CREATE a string getter method for Workinghours()
DO
```

```
        Return the value of Workinghours
END DO
CREATE a void type of setter method for Workinghours(int Workinghours)
DO
        SET Workinghours to assign Workinghours
CREATE a void type of method for Display()
DO
        Display the value of Teacherid
        Display the value of Teachername
        Display the value of Addresh
        Display the value of Wotkingtype
        Display the value of Employementstatus
        IF Workinghours is greater than  zero
DO
        Display the value of Workinghours
END DO
Else
Display the value of working hour as “working hours is not assigned”
END DO
END DO
END DO
```

The pseudocode for Lecturer class is given below:

```
CREATE a child class Lecturer that extends with Teacher
DO
   DECLARE instance variables Department as private String
    DECLARE instance variables YearsOfExperience as private integer
   DECLARE instance variables GradedScore as private integer
   DECLARE instance variables HasGraded as private boolean
   CREATE a constructor Lecturer(TeacherId, TeacherName, Address, WorkingType,
EmploymentStatus, Department, YearsOfExperience, WorkingHours)
   DO
      SET Department to assign Department
      SET YearsOfExperience to assign YearsOfExperience
      SET GradedScore to 0
      SET HasGraded to false
   ENDDO
   CREATE String getter method for getDepartment()
   DO
      Return the value of Department
   END DO

   CREATE integer getter method for getYearsOfExperience()
   DO
      Return the value of YearsOfExperience
```

```
END DO
CREATE int getter method for getGradedScore()
DO
    Return the value of GradedScore
END DO
CREATE boolean getter method for getHasGraded()
DO
    Return the value of HasGraded
END DO
CREATE a void type setter method for setGradedScore(int NewGradedScore)
DO
    Set GradedScore to assign NewGradedScore
END DO
CREATE public method gradeAssignment(int GradedScore, String Department, int YearsOfExperience)
DO
    IF HasGraded is false
    DO
        IF YearsOfExperience is greater than or equal to 5 AND Department is equal to provided Department
        DO
IF  the value of Gradedscore is Greater than or equal to 70
DO
```

```
Print the value of grade scored as “Graded scored= ‘a’”
END DO
    IF the value of Gradedscore is Greater than or equal to 60
DO
Print the value of Grade scored as “Graded scored = ‘b’”
ENDDO
IF  the value of Gradedscore is Greater than or equal to 50
DO
Print the value of Graded scored as “Graded Scored=’c’”
ENDDO
IF  the value of Gradedscore is Greater than or equal to 40
DO
Print the value of Graded scored as “Graded Scored=’d’”
END DO
ELSE
DO
Print the value of Graded scored as “Graded Scored=’E’”
END DO
```

```
          Set HasGraded to true
```

```
        ELSE
            Print "Lecturer cannot grade assignments for this student."
        ENDDO
      ELSE
        Print " The assignments is already  graded."
      ENDDO
    ENDDO
    OVERRIDE the displayTeacherInfo() method
    DO
      Call the displayTeacherInfo() method in the superclass (Teacher)
      Print "Department= " + Department;
      Print "Years of Experiences " + YearsOfExperience
      IF HasGraded is true
      DO
        Print "Grade Scores: " + GradedScore
      ELSE
        Print "Grade Scores: Score is not graded till now"
      ENDDO
    ENDDO
ENDDO
```

- **The pseudocodes of Tutor class is given below:**

```
CREATE a child class Tutor and extends Teacher
DO
  DECLARE additional attributes
  DECLARE instance variables Salary as private double
  DECLARE instance variables Specialization as private String
  DECLARE instance variables AcademicQualifications as private String
  DECLARE instance variables PerformanceIndex as private integer
  DECLARE instance variables IsCertified as private boolean

    CREATE a constructor Tutor(TeacherId, TeacherName, Address, WorkingType, EmploymentStatus, WorkingHours, Salary, Specialization, AcademicQualifications, PerformanceIndex)
    DO
        Call the constructor of the superclass teacher with required attributes
        Set Salary to provided Salary
        Set Specialization to provided Specialization
        Set AcademicQualifications to provided AcademicQualifications
        Set PerformanceIndex to provided PerformanceIndex
        Set IsCertified to false
    END DO

    CREATE double getter method for getSalary()
```

```
DO
    Return the value of Salary.
ENDDO
CREATE String getter method for getSpecialization()
DO
    Return the value of Specialization
ENDDO
CREATE String getter method for getAcademicQualifications()
DO
    Return the value of AcademicQualifications
ENDDO
CREATE int getter method for getPerformanceIndex()
DO
    Return the value of PerformanceIndex
ENDDO
CREATE boolean getter method for isCertified()
DO
    Return the value of IsCertified
END DO

CREATE void setter method for setSalary(double newSalary, int
newPerformanceIndex)
DO
```

```
      IF IsCertified is false
      DO
      IF newPerformanceIndex is greater than 5 AND getWorkingHours is greater than 20
         Do
End do
            Set Salary equals to salary + appraisalAmount
            Set PerformanceIndex to newPerformanceIndex
            Set IsCertified to true
         ELSE
            Print the value of isCertified to "The tutor is uncertified and cannot approved for
salary"
         ENDDO
      ELSE
DO
         Print the value of iscertified to "Certified tutor cannot be modified for salary."
      ENDDO
   ENDDO


   CREATE void method removeTutor()
   DO
      IF IsCertified is false
      DO
         Set Salary to 0
```

```
          Set Specialization to ""
          Set AcademicQualifications to ""
          Set PerformanceIndex to 0
          Set IsCertified to false
      **ELSE**
          Print the value of removeTutoento "Certified tutor cannot be removed."
      **ENDDO**
  **ENDDO**


  **OVERRIDE** the displayTeacherInfo() method
  **DO**
      Call the displayTeacherInfo() method in the superclass (Teacher)
      **IF** IsCertified is true
      **DO**
          Print the value of Salary as  “Salary:”
          Print the value of  Specialization as  “Specialization”
          Print the value of Academic Qualifications as “AcademicQualifications”
**ENDDO**
**ENDDO**
**ENDDO**
```

4 METHOD DISCRIPTION

In java, a method is the collection of statements that perform a specific task. It is a block of code within a class that can be executed when called. There are various method used in this coursework. The brief description of each method is given below:

4.1 Method Description of Teacher Class:

Table 4:Method Descriptions of Teacher Class

Method Name	Descriptions	Type
Teacher (int teacherId, String TeacherName, String Address, String WorkingType, String EmploymentStatus)	This is the constructor for the teacher's class. In Java, a constructor is a special type of method that is used for initializing objects when they are created. It consists of method type, parameters, and initialization. The parameters in this constructor are teacher id which accepts integer value, teacherName, address, workingType and employmentStatus accepts Strings value.	Constructor
getTeacherID()	The mentioned method is the accessor method with a return type of “int” The purpose of this method is to retrieve the teacher ID and return it.	Data type/Integer
getTeacherName()	An accessor method with a return type of string which retrieves the teacher name and return it as string value.	Data type/String
getAddress()	This is an accessor method with a return type of ‘string’ which retrieves the address of teacher and returns it	Data type/Strings
getworkingType()	This is an accessor method with a return type of ‘string’. Its purpose is to retrieve the teacher's working type and	Data type/String

	it return the string value that represents the teacher's working type	
getEmploymentStatus()	This is an accessor method with a return type of 'string', indicating that it retrieves and returns a string value for employment status for teacher.	Data type/String
getWorkinHours()	This a an accessor method with a return type of 'int', indicating that it retrieves and returns an integer value for the teacher's working hours.	Data type/Integer
setWorkingHours()	This is a mutator method with a 'void' return type that does not return any values.it purpose is to assign working hours for teacher.	Data type/Void
DisplayTeacherInfo()	This is a display method with a 'void' return type. Its purpose is to display teacher information. It includes conditional statements to check if the 'workingHours' attribute is greater than 0 it prints the working hours. Otherwise, it will print statements "working hours not assigned"	Data type/Void

4.2 Method Descriptions of Lecturer class:

Table 5:Method Descriptions of Lecturer Class

Method Name	Descriptions	Type
Lecturer (int TeacherID, String teacherName, string Address, String workingType, String EmploymentStatus, String Department, int YearofExperiences, int workingHours)	This constructor initializes a new instance of the lecturer class using the super keyword to call the constructor of the teacher class. It also sets the values of additional attributes in the lecturer class. The additional attributes in the lecturer's class are departments with the String data type, year of experience, and graded score with the integer data type, and has graded with Boolean	Constructor
getDepartment()	This is an accessor method that retrieves the department of lecturer and returns the department of lecturer and its return type is String.	Data type/String
getYearOfExperiences()	This is an accessor method that retrieves the year of experience of the lecturer and returns the year of experience for the lecture and its return type is an integer	Data type/Integer
getGradedScore()	This is an accessor method that retrieves the graded score assigned to the lecturer and	Data type/Integer

	returns the graded score as an integer	
gethasGraded()	An accessor method that retrieves the status of whether the lecturer has graded the assignments of students. The return type of this method is Boolean, and it returns true if the assignments have been graded and it returns false if the assignments have not been graded.	Data type/Boolean
setGradedScore()	A mutator method that sets the obtained graded score for the lecturer. It takes a new graded score as a parameter and updates the gradedScore attributes.	Mutator/setter method.
gradeAssignment(int gradedscore, String Department, int yearsOfExperiences)	This is the void method type that is used for grading assignments. Grades are assigned based on certain conditions. If the lecturer has not graded yet, it checks the year of experience and department of lecture before assigning a grade to the student's assignment. After that, the result is printed on the terminal	Constructor
DisplayTeacherInfo()	This is a display method in a lecture class. It is used to display	Void display method

	information in the lecturer's class. It first calls the display method of the superclass using ‘super.DisplayTeacherInfo()’ and then it adds the additional attributes of the lecturer class and also displays the additional attributes. It also check whether grading has occurred and display the corresponding information	

4.3 Method Description for Tutor class:

Table 6:Method Description of Tutor class

Name of the method	Description	Data Type
Tutor (int teacherID, String teacherName, String address, String workingType, String employmentStatus, int workinHours, Double Salary, String Specialization, string academicQualification, Int performanceIndex)	This is the constructor for tutor class and is used for initializing the attributes of tutor class. It also call the constructor of super class. It also initializes additional attributes specific to tutor class such as salary & and performance index with integer data type, specialization and academic qualification with strings data type, and certification of tutor with Boolean data type.	Constructor
getSalary()	This is an accessor method that retrieves the salary for the tutor and returns it. Its return type is integer.	Data type/Double
getspecialization()	This is an accessor method that retrieves the specialization of the tutor and returns it with a return type of string. ,	Data type/Strings
getacademicQualification()	This is an accessor method that returns the academic qualification of the tutor.	Data type/
getPerformanceIndex()	This is an accessor method that returns the performance index of a tutor.	
isCertified()	An accessor method that returns the certification status of the tutor	
setSalary()	A mutator or setter method that is used to set the salary of the tutor based on certain conditions. It accepts a new	

	salary and a new performance index. The method first checks if the tutor is not certified and working hours and performance index does not meet certain criteria, it calculates a salary appraisal on the performance index. After that new salary is updated and the certification status is changed.	
removeTutor()	This method is used to remove the tutor if he/she is not certified. The method first checks if the tutor is not certified then it resets some of the attributes, if the tutor is certified it displayed; the certified tutor cannot remove	
DisplayTeacherInfo	This method is overridden from the superclass and is used to display details of the tutor including the additional attributes of a tutor. It displays the common attributes of teacher class. If the tutor is certified the additional attributes such as salary, specialization, academic qualification, and performance index.	

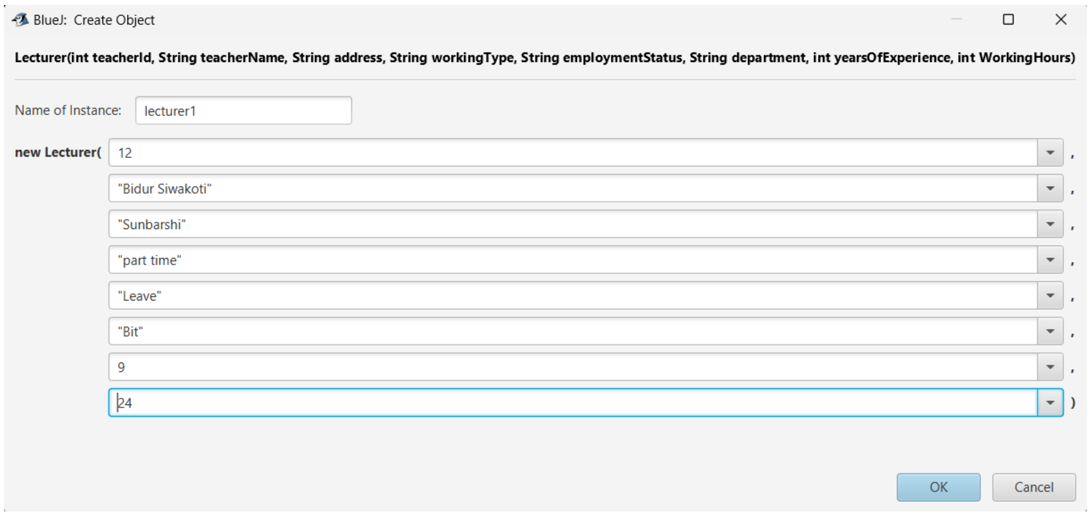
5 Testing

Testing of the program is the process of making the program error free. To make our program well perform we must test our program multiple times. If the testing is done in proper way it will remove all the errors from our program. Testing involves executing the program under controlled conditions and comparing the actual results with expected results to ensure the quality of the program. In this coursework, four tests are carried out.

5.1 Test 1

Objective	To inspect the lecturer's class, grade the assignment and re-inspect the Lecturer's class.
Action	First, we enter all the details of lecture class and inspect it. Then, we grade the assignment based on certain conditions and finally re-inspect it to see the updated info.
Excepted Result	It should show details as I entered.
Actual Result	First display as input and also update as enter.
Test Result	Test successful

Below are the screenshots involved in this test:

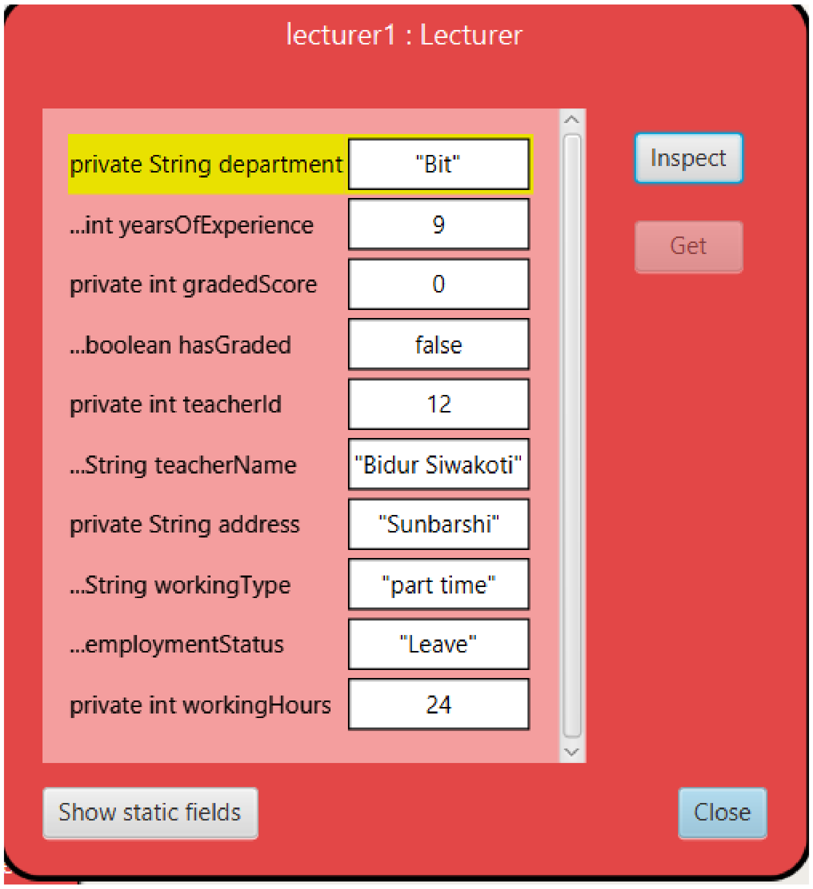

BlueJ: Create Object

Lecturer(int teacherId, String teacherName, String address, String workingType, String employmentStatus, String department, int yearsOfExperience, int WorkingHours)

Name of Instance: lecturer1

```
new Lecturer( 12 ,
"Bidur Siwakoti" ,
"Sunbarshi" ,
"part time" ,
"Leave" ,
"Bit" ,
9 ,
24 )
```

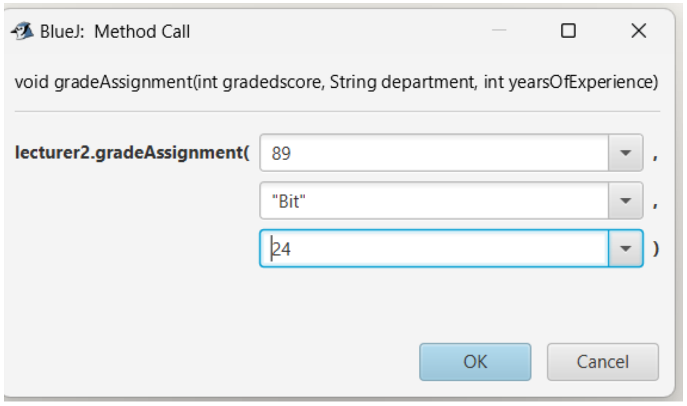
Figure 6:Initial object creation



lecturer1 : Lecturer

private String department	"Bit"
...int yearsOfExperience	9
private int gradedScore	0
...boolean hasGraded	false
private int teacherId	12
...String teacherName	"Bidur Siwakoti"
private String address	"Sunbarshi"
...String workingType	"part time"
...employmentStatus	"Leave"
private int workingHours	24

Figure 7:Inspecting tutor class.

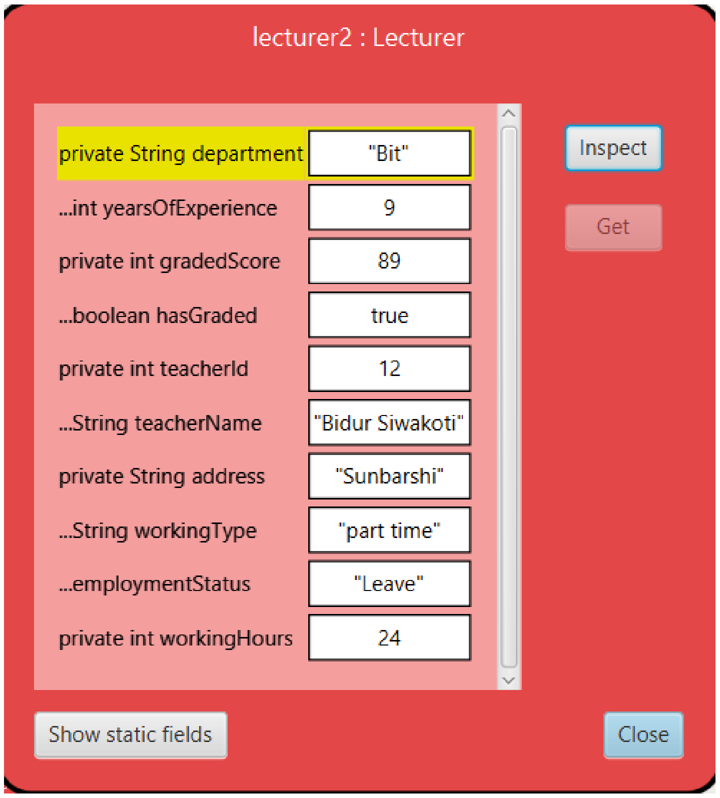


BlueJ: Method Call

void gradeAssignment(int gradedscore, String department, int yearsOfExperience)

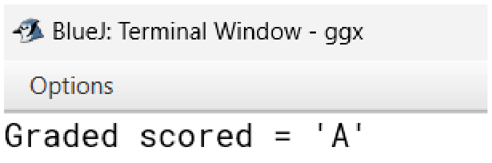
lecturer2.gradeAssignment(89 , "Bit" , 24)

Figure 8:calling grade assignment method



private String department	"Bit"
...int yearsOfExperience	9
private int gradedScore	89
...boolean hasGraded	true
private int teacherId	12
...String teacherName	"Bidur Siwakoti"
private String address	"Sunbarshi"
...String workingType	"part time"
...employmentStatus	"Leave"
private int workingHours	24

Figure 9:Final inspection



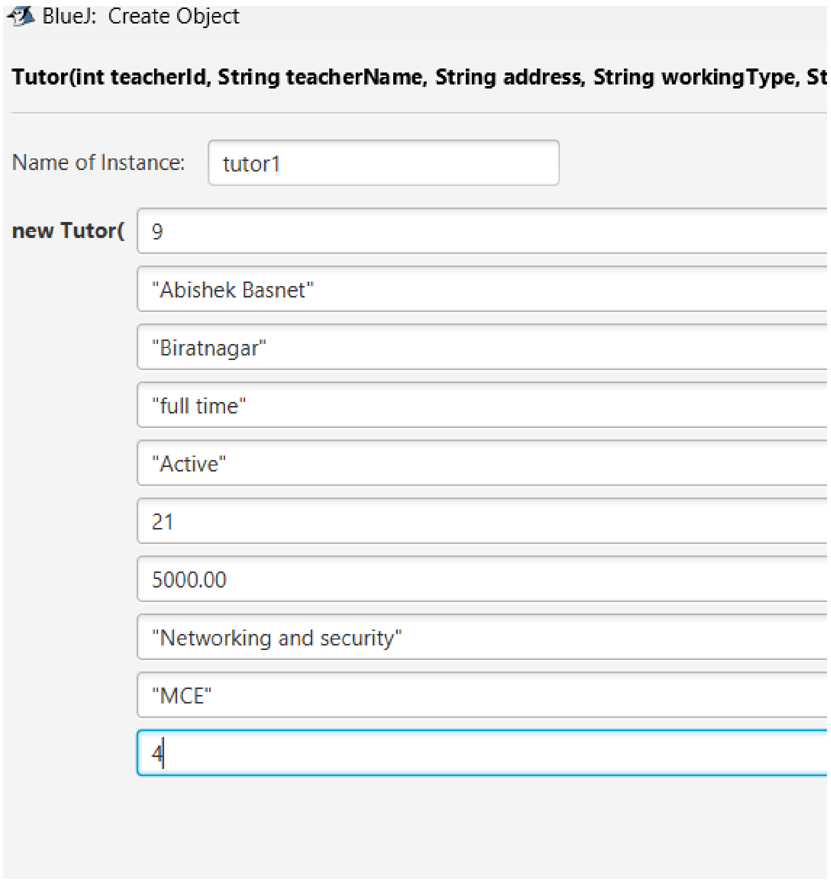
```
Graded scored = 'A'
```

Figure 10:Grades for Assignments

5.2 Test 2

Objective	To inspect Tutor class, set the salary for tutor and re-inspect the tutor class.
Action	First, we enter all the details of lecture class and inspect it. Then, we set salary for tutor based on certain conditions. After it we reinspect it and new salary is set for the tutor according to his/her performance index.
Excepted Result	It should show details as I entered.
Actual Result	It displays as input.
Test Result	Test successful.

Below are the screenshots involved in this test:



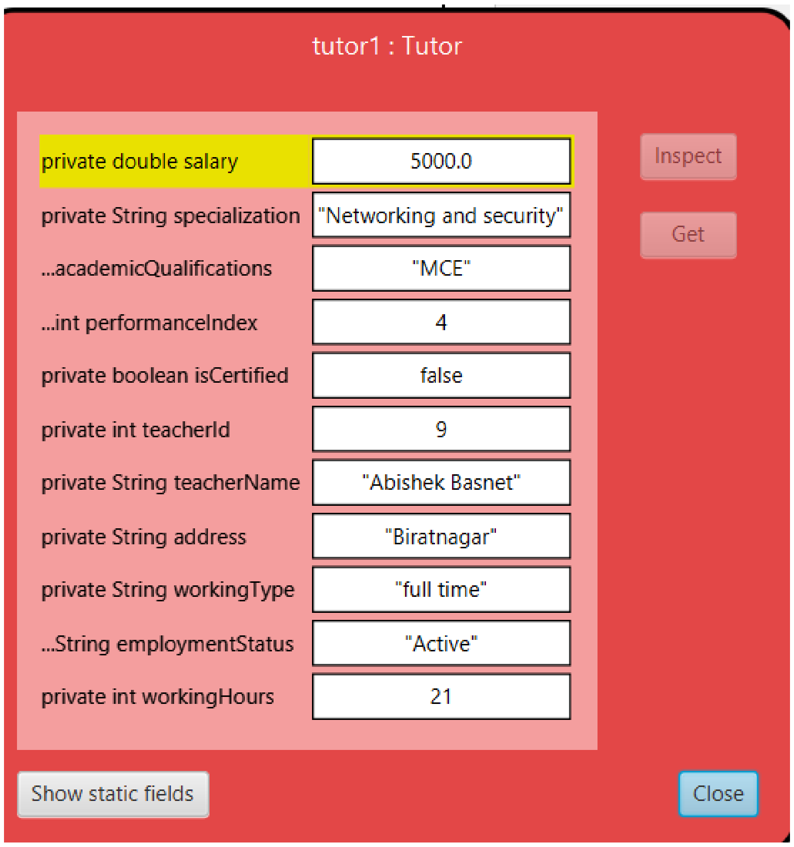
BlueJ: Create Object

Tutor(int teacherId, String teacherName, String address, String workingType, St

Name of Instance: tutor1

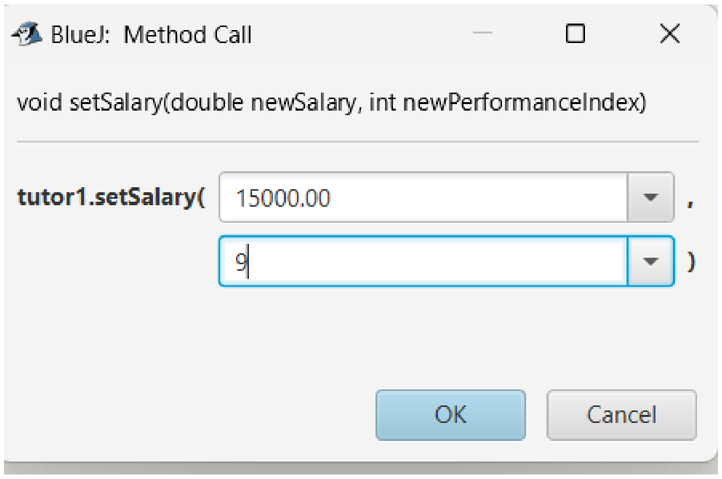
new Tutor(9
"Abishek Basnet"
"Biratnagar"
"full time"
"Active"
21
5000.00
"Networking and security"
"MCE"
4

Figure 11:Object creation of tutor class.


tutor1 : Tutor

private double salary	5000.0
private String specialization	"Networking and security"
...academicQualifications	"MCE"
...int performanceIndex	4
private boolean isCertified	false
private int teacherId	9
private String teacherName	"Abishek Basnet"
private String address	"Biratnagar"
private String workingType	"full time"
...String employmentStatus	"Active"
private int workingHours	21

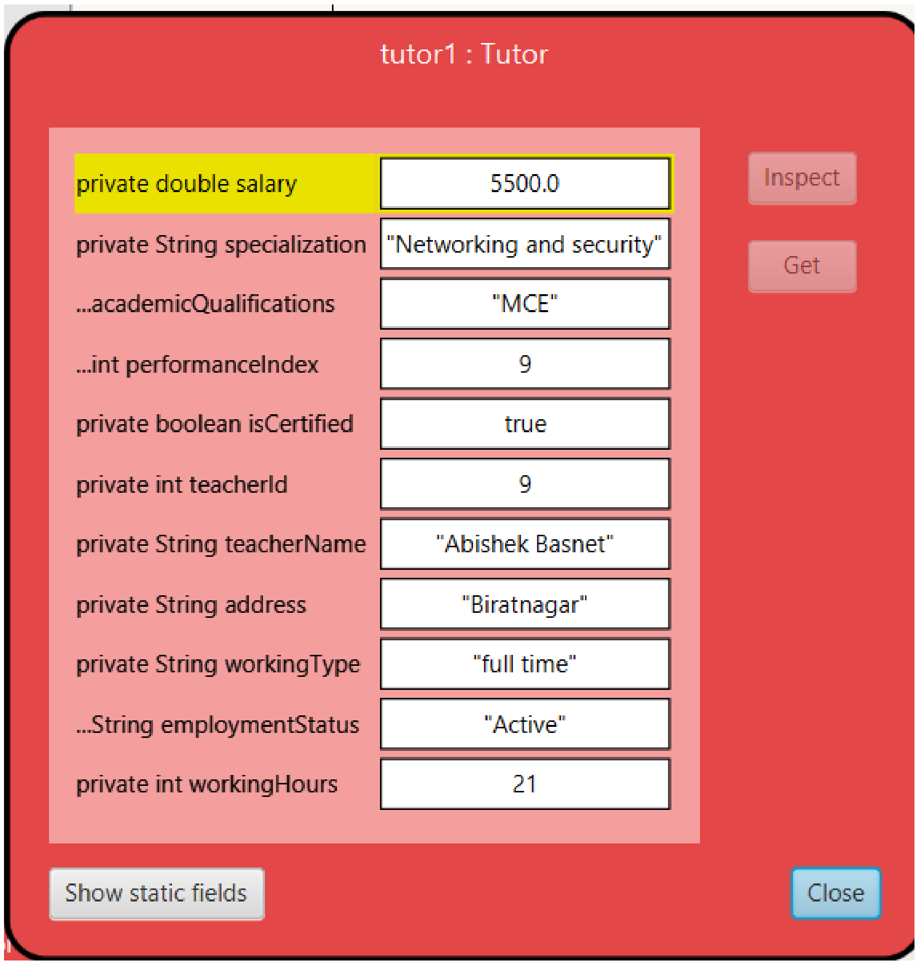
Figure 12:Inspection of tutor class


BlueJ: Method Call

void setSalary(double newSalary, int newPerformanceIndex)

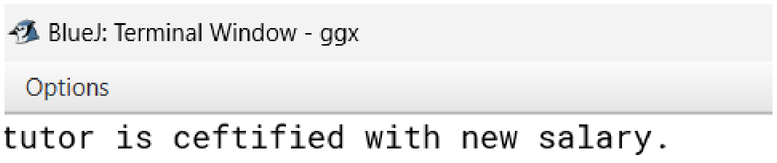
tutor1.setSalary(15000.00 , 9)

Figure 13:Assigning of salary.



private double salary	5500.0
private String specialization	"Networking and security"
...academicQualifications	"MCE"
...int performanceIndex	9
private boolean isCertified	true
private int teacherId	9
private String teacherName	"Abishek Basnet"
private String address	"Biratnagar"
private String workingType	"full time"
...String employmentStatus	"Active"
private int workingHours	21

Figure 14:Reinspection of tutor class.



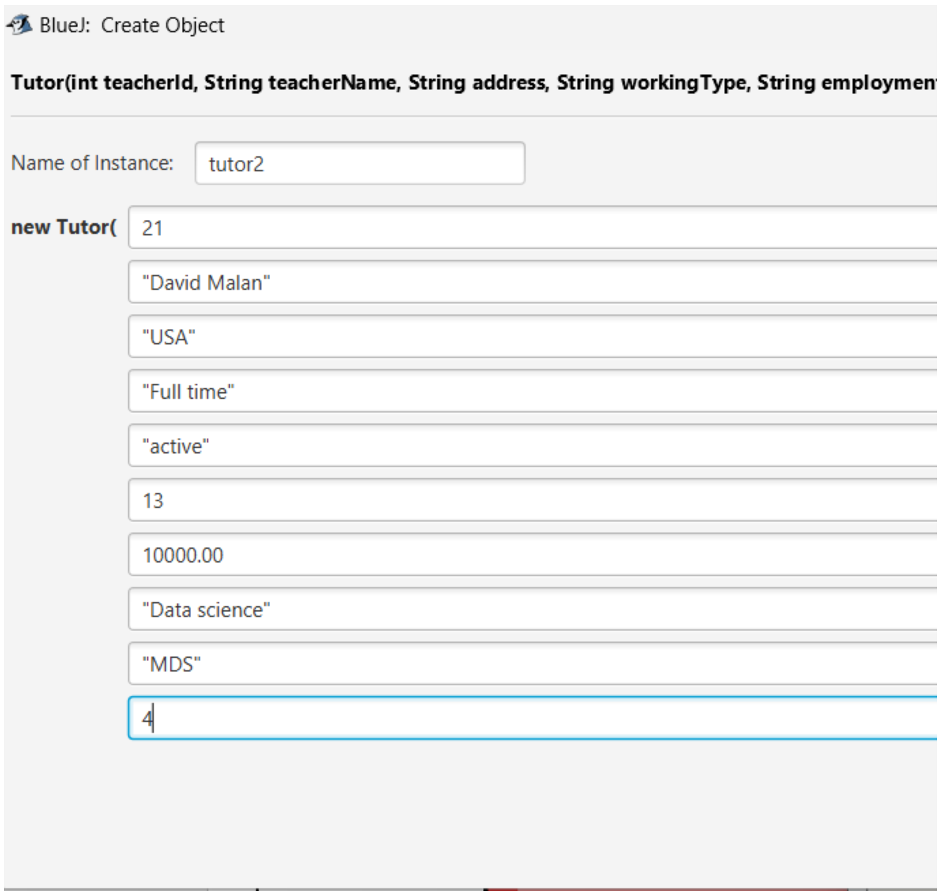
```
tutor is ceftified with new salary.
```

Figure 15:Final output

5.3 Test 3

Objective	To inspect the tutor class after removing the tutor.
Action	First enter all the details of tutor class and run void remove method.
Excepted Result	It should remove salary, specialization, academic qualifications and performance index should be set to zero
Actual Result	Yes, it removes above attributes.
Test Result	Test successful

Below are the screenshots involved in this test:

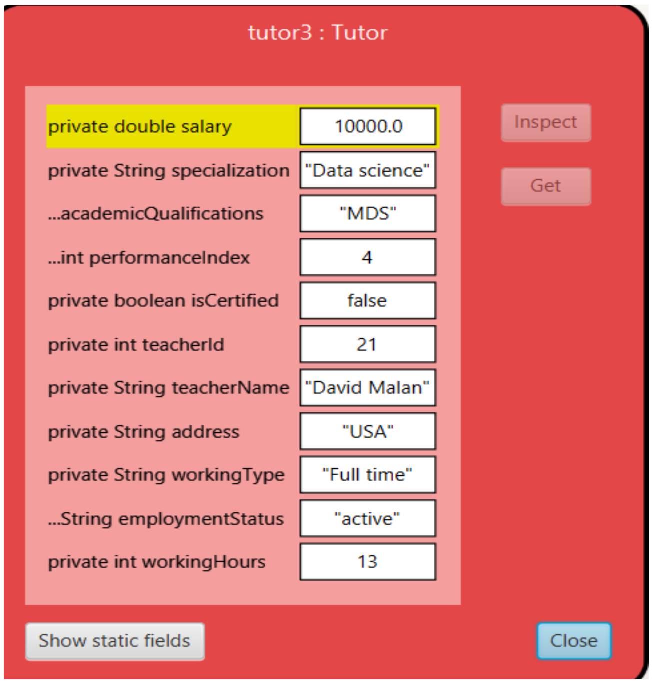

BlueJ: Create Object

Tutor(int teacherId, String teacherName, String address, String workingType, String employmen

Name of Instance: tutor2

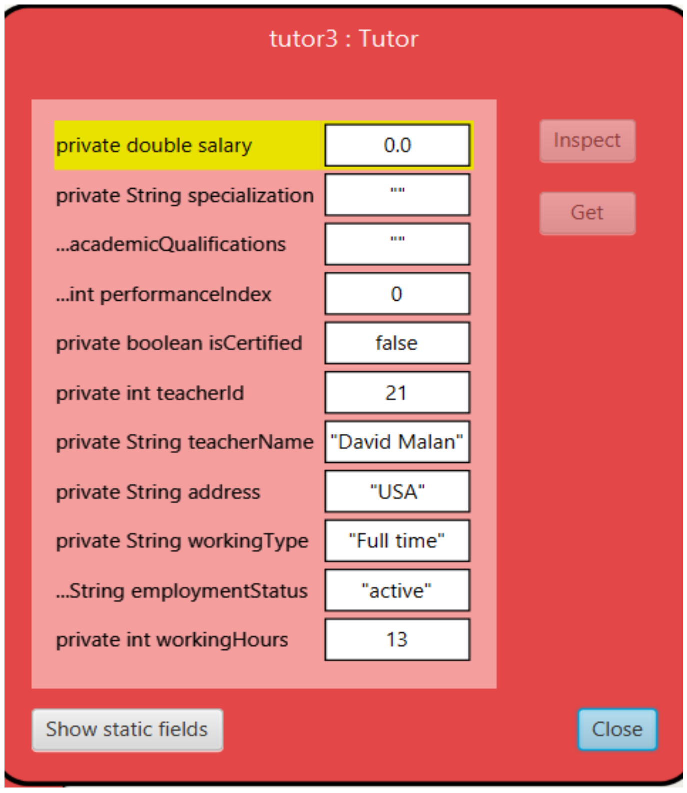
new Tutor(
21
"David Malan"
"USA"
"Full time"
"active"
13
10000.00
"Data science"
"MDS"
4

Figure 16:creation of object for test 3.



private double salary	10000.0
private String specialization	"Data science"
...academicQualifications	"MDS"
...int performanceIndex	4
private boolean isCertified	false
private int teacherId	21
private String teacherName	"David Malan"
private String address	"USA"
private String workingType	"Full time"
...String employmentStatus	"active"
private int workingHours	13

Figure 17:initial inspection



private double salary	0.0
private String specialization	""
...academicQualifications	""
...int performanceIndex	0
private boolean isCertified	false
private int teacherId	21
private String teacherName	"David Malan"
private String address	"USA"
private String workingType	"Full time"
...String employmentStatus	"active"
private int workingHours	13

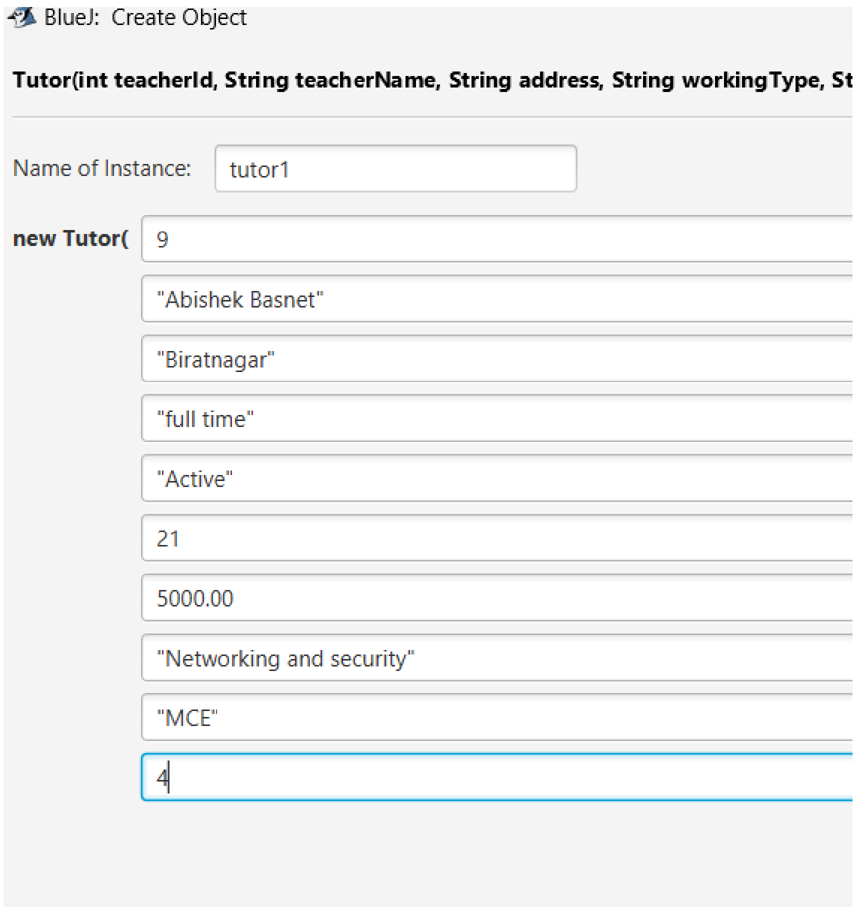
Figure 18:Inspection after remove method

5.4 Test 4

Objective	To display the details of Lecture and Tutor class.
Action	First enter all the details in teacher and lecture class and display it using display method.
Excepted Result	It should display all the details in lecture and tutor class.
Actual Result	It displays all the details of lecture and tutor class.
Test Result	Test successful

Below are the screenshots involved in this test:

For tutor class:

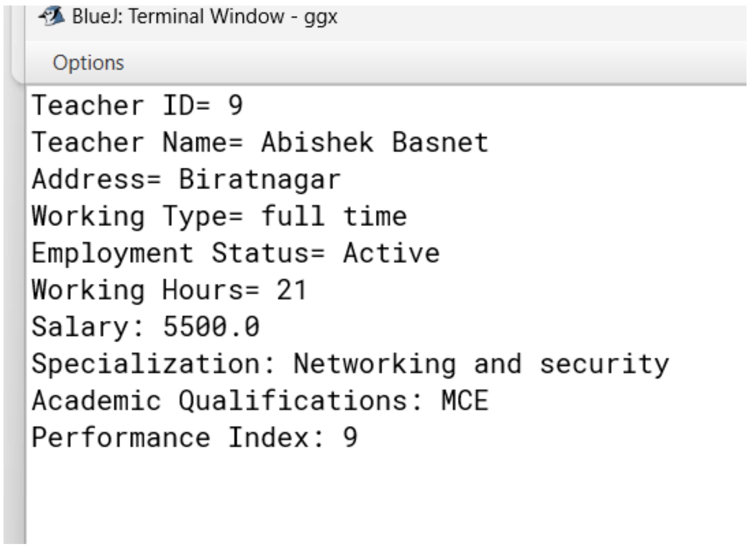

BlueJ: Create Object

Tutor(int teacherId, String teacherName, String address, String workingType, St

Name of Instance: tutor1

new Tutor(9
"Abishek Basnet"
"Biratnagar"
"full time"
"Active"
21
5000.00
"Networking and security"
"MCE"
4

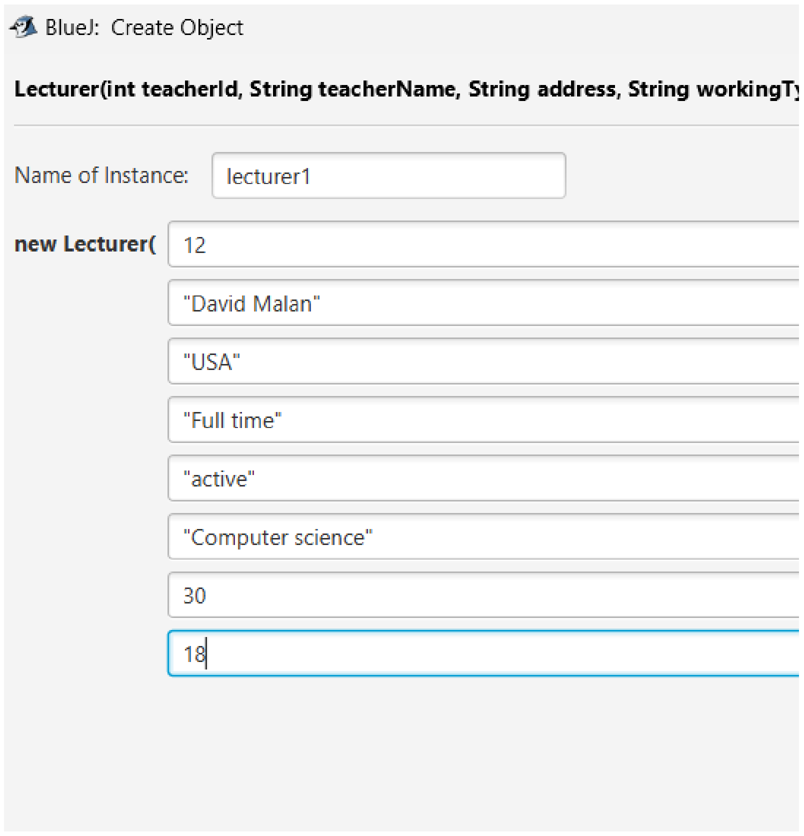
Figure 19:creating object for displaying details.



```
Teacher ID= 9
Teacher Name= Abishek Basnet
Address= Biratnagar
Working Type= full time
Employment Status= Active
Working Hours= 21
Salary: 5500.0
Specialization: Networking and security
Academic Qualifications: MCE
Performance Index: 9
```

Figure 20:Details of tutor class.

For Lecturer class:



Lecturer(int teacherId, String teacherName, String address, String workingTy

Name of Instance: lecturer1

new Lecturer(12

"David Malan"

"USA"

"Full time"

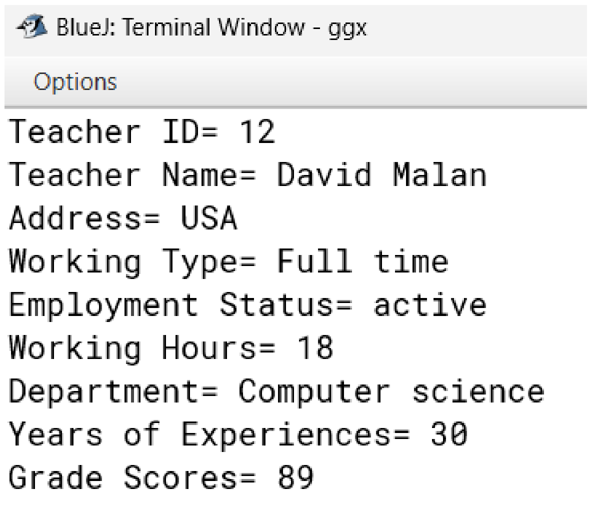
"active"

"Computer science"

30

18

Figure 21:creating object for Lecturer class.



```
Teacher ID= 12
Teacher Name= David Malan
Address= USA
Working Type= Full time
Employment Status= active
Working Hours= 18
Department= Computer science
Years of Experiences= 30
Grade Scores= 89
```

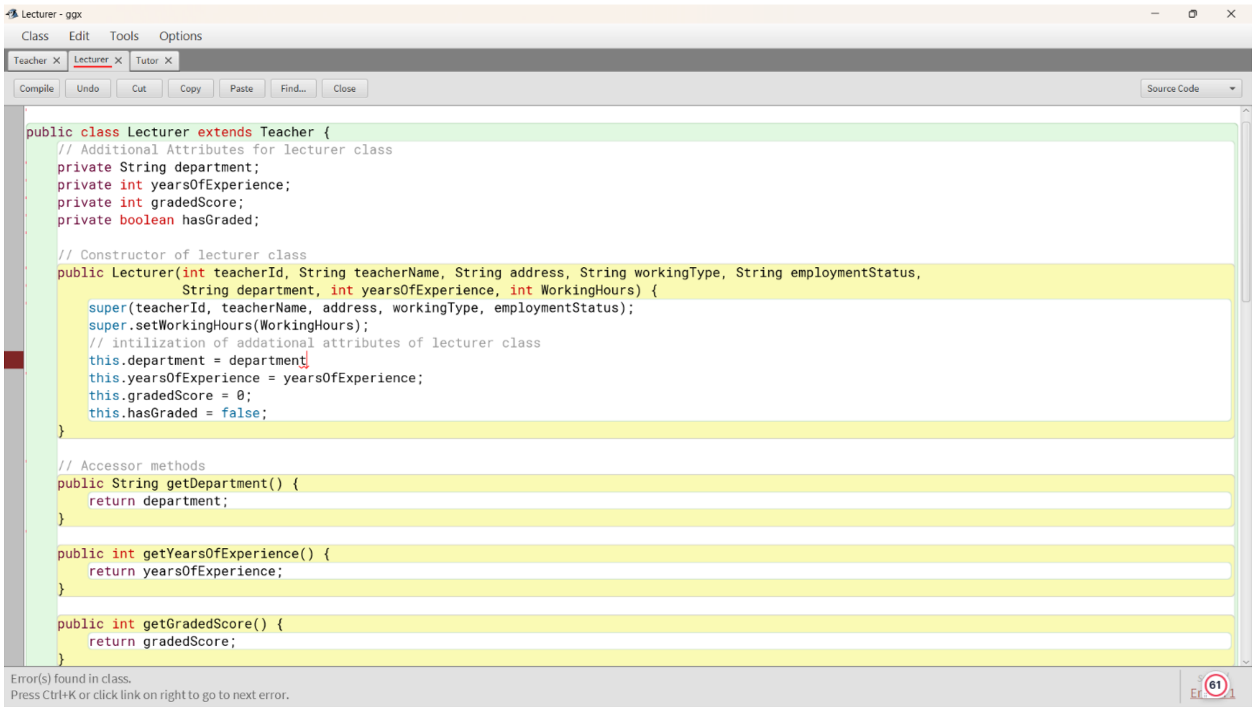
Figure 22:Details of Lecturer class.

6 ERROR

Programming errors are those errors that occur while developing a program that produces undesired outcomes. I have faced several challenges logical errors, syntax errors, and semantic errors. A brief description of every type of error with a screenshot is given below.

6.1 Syntax errors:

A syntax error in programs occurs when the code does not match to actual syntax of the program. It means that the structure of the code is incorrect according to the syntax of the programming language. Such errors are usually detected by the compiler. Missing commas, semicolons, misspelling keywords, etc. are examples of syntax errors.

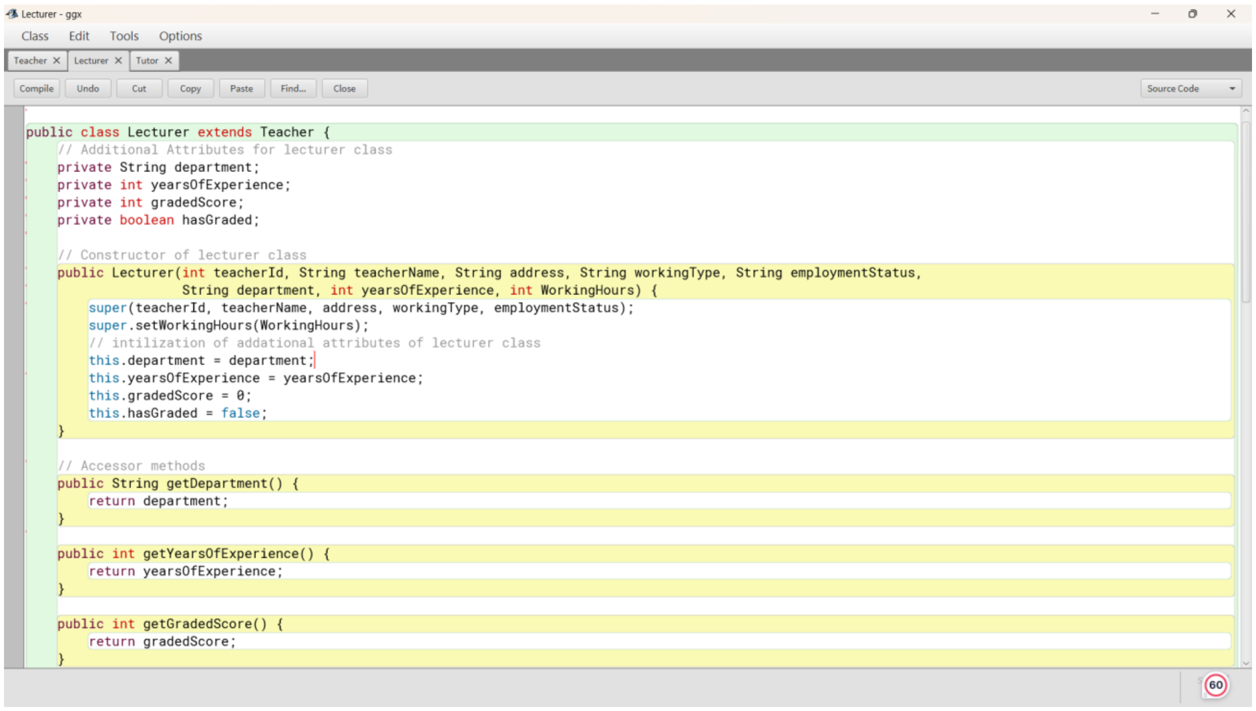


```
public class Lecturer extends Teacher {
    // Additional Attributes for lecturer class
    private String department;
    private int yearsOfExperience;
    private int gradedScore;
    private boolean hasGraded;

    // Constructor of lecturer class
    public Lecturer(int teacherId, String teacherName, String address, String workingType, String employmentStatus,
                    String department, int yearsOfExperience, int WorkingHours) {
        super(teacherId, teacherName, address, workingType, employmentStatus);
        super.setWorkingHours(WorkingHours);
        // intilization of addational attributes of lecturer class
        this.department = department
        this.yearsOfExperience = yearsOfExperience;
        this.gradedScore = 0;
        this.hasGraded = false;
    }

    // Accessor methods
    public String getDepartment() {
        return department;
    }

    public int getYearsOfExperience() {
        return yearsOfExperience;
    }

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

```
Error(s) found in class.
Press Ctrl+K or click link on right to go to next error.
```

Figure 23:Syntax Error

In this code, there is no semi-colon that’s why the compiler shows this error. We can fix this error as:

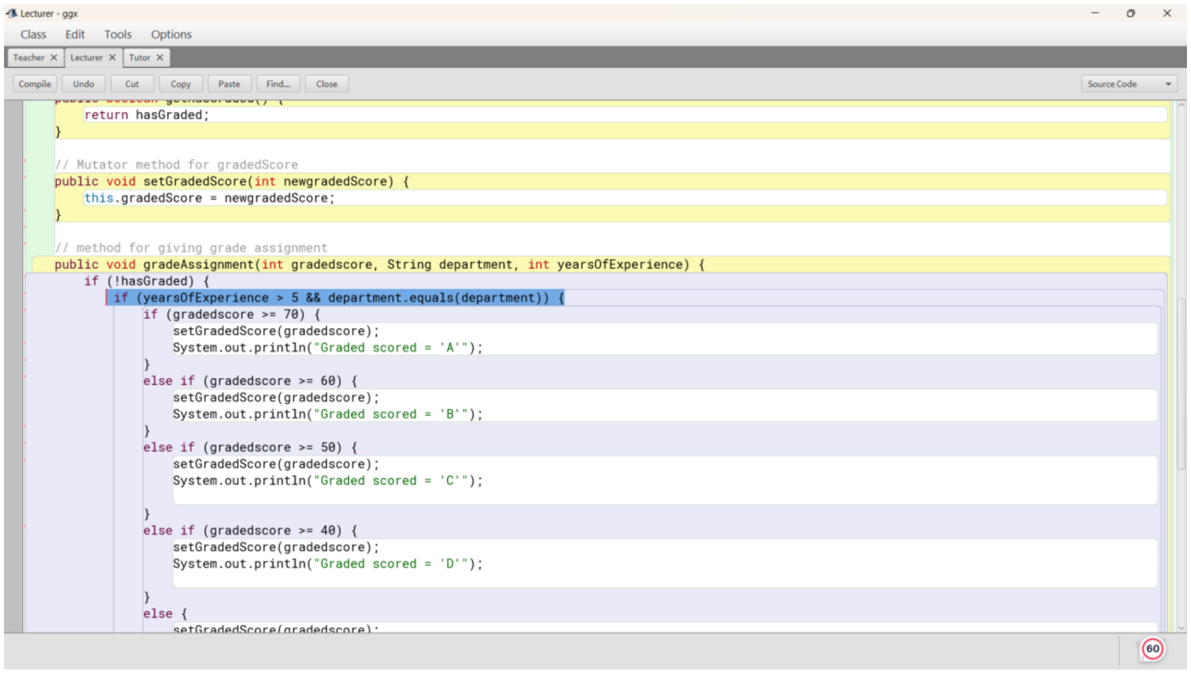


```
public class Lecturer extends Teacher {
    // Additional Attributes for lecturer class
    private String department;
    private int yearsOfExperience;
    private int gradedScore;
    private boolean hasGraded;

    // Constructor of lecturer class
    public Lecturer(int teacherId, String teacherName, String address, String workingType, String employmentStatus,
                    String department, int yearsOfExperience, int WorkingHours) {
        super(teacherId, teacherName, address, workingType, employmentStatus);
        super.setWorkingHours(WorkingHours);
        // intilization of addational attributes of lecturer class
        this.department = department;
        this.yearsOfExperience = yearsOfExperience;
        this.gradedScore = 0;
        this.hasGraded = false;
    }

    // Accessor methods
    public String getDepartment() {
        return department;
    }

    public int getYearsOfExperience() {
        return yearsOfExperience;
    }

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

Figure 24:Correction of syntax error

6.2 Semantics error

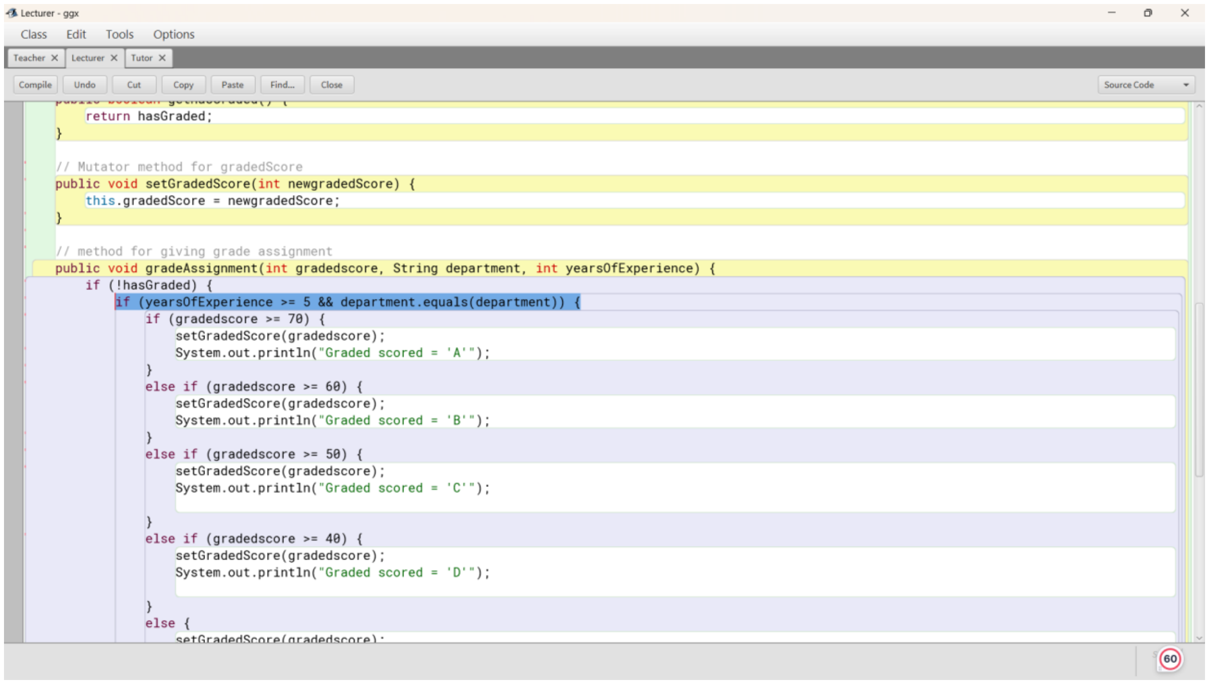
A semantics error in programming occurs when the code runs without producing any syntax error it might be a logical error. Unlike syntax errors semantics errors can be compiled by compiler but it does not produce the required or desired output. these errors are very difficult to detect because the code compiles and executes without any apparent issues.



```
        return hasGraded;
    }

    // Mutator method for gradedScore
    public void setGradedScore(int newgradedScore) {
        this.gradedScore = newgradedScore;
    }

    // method for giving grade assignment
    public void gradeAssignment(int gradedscore, String department, int yearsOfExperience) {
        if (!hasGraded) {
            if (yearsOfExperience > 5 && department.equals(department)) {
                if (gradedscore >= 70) {
                    setGradedScore(gradedscore);
                    System.out.println("Graded scored = 'A'");
                }
                else if (gradedscore >= 60) {
                    setGradedScore(gradedscore);
                    System.out.println("Graded scored = 'B'");
                }
                else if (gradedscore >= 50) {
                    setGradedScore(gradedscore);
                    System.out.println("Graded scored = 'C'");

                }
                else if (gradedscore >= 40) {
                    setGradedScore(gradedscore);
                    System.out.println("Graded scored = 'D'");

                }
                else {
                    setGradedScore(gradedscore);
```

Figure 25:Semantics error

In highlighted line the program is write in the context of syntax but logically it is wrong as the experiences should be greater than equal to five. We can fix this as

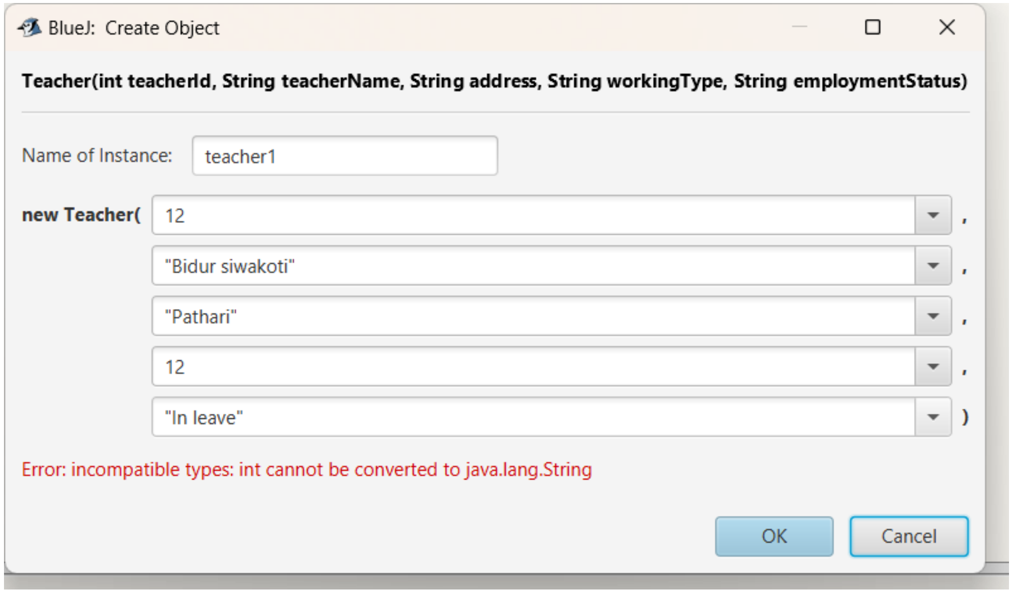


```
        return hasGraded;
    }

    // Mutator method for gradedScore
    public void setGradedScore(int newgradedScore) {
        this.gradedScore = newgradedScore;
    }

    // method for giving grade assignment
    public void gradeAssignment(int gradedscore, String department, int yearsOfExperience) {
        if (!hasGraded) {
            if (yearsOfExperience >= 5 && department.equals(department)) {
                if (gradedscore >= 70) {
                    setGradedScore(gradedscore);
                    System.out.println("Graded scored = 'A'");
                }
                else if (gradedscore >= 60) {
                    setGradedScore(gradedscore);
                    System.out.println("Graded scored = 'B'");
                }
                else if (gradedscore >= 50) {
                    setGradedScore(gradedscore);
                    System.out.println("Graded scored = 'C'");

                }
                else if (gradedscore >= 40) {
                    setGradedScore(gradedscore);
                    System.out.println("Graded scored = 'D'");

                }
                else {
                    setGradedScore(gradedscore);
```

Figure 26:correction of semantics error

6.3 Run-time error

Runtime error is those error that occurs during the execution of the program. Run time errors are not detected by the Java compiler. Java virtual machines detect it while the program is running.



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

Name of Instance: teacher1

new Teacher(12 ,
"Bidur siwakoti" ,
"Pathari" ,
12 ,
"In leave" **)**

Error: incompatible types: int cannot be converted to java.lang.String

Figure 27:Run-time error

7 CONCLUSION

Finally, I reached to conclusion part of this coursework, the closing section of this documentation. At the beginning of this coursework, the project seemed a bit confusing. I even didn’t understand the questions of this coursework but with the help of my module leader, tutor, friends, and the internet I got it. In this project, I have to make three classes. They are teacher lectures and tutors where the teacher is the superclass, and the lecturer and tutor class are child classes.

I have also deal with different types of errors like syntax, semantics, logical, and run-time errors. It teaches me how to understand error messages, review code more carefully. After the completion of this coursework figuring out and fixing errors became an important part of my programming skills. The most interesting and important part of this coursework is testing because it gives me a piece of practical programming knowledge.

Finally, it was a very nice coursework that taught me many things that I had not learned before. This module has been a great learning journey for me. I have gained a lot of new knowledge that I never knew before. I am really grateful for this module and the teacher. At last, I appreciate the entire course for providing me with this valuable learning experience.

8 References

Anon., n.d. *Pseudocode in Programming* **|** *Definition, Examples & Advantages.* [Online] Available at: https://study.com/learn/lesson/pseudocode-examples-what-is-pseudocode.html
[Accessed 20 01 2024].

GeeksForGeeks, 03 April 2023. *Introduction to java.* [Online] Available at: https://www.geeksforgeeks.org/introduction-to-java/
[Accessed 24 01 2024].

GeeksForGeeks, 18 jan 2024. *Class diagram and UML diagram.* [Online] Available at: https://www.geeksforgeeks.org/unified-modeling-language-uml-class-diagrams/#uml-class-notation
[Accessed 20 01 2024].

Study.com, n.d. *Pseudocode in programming langauge.* [Online] Available at: https://study.com/learn/lesson/pseudocode-examples-what-is-pseudocode.html
[Accessed 23 01 2024].

9 APPENDIX

9.1 Source code for teacher class:

```
 public class Teacher {

// Attributes of Teacher class

private int teacherId;

private String teacherName;

private String address;

private String workingType;

private String employmentStatus;

private int workingHours; //  the default value of working hours is 0

// Constructor of teacher class

public Teacher(int teacherId, String teacherName, String address, String workingType, String employmentStatus) {

    this.teacherId = teacherId;

    this.teacherName = teacherName;

    this.address = address;

    this.workingType = workingType;

    this.employmentStatus = employmentStatus;

}

// Accessor methods

public int getTeacherId() {

    return teacherId;

}

public String getTeacherName() {

    return teacherName;
```

```
}
public String getAddress() {
    return address;
}

public String getWorkingType() {
    return workingType;
}

public String getEmploymentStatus() {
    return employmentStatus;
}

public int getWorkingHours() {
    return workingHours;
}

// Method to seting working hours
public void setWorkingHours(int newWorkingHours) {
    this.workingHours = newWorkingHours;
}
// Display method to display teacher info
public void DisplayTeacherInfo() {
    System.out.println("Teacher ID= " + this.teacherId);
    System.out.println("Teacher Name= " + this.teacherName);
    System.out.println("Address= " + this.address);
```

```
        System.out.println("Working Type= " + this.workingType);
        System.out.println("Employment Status= " + this.employmentStatus);


        if (workingHours > 0) {
            System.out.println("Working Hours= " + workingHours);
        } else {
            System.out.println("Working Hours= Not assigned");
        }
    }
}
```

9.2 Source code for lecturer class:

```
public class Lecturer extends Teacher {
    // Additional Attributes for lecturer class
    private String department;
    private int yearsOfExperience;
    private int gradedScore;
    private boolean hasGraded;


    // Constructor of lecturer class
    public Lecturer(int teacherId, String teacherName, String address, String workingType, String
employmentStatus,
                    String department, int yearsOfExperience, int WorkingHours) {
        super(teacherId, teacherName, address, workingType, employmentStatus);
        super.setWorkingHours(WorkingHours);
        // intilization of addational attributes of lecturer class
        this.department = department;
        this.yearsOfExperience = yearsOfExperience;
        this.gradedScore = 0;
        this.hasGraded = false;
    }
    public String getDepartment() {
        return department;
    }

    public int getYearsOfExperience() {
        return yearsOfExperience;
    }

    public int getGradedScore() {
        return gradedScore;
    }

    public boolean gethasGraded() {
        return hasGraded;
    }

    // Mutator method for gradedScore
    public void setGradedScore(int newgradedScore) {
        this.gradedScore = newgradedScore;
    }
    // method for giving grade assignment

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

        if (!hasGraded) {

            if (yearsOfExperience >= 5 && department.equals(department)) {

                if (gradedscore >= 70) {

                    setGradedScore(gradedscore);

                    System.out.println("Graded scored = 'A'");

                }

                else if (gradedscore >= 60) {

                    setGradedScore(gradedscore);

                    System.out.println("Graded scored = 'B'");

                }

                else if (gradedscore >= 50) {

                    setGradedScore(gradedscore);

                    System.out.println("Graded scored = 'C'");


                }

                else if (gradedscore >= 40) {

                    setGradedScore(gradedscore);

                    System.out.println("Graded scored = 'D'");


                }

                else {

                    setGradedScore(gradedscore);

                    System.out.println("Graded scored = 'E'");
                }


                this.hasGraded = true;
            } else {
                System.out.println("Lecturer cannot grade assignments for this student.");
            }
        } else {
            System.out.println(" The assignments is  already  graded.");
}}
    // overriding the display method to add new attributes
    @Override
    public void DisplayTeacherInfo() {
        super.DisplayTeacherInfo(); // Calling the display methodof the  superclass
        //displaying the addtional attributes of this class
        System.out.println("Department= " + this.department);
        System.out.println("Years of Experiences= " + this.yearsOfExperience);
        if (hasGraded) {
            System.out.println("Grade Scores= " + this.gradedScore);
        } else {
            System.out.println("Grade Scores= Score has not graded till now.");
        }
    }
}
```

9.3 Source code for tutor class

```
public class Tutor extends Teacher {
    // Adding additional attributes to tutor class

    private double salary;

    private String specialization;

    private String academicQualifications;

    private int performanceIndex;

    private boolean isCertified;


    // making constructor for tutuor class

    public Tutor(int teacherId, String teacherName, String address, String workingType, String
employmentStatus,

                int workingHours, double salary, String specialization, String academicQualifications, int
performanceIndex) {

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

        super.setWorkingHours(workingHours);

        //  initializationing the additional attributes

        this.salary = salary;

        this.specialization = specialization;

        this.academicQualifications = academicQualifications;

        this.performanceIndex = performanceIndex;

        this.isCertified = false;

    }


    // Accessor methods

    public double getSalary() {

        return salary;

    }
```

```
public String getSpecialization() {
    return specialization;
}

public String getAcademicQualifications() {
    return academicQualifications;
}

public int getPerformanceIndex() {
    return performanceIndex;
}

public boolean isCertified() {
    return isCertified;
}

// Mutator method for salary
public void setSalary(double newSalary, int newPerformanceIndex) {
    if (!isCertified) {
        if (newPerformanceIndex > 5 && super.getWorkingHours() > 20) {
            double appraisalPercentage;

            if (newPerformanceIndex >= 5 && newPerformanceIndex <= 7) {
                appraisalPercentage = 0.05;
            } else if (newPerformanceIndex >= 8 && newPerformanceIndex <= 9) {
                appraisalPercentage = 0.10;
            } else {
                appraisalPercentage = 0.20;
            }

            double appraisalAmount = salary * appraisalPercentage;
            this.salary = salary + appraisalAmount;
            this.performanceIndex = newPerformanceIndex;
            this.isCertified = true;
            System.out.println("tutor is ceftified with new salary. ");
        } else {
            System.out.println("The tutor is uncertified and cannot approved for salary");
        }
    } else {
        System.out.println("Certified tutor cannot be modified for salary.");
    }
}

// Method to remove tutor
public void removeTutor() {
    if (!isCertified) {
        this.salary = 0;
        this.specialization = "";
        this.academicQualifications = "";
        this.performanceIndex = 0;
        this.isCertified = false;
        } else {
            System.out.println("Certified tutor cannot be removed.");
        }
    }

    // Override display method to include additional details
    @Override
    public void DisplayTeacherInfo() {
        super.DisplayTeacherInfo(); // Call the display method in the superclass

        if (isCertified) {
            System.out.println("Salary: " + this.salary);
            System.out.println("Specialization: " + this.specialization);
            System.out.println("Academic Qualifications: " + this.academicQualifications);
            System.out.println("Performance Index: " + this.performanceIndex);
        }
    }

}
```

10 PLAGIARISM TEST REPORT

Originality report

COURSE NAME

Programming 2023 Autumn

STUDENT NAME

IIC Bidur

FILE NAME

NP05CP4A230013_BIDUR_SIWAKOTI

REPORT CREATED

Jan 25, 2024

Summary

Flagged passages	12	4%
Cited/quoted passages	5	2%

Web matches

cliffsnotes.com	5	2%
studocu.com	1	1%
study.com	2	0.8%
geeksforgeeks.org	2	0.5%
javatpoint.com	2	0.4%
unogeeks.com	1	0.3%
prepbytes.com	1	0.3%

lucidchart.com	1	0.2%
quora.com	1	0.2%
codingninjas.com	1	0.2%

1 of 17 passages

Student passage FLAGGED

I confirm that I understand my coursework needs to be submitted online via MySecondTeacher **under the relevant module page before the deadline for my assignment to be accepted and marked**. **I am fully**…

Top web match

Student Name: Saugat Ghimire London Met ID: 18028937 College ID: NP05CP4A Assignment Due Date: 7
Feburary 2021 Assignment Submission Date: 7 Feburary 2021 Title: Hybrid Movie Recommendation System **I**… Recommendation System - I am fully aware that late submissions
... https://www.studocu.com/row/document/itahari-international-college/artificial-intelligence/recommendationsystem/76014271

2 of 17 passages

Student passage FLAGGED

…language, it uses the concept of class and object. **In Java, a class serves as a blueprint or template that defines the attributes** and behaviors **of** a certain class where **objects**

Top web match

In Java, a class serves as a blueprint or template that defines the structure, behavior, and **attributes of objects.** It acts as a logical entity ...

Class and object in Java - PrepBytes https://www.prepbytes.com/blog/java/class-and-object-in-java/

3 of 17 passages

Student passage FLAGGED

The Teacher class serves as the superclass or the foundational blueprint, consisting of six essential **attributes: Teacher ID, Teacher Name, Address, working type, employment status**, and **working hours**…

Top web match

16 marks) 3) **The** Tutor class is a subclass of **Teacher class** and has five **attributes**: salary - a double specialization - a String academic qualifications - a String performanceIndex - an Integer…

1 1st Sit Coursework 1 Question Paper Year Long 2023 2024 ... https://www.cliffsnotes.com/tutorsproblems/Computer-Science/58271968-1-1st-Sit-Coursework-1-Question-Paper-Year-Long-2023-2024-Module/

---

4 of 17 passages

Student passage QUOTED

BlueJ comes with **an integrated debugger that allows** user **to** visualize **their code, inspect variables, and understand the flow of** program execution. Therefore, I use BlueJ as a code…

Top web match

Interactive Debugger: **BlueJ** has **an integrated debugger that allows** users **to** step through **their code**, set breakpoints, **inspect variables, and understand the flow of** their programs.

BlueJ - UnoGeeks https://unogeeks.com/bluej/

---

5 of 17 passages

Student passage FLAGGED

…that represents the static view of an application. It **is used** not just to create executable code for software
applications but also **to visualize, describe**, and **document various** parts **of** a **system**

Top web match

A class diagram **is used to visualize, describe, document various** different aspects **of** the **system**, and also construct executable software code. It shows the ...

UML Class Diagram - Javatpoint https://www.javatpoint.com/uml-class-diagram

---

6 of 17 passages

Student passage FLAGGED

…Java, and C++. The class diagram has three sections. **The** upper section has **the name of the class, the middle** section has all **the attributes, and the** lower **section** has **methods** and **operations**.

Top web match

The top row contains **the name of the class, the middle** row contains **the attributes** of the class, **and the** bottom **section** expresses the **methods** or **operations** that ...

UML Class Diagram Tutorial | Lucidchart https://www.lucidchart.com/pages/uml-class-diagram

---

7 of 17 passages

Student passage CITED

The attributes are written along with its visibility factor

Top web match

The attributes have the following characteristics: **The attributes are written along with its visibility** factors, which are public (+), private (-), protected (#), and package

UML Class Diagram - Javatpoint https://www.javatpoint.com/uml-class-diagram

---

8 of 17 passages

Student passage FLAGGED

Pseudocode is defined as a method of describing a process or writing programming code and algorithm **using natural language such as English**. It is not a code itself, but rather a…

Top web match

Pseudocode is defined as a method of describing a process or writing programming code and algorithms **using** a **natural language such as English**. It is not the ...

Pseudocode in Programming | Definition, Examples & Advantages https://study.com/learn/lesson/pseudocodeexamples-what-is-pseudocode.html

---

9 of 17 passages

Student passage CITED

…code and algorithm using natural language such as English. **It is not** a **code itself, but rather a description of what the code should do**. (Study.com, n.d.) Pseudocode is not complied by computer, its…

Top web match

It is not the **code itself, but rather a description of what the code should do**. ... First, do not use languagespecific commands in your statements. Pseudocode ...

Pseudocode in Programming | Definition, Examples & Advantages https://study.com/learn/lesson/pseudocodeexamples-what-is-pseudocode.html

---

10 of 17 passages

Student passage FLAGGED

In **java**, a **method is** the **collection of statements that perform** a **specific task**. It is a block of code within a class…

Top web match

What is a method in Java programming? **Java Method is** a **collection of statements that perform** some **specific task** and return the result to the caller. 2.

Java Methods - GeeksforGeeks https://www.geeksforgeeks.org/methods-in-java/

---

11 of 17 passages

Student passage FLAGGED

…collection of statements that perform a specific task. It **is a block of code within a class that** can be **executed when** called. There are various method used in this coursework.

Top web match

A static block, also known as a static initialization block, **is a block of code within a class that** is **executed when** the class is first loaded ...

Static blocks in Java - Coding Ninjas https://www.codingninjas.com/studio/library/static-blocks-in-java

---

12 of 17 passages

Student passage CITED

Lecturer (int TeacherID, String teacherName, string Address, String workingType, String EmploymentStatus, String Department, int YearofExperiences, int workingHours)

Top web match

public class Lecturer extends Teacher (private String department; private int yearsOfExperience; private int gradedScore; private boolean hasGraded; public **Lecturer(int teacherId, String teacherName,**…

1 1st Sit Coursework 1 Question Paper Year Long 2023 2024 ... https://www.cliffsnotes.com/tutorsproblems/Computer-Science/58271968-1-1st-Sit-Coursework-1-Question-Paper-Year-Long-2023-2024-Module/

13 of 17 passages

Student passage CITED

Tutor (int teacherID, String teacherName, String address, String workingType, String employmentStatus, int workinHours, **Double Salary, String Specialization, string** academicQualification, **Int**…

Top web match

public class Tutor extends Teacher (private double salary; private String specialization; private String academicQualifications; private int performanceIndex; private boolean isCertified; public …

1 1st Sit Coursework 1 Question Paper Year Long 2023 2024 ... https://www.cliffsnotes.com/tutorsproblems/Computer-Science/58271968-1-1st-Sit-Coursework-1-Question-Paper-Year-Long-2023-2024-Module/

14 of 17 passages

Student passage FLAGGED

…well perform we must test our program multiple times. **If** the **testing is done** in proper way **it will remove all the errors from** our program. Testing involves executing the program under controlled…

Top web match

To make our software perform well it should be error-free. **If testing is done** successfully **it will remove all the errors from** the software.

Types of Software Testing - GeeksforGeeks https://www.geeksforgeeks.org/types-software-testing/

---

15 of 17 passages

Student passage FLAGGED

To **inspect the** lecturer's **class, grade the assignment and re-inspect the** Lecturer's **class**.

Top web match

You should give evidence (through inspection tables and appropriate screenshots) of the following testing that you carried out on your program: Test 1: **Inspect the** Lecturer **class, grade the**…

1 1st Sit Coursework 1 Question Paper Year Long 2023 2024 ... https://www.cliffsnotes.com/tutorsproblems/Computer-Science/58271968-1-1st-Sit-Coursework-1-Question-Paper-Year-Long-2023-2024-Module/

---

16 of 17 passages

Student passage FLAGGED

It should remove **salary, specialization, academic qualifications and performance index** should be **set to zero**

Top web match

The attributes **salary, specialization, academic qualifications and performance index** is **set to zero**. The attribute isCertified is then set to ...

1 1st Sit Coursework 1 Question Paper Year Long 2023 2024 ... https://www.cliffsnotes.com/tutorsproblems/Computer-Science/58271968-1-1st-Sit-Coursework-1-Question-Paper-Year-Long-2023-2024-Module/

---

17 of 17 passages

Student passage FLAGGED

Runtime error is those **error** that **occurs during the execution of** the **program**. Run time errors are not detected by the Java…

Top web match

The **runtime error is** the **error** which **occurs during the execution of** a **program**. In contrast, compile-time errors occur while a program is being ...

How does a runtime error occur? - Quora https://www.quora.com/How-does-a-runtime-error-occur