



Islington college
(इसलिंग्टन कलेज)

CS4001NI Programming

30% Individual Coursework

2023-24 Autumn

Student Name: OM SHANKAR SAH

London Met ID: 23048483

College ID: NP01CP4A230467

Group: Individual

Assignment Due Date: Friday, January 26, 2024

Assignment Submission Date: Thursday, January 25, 2024

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

Table of Contents

INTRODUCTION.....	1
About The Coursework:	1
Tools used:	2
BlueJ:	2
MS-Word:	2
Draw.io:	3
CLASS DIAGRAM.....	4
Teacher Class Diagram	4
Lecturer Class Diagram.....	4
Tutor Class Diagram.....	5
Relationship Class Diagram	6
DESCRIPTION OF ALL METHODS.....	7
Teacher Class:	7
Accessor Method:	7
setWorkingHour():	7
Display():.....	7
Lecturer Class:	7
Accessor Method:	7
setGradedScore():.....	7
gradeAssignment():.....	7
Display():.....	8
Tutor Class:	8
Accessor Method:	8
setSalary():.....	8
removeTutor():	8
Display():.....	8
PSEUDOCODE.....	9
Pseudocode Of Teacher Class:	9
Pseudocode Of Lecturer Class	12
Pseudocode Of Tutor Class.....	18

TESTING.....	23
Test-1: Inspect the Lecturer Class, grade the assignment, and re-inspect the Lecturer Class.....	23
Output Result:	24
Test-2:Inspect Tutor class, set salary and re-inspect the Tutor Class.	26
Output Result:	27
Test-3:Inspect Tutor Class again after removing the tutor.	29
Output Result:	30
TEST-4:Display The Details Of Lecturer and Tutor Class.	31
Result Output:.....	32
ERROR DETECTION AND CORRECTION	35
Syntax Error:.....	35
Correction Of Syntax Error:	35
Semantics Error:.....	36
Correction Of Semantics Error:.....	37
Logical Error:	38
Correction Of Logical Error:.....	39
CONCLUSION	40
Bibliography	42
APPENDIX.....	43
Code Of Teacher.java:.....	43
Code of Lecturer.java:	46
Code Of Tutor.Java	53

Table Of Figures

Figure 1:Teacher Class Diagram	4
Figure 2:Lecturer Class Diagram	4
Figure 3:Tutor Class Diagram	5
Figure 4:Combined Class Diagram	6
Figure 5:Screenshot of assigning the data in Lecturer Class.	24
Figure 6:Screenshot for the inspecting of the Lecturer Class.	24
Figure 7:Sceenshot of assigning data in the gradeAssignment Method.	25
Figure 8:Screenshot of Result of gradeAssignment Method.	25
Figure 9:Screenshot for re-inspecting of Lecturer Class.	25
Figure 10:Screenshot of assigning the data in Tutor class.	27
Figure 11:Screenshot for inspecting of Tutor Class.	27
Figure 12:Screenshot of assigning the data of setSalary Method.	28
Figure 13:Screenshot for re-inspecting of Tutor Class.	28
Figure 14:Screenshot of assigning the data of Tutor Class.	30
Figure 15:Inspecting the Tutor Class.	30
Figure 16:Re-inspecting the Tutor Class after removing The Tutor.	30
Figure 17:Screemshot of assigning the data of Lecturer Class.	32
Figure 18:Screenshot of assigning the data of gradeAssignment Method.	32
Figure 19:Screenshot of assigning the data of setGradedScore Method.	33
Figure 20:Screenshot of graded assignment.	33
Figure 21:Screenshot of displayed Detail of Lecturer Class.	33
Figure 22: Screenshot of assigning the data of setSalary Method	34
Figure 23: Screenshot of assigning the data of Tutor Class	34
Figure 24:Screenshot of displayed the detail of Tutor class.	34
Figure 25:Screenshot of Syntax Error	35
Figure 26:Correction of syntax Error	35
Figure 27:Screenshot of Semantic error.	36
Figure 28:Screenshot of Output of Semantic error.	36
Figure 29:Screenshot of correction of semantic error.	37
Figure 30:Screenshot of Output of Corrected Semantic error	37

Figure 31:Screenshot of Logical Error.....	38
Figure 32:Screenshot of Output of Logical Error	38
Figure 33:Screenshot of Corrected Logical Error	39
Figure 34:Screenshot of Corrected Logical Error Output.	39

Table Of Tables

Table 1:Inspect the Lecturer Class ,grade the assignment , and re-inspect the Lecturer Class. -----	23
Table 2:To inspect Tutor Class, set salary and re-inspect the Tutor Class. -----	26
Table 3:Inspect the Tutor Class again after removing the Tutor. -----	29
Table 4:Display detail of Lecturer and Tutor Class. -----	32

INTRODUCTION

About The Coursework:

In this coursework, we will use the OOP (Object-Oriented Programming) concept of Java to implement a real-world problem scenario that involves Teachers, Lecturer and Tutors. Teachers are professionals who educate students in various subjects and skills. Lecturers are teachers who work in higher education institutions and deliver lecturer to undergraduate or postgraduate students. Tutors are teachers who provide individual or small group instruction to students who needs extra help or guidance.

We will create a class to represent a teacher, together with its two subclass to represent a lecturer and tutor respectively. The teacher class will have some common attributes and methods that are shared by all teachers, such as teacherId , teacherName, workingHour, address, empStatus, workingType and Display(). All the attribute of Teacher class have their own accessor method and only setWorkingHour has setter method . The lecturer class will inherit from Teacher class , such as teacherId, teacherName, address, workingType, empStatus, setter method as setWorkingHour and Display() and have some additional attributes such as, department , yearOfExp, gradedScore, hasGraded and setter method as, setGradedScore (),gradeAssignment() and all the additional attributes of Lecture class has own accessor method.The Tutor class will inherit from Teacher class such as, teacherId, teacherName, address, workingType, empStatus, setter method as setWorkingHour and Display() and have some additional attributes such as salary, speci, acaQual, performIndex, isCertified and setSalary() ,removeTutor() as setter Method.

The lecturer class will override the Display() method of Teacher class with print syntax of department , yearOfExp and if statement of hasGraded which means if hasGraded is true then all the detail of Teacher class and lecturer class will be printed ,else only detail of teacher class and department and yearOfExp will be printed.The setter method i.e gradeAssignment() of lecturer class have if statement which helps to grade assignment according to the grade Scored.The Tutor method also override the Display() method of teacher class which has additional of if statement which checks if the tutor is certified or

not and print the detail of tutor according to it. There are two setter method of tutor method i.e setSalary() which increase the salary of tutor according to the performance index and another method is removeTutor() which remove the detail of tutor if the tutor is not certified.

Tools used:

BlueJ:

BlueJ is a window based platform for java Development Kit(JDK) that is designed to support learning and teaching of object-Oriented programming(OOP) in java. It is free java environment that was started in 1999 by Michael Kolling and Jhon Rosenberg at Monash university, Australia, as a successor to Blue which allows us to create and interact with objects and classes in a graphical and interactive way.

In this coursework, we will use BlueJ to implement a real-world problem scenario that involves teachers, lecturer and tutors . We will create a class to represent a teacher, together with its two subclass to represent a lecturer and tutor respectively.

MS-Word:

Microsoft Word, or MS Word , is a word processor software developed by Microsoft that allows you to create and edit professional quality documents, letters, reports , resumes, etc. It is one of the most widely used programs of the Microsoft Office suite , and it is available for both Window and macOS platforms. It has many features and tools that can help us to format, style, insert, and layout your documents in the best possible way. MS word is a powerful and versatile software that can help us to create professional and impressive documents for various purposes.

In this coursework, we will use MS-word to create ,edit,format, and style our report according to the guidelines given by our instructor. We will also use Ms word to insert tables, diagrams and its Output. We will also use MS-word to add references and citations to our report using the appropriate style.

Draw.io:

Draw.io is a free and cross-platform online tools for making diagrams and charts. It allows us to choose from automatic layout function, to create a custom layout. It has a large selection of shapes and hundreds of visual elements to make your diagram or chart one of a kind. It is a powerful and easy to use tool that help to create professional and impressive diagrams and chart for various purposes.

In this coursework, we will use draw.io to create and edit diagrams that illustrate our program and its output. We will use it to create a class diagram that shows the relationship between the teacher, lecturer and tutor classes. We will also use it to insert these diagrams into our MS-word report.

CLASS DIAGRAM

Teacher Class Diagram

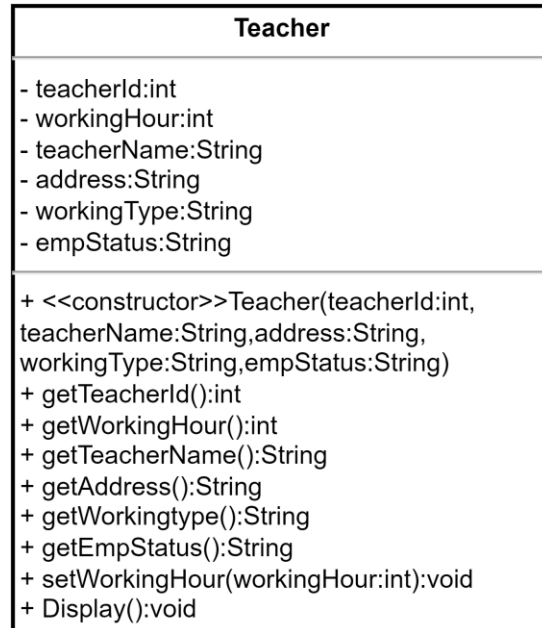


Figure 1:Teacher Class Diagram

Lecturer Class Diagram

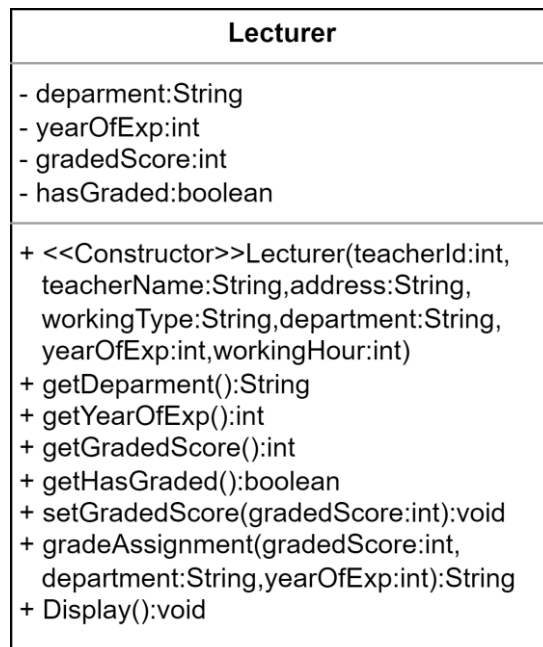


Figure 2:Lecturer Class Diagram

Tutor Class Diagram

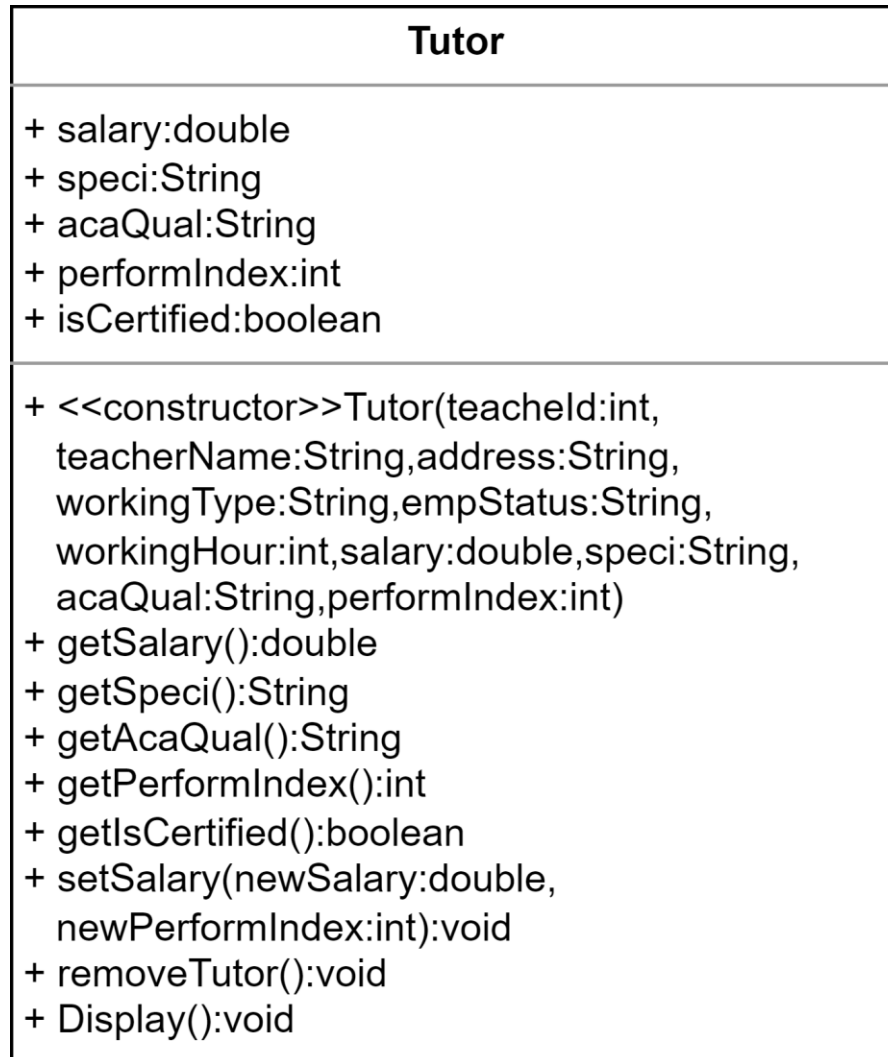


Figure 3:Tutor Class Diagram

Relationship Class Diagram

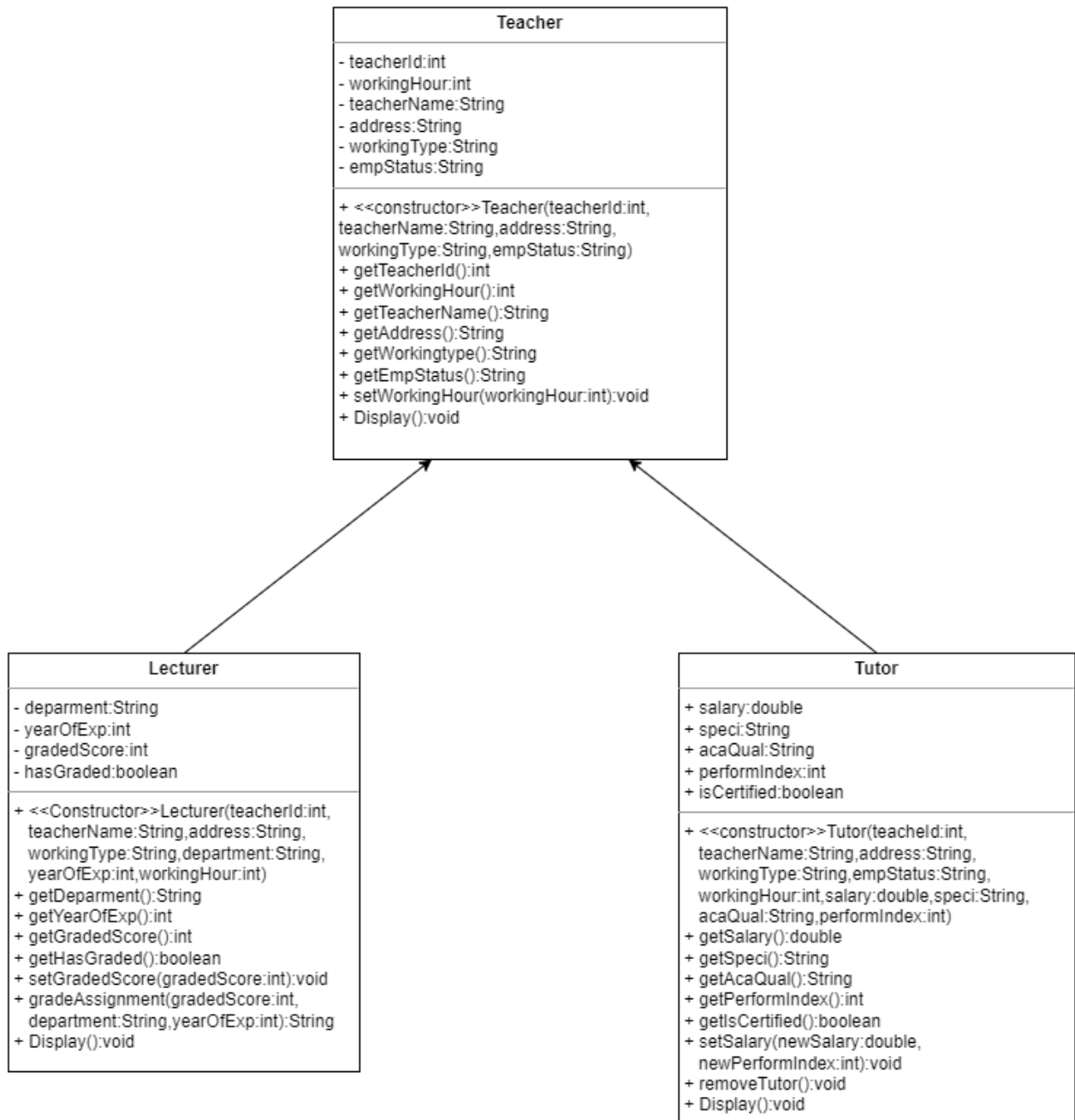


Figure 4: Combined Class Diagram

DESCRIPTION OF ALL METHODS

Teacher Class:

Accessor Method:

The Teacher Class contains accessor methods for all its attributes. These all accessor methods helps to access the data which is private.

setWorkingHour():

This method helps to update the data of workingHour attributes which is used as parameter and also private.

Display():

This method is used to print all the details or data of Teacher class such as TeacherId, teacherName, address, workingtype, empStatus and there used if statement to check if the workingHour is set or not .If the workingHour is set then it will print the data od workingHour ,else it will print the suitable messge as "Teacher working Hour isn't assigned yet".

Lecturer Class:

Accessor Method:

The Lecturer Class also contains accessor methods of all its additional attributes and these method helps to access the data which is private.

setGradedScore():

This method helps to update the data of gradedScore attribute which is passed with parameter as well . It also set the hasGraded to true.

gradeAssignment():

This method helps to assign the grade according to Graded Score which accept the parameters as gradedScore ,department, and yearOfExp. The Lecturer can assign the grade only on the condition if the lecturer have more than 5 year of experience and their department must be same otherwise lecturer wont be qualified for assigning the grade.If

the lecturer is qualified then they can assign the grade according to the graded score. It also updates the value of the gradedScore, yearOfExp and set the hasGraded to true.

Display():

In lecturer class, the Display() method is inherited from the teacher class and override which additionally print the department, yearOfExp and there is if condition which check if hasGraded is true or false, if hasGraded is true it will print GradedScore else suitable message is printed i.e "score has not been graded yet".

Tutor Class:

Accessor Method:

All the additional attributes of tutor class have accessor method which allow us to access the data of respective attributes.

setSalary():

This method accept two parameters i.e newSalary, newPerformanceIndex. In this method the salary of the tutor is appraised according to the performance index of the tutor. If the tutor falls in the condition of appraisal salary then salary is appraised and salary is updated and isCertified is set to true else, the suitable message is printed i.e "The Tutor isn't certified yet so that salary can't be appraised".

removeTutor():

In this Method, if the tutor is not certified then their data of salary is set to zero, specification, academic qualification is set to null, performance index is set to zero and isCertified is set to false. If certified then suitable message is printed i.e "The Tutor is certified and Salary is Appraised".

Display():

This method is inherited from the teacher class which is overridden which print the detail of the tutor class accordingly. If the tutor is certified then all the detail teacher class and tutor class is printed else only the detail of teacher class is printed.

PSEUDOCODE

Pseudocode Of Teacher Class:

CREATE a parent class Teacher

DO

DECLARE instance variable teacherID as int using Private as Access modifier.

DECLARE instance variable workingHour as int using Private as Access modifier.

DECLARE instance variable teacherName as String using Private as Access
Modifier.

DECLARE instance variable address as String using Private as Access modifier.

DECLARE instance variable workingType as String using Private as Access
modifier.

DECLARE instance variable empStatus as String using Private as Access modifier.

DEFINE the constructor Teacher(teacherId as Int,teacherName as String,address as
String,workingType as String,empStatus as String) as parameter

DO

SET this.teacherId to teacherId

SET this.teacherName to teacherName

SET this.address to address

SET this.workingType to workingType

SET this.empStatus to empStatus

SET this.workingHour to 0.

END DO

CREATE an accessor method as getTeacherId() with return type int

DO

RETURN teacherId

END DO

CREATE an accessor method as getWorkingHour() with return type int

DO

RETURN workingHour

END DO

CREATE an accessor method as getTeacherId() with return type int

DO

RETURN teacherId

END DO

CREATE an accessor method as getTeacherName() with return type String

DO

RETURN teacherName

END DO

CREATE an accessor method as getAddress() with return type String

DO

RETURN address

END DO

CREATE an accessor method as getWorkingType() with return type String

DO

RETURN workingType

END DO

CREATE an accessor method as getEmpStatus() with return type String

DO

RETURN empStatus

END DO

CREATE an setter method as setWorkingHour() with return type void

DO

SET this.workingHour to workingHour

END DO

DEFINE method as Display() as return type void

DO

Print the teacherId

Print the teacherName

Print the address

Print the workingType

Print the empStatus

IF workingHour is zero **THEN**

Print teacher working hour isn't assigned

ELSE

Print the workingHour

END IF

END DO

END DO

Pseudocode Of Lecturer Class

CREATE subclass Lecturer of Parent class Teacher

DO

DECLARE the instance variable department as String using Private as access modifier.

DECLARE the instance variable yearOfExp as int using Private as access modifier

DECLARE the instance variable gradedScore as int using Private as access modifier

DECLARE the instance variable hasGraded as boolean using Private as access modifier

DEFINE a constructor Lecturer(teacherId as int, teacherName as String, address as String, workingType as String, empStatus as String, department as String, yearOfExp as int, workingHour as int) as parameter

DO

GET the Teacher constructor with parameter as(teacherId, teacherName, workingType, empStatus)

SET this.department to department

SET this.yearOfExp to yearOfExp

SET this.gradedScore to zero

SET hasGraded to false

GET setWorkingHour(workingHour) from teacher class.

END DO

CREATE an accessor method as getDepartment() with return type String

DO

RETURN department

END DO

CREATE an accessor method as getYearOfExp() with return type int

DO

RETURN yearOfExp

END DO

CREATE an accessor method as getGradedScore() with return type int

DO

RETURN gradedScore

END DO

CREATE an accessor method as getHasGraded() with return type boolean

DO

RETURN hasGraded

END DO

CREATE an setter method as setGradedScore() with return type void

DO

SET this.gradedScore to gradedScore

SET this.hasGraded to true

END DO

CREATE an method as gradeAssignment with parameter as (gradedScore as int,
department as String, yearOfExp as int) with return type String

DO

IF yearOfExp is greater than 5 and accessor method of getDepartment is equal
to department **THEN**

SET this.hasGraded to true

SET this.gradedScore to gradedScore

SET yearOfExp to yearOfExp

DEFINE a variable grade

IF gradedScore is greater than equal to 70 **THEN**

ASSIGN the value of grade as GradeA

RETURN grade

ELSE IF gradedScore is greater than equal to 60 **THEN**

ASSIGN the value of grade as Grade B

RETURN grade

ELSE IF gradedScore is greater than equal to 50 **THEN**

ASSIGN the value of grade as Grade C

RETURN grade

ELSE IF gradedScore is greater than equal to 40 **THEN**

ASSIGN the value of grade as Grade D

RETURN grade

ELSE

ASSIGN the value of grade as Grade E

RETURN grade

END IF

ELSE

SET this.hasGraded to false

ASSIGN string message as “the lecturer is not qualified to grade a assignment”

RETURN message

END IF

END DO

CREATE a method Display() with return type as void

DO

GET the Display() method from teacher class

Print the department

Print the yearOfExp

IF hasGraded is True **THEN**

Print the gradedScore

ELSE

Print "score has not been Graded Yet"

END IF

END DO

END DO

Pseudocode Of Tutor Class

CREATE a subclass as Tutor of Parent teacher class

DO

DECLARE the instance variable salary as double with private as access modifier.

DECLARE the instance variable speci as String with private as access modifier.

DECLARE the instance variable acaQual as String with private as access modifier.

DECLARE the instance variable performIndex as int with private as access modifier.

DECLARE the instance variable isCertified as boolean with private as access modifier.

DEFINE a constructor Tutor(teacherId as int, teacherName as String, address as String, workingType as String, empStatus as String, workingHour as int, salary as double, speci as String, acaQual as String, performIndex as int) as parameters

DO

GET the constructor of Teacher with parameter as(teacherId, teacherName, address, workingType, empStatus)

SET this.salary to salary

SET this.speci to speci

SET this.acaQual to acaQual

SET this.performIndex to performIndex

SET this.isCertified to isCertified

END DO

DEFINE an accessor method getSalary() with return type double

DO

RETURN salary

END DO

DEFINE an accessor method getSpeci() with return type String

DO

RETURN speci

END DO

DEFINE an accessor method getAcaQual() with return type String

DO

RETURN acaQual

END DO

DEFINE an accessor method getPerfomIndex() with return type int

DO

RETURN performIndex

END DO

DEFINE an accessor method getIsCertified() with return type boolean

DO

RETURN isCertified

END DO

CREATE a method setSalary(newSalary as double, newPerformanceIndex as int) as parameters

DO

IF newPerformanceIndex is greater than 5 and accessor method of getWorkingHour() is greater than 20 **THEN**

DEFINE a variable appraisalPer

IF the value of newPerformanceIndex is between equal to 5 and 7 **THEN**

ASSIGN appraisalPer to 0.05

ELSE IF newPerformanceIndex is equal to between 8 and 9 **THEN**

ASSIGN appraisalPer to 0.1

ELSE

ASSIGN appraisalPer to 0.2

END IF

SET this.salary is equal to newSalary +(appraisalPer *newSalary)

SET this.performIndex to newPerformIndex

SET isCertified to True

ELSE

Print “the tutor isn’t certified yet so that salary cannot be appraised”

END IF

END DO

CREATE a method removeTutor() as return type void

DO

IF isCetified is False **THEN**

SET this.salary to 0

SET this.speci to null

SET this.acaQual to null

SET this.performIndex to 0

SET this.isCertified to false

ELSE

Print “the tutor is certified and salary is appraised”

END IF

END DO

CREATE a method Display() as return type void

DO

IF isCetified is True **THEN**

GET the Display() method from Teacher Class

Print the salary

Print the speci

Print the acaQual

Print the perfomlIndex

ELSE

GET the Display() method from Teacher Class

END IF

END DO

TESTING

Test-1: Inspect the Lecturer Class, grade the assignment, and re-inspect the Lecturer Class.

Test No:	1
Objective:	To inspect Lecturer Class ,grade the assignment , and re-inspect the lecturer Class.
Action:	<ul style="list-style-type: none"> ➤ The lecturer is Called with the following arguments: teacherId=1 teacherName ="KulRaj Poudel" address ="Biratnagar" workingType ="Private" empStatus ="Full-Time" department ="Computing" yearOfExp=5 workingHour=20 ➤ Inspect of the Lecturer Class. ➤ String gradeAssignment is called with the following arguments: gradedScore=80 department ="Computing" yearOfExp=6 ➤ Re-inspect of the Lecturer Class
Expected Result:	The Lecturer would assign the Grade.
Actual Result:	The Lecturer assigned the Grade.
Conclusion:	This test is successful.

Table 1:Inspect the Lecturer Class ,grade the assignment , and re-inspect the Lecturer Class.

Output Result:

The screenshot shows a 'Blue: Create Object' window. At the top, the class signature is displayed: `Lecturer(int teacherId, String teacherName, String address, String workingType, String empStatus, String department, int yearOfExp, int workingHour)`. Below this, the 'Name of Instance:' field contains 'lecturer2'. The 'new Lecturer(' line is followed by a series of input fields: '1' for teacherId, 'KulRaj Poudel' for teacherName, 'Biratnagar' for address, 'Private' for workingType, 'Full-Time' for empStatus, 'Computing' for department, '5' for yearOfExp, and '20' for workingHour. The window ends with 'OK' and 'Cancel' buttons.

Figure 5: Screenshot of assigning the data in Lecturer Class.

The screenshot shows an object inspection window titled 'lecturer1 : Lecturer'. It displays a list of fields with their corresponding values in input boxes. The fields are: '...String department' (value: 'Computing'), 'private int yearOfExp' (value: 5), 'private int gradedScore' (value: 0), '...boolean hasGraded' (value: false), 'private int teacherId' (value: 1), 'private int workingHour' (value: 20), '...String teacherName' (value: 'KulRaj Poudel'), 'private String address' (value: 'Biratnagar'), '...String workingType' (value: 'Private'), and 'private String empStatus' (value: 'Full-Time'). To the right of the list are 'Inspect' and 'Get' buttons. At the bottom, there are 'Show static fields' and 'Close' buttons.

Figure 6: Screenshot for the inspecting of the Lecturer Class.

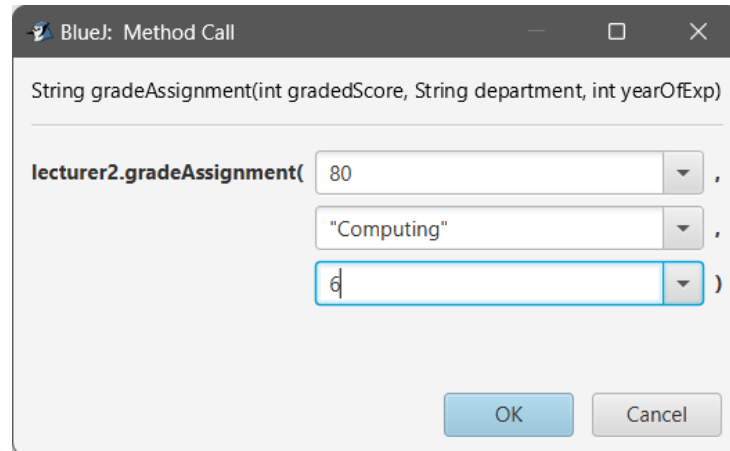


Figure 7: Screenshot of assigning data in the gradeAssignment Method.

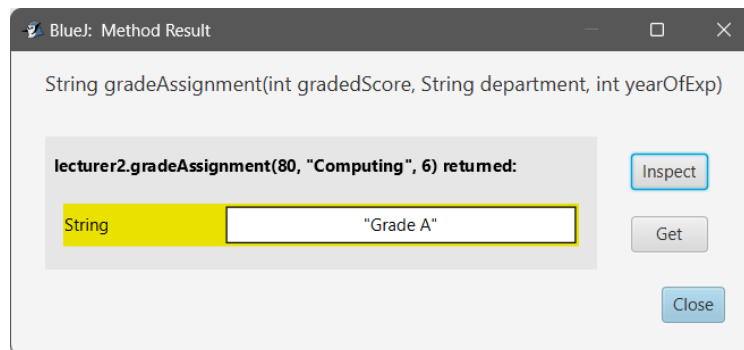


Figure 8: Screenshot of Result of gradeAssignment Method.

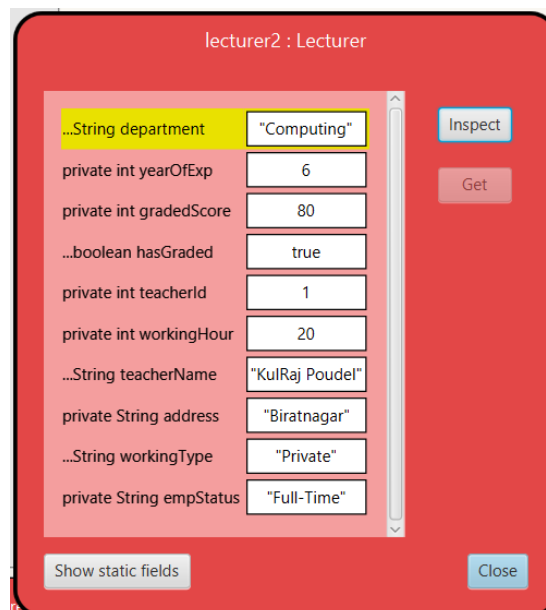


Figure 9: Screenshot for re-inspecting of Lecturer Class.

Test-2:Inspect Tutor class, set salary and re-inspect the Tutor Class.

Test No:	2
Objective:	To inspect Tutor Class, set salary and re-inspect the Tutor Class.
Action:	<ul style="list-style-type: none">➤ The Tutor is Called with the following arguments: teacherId=1 teacherName ="KulRaj Poudel" address ="Biratnagar" workingType ="Private" empStatus ="Full-Time" workingHour=23 salary=5000 speci="Java" acaQuali="Master" performIndex=6➤ Inspect of the Tutor Class.➤ Void setSalary is called with the following arguments: newSalary=50000 newPerformIndex=6➤ Re-inspect of the Tutor Class
Expected Result:	Appraising the salary of Tutor.
Actual Result:	Salary of the Tutor is Appraised.
Conclusion:	This test is successful.

Table 2:To inspect Tutor Class, set salary and re-inspect the Tutor Class.

Output Result:

BlueJ: Create Object

Tutor(int teacherId, String teacherName, String address, String workingType, String empStatus, int workingHour, double salary, String speci, String acaQual, int performIndex)

Name of Instance: tutor2

new Tutor(1 ,
"KulRaj Poudel"
"Biratnagar"
"Private"
"Full-Time"
23
5000
"Java"
"Master"
6)

OK Cancel

Figure 10: Screenshot of assigning the data in Tutor class.

tutor2 : Tutor

private double salary	5000.0
private String speci	"Java"
private String acaQual	"Master"
private int performIndex	6
private boolean isCertified	false
private int teacherId	1
private int workingHour	23
private String teacherName	"KulRaj Poudel"
private String address	"Biratnagar"
private String workingType	"Private"
private String empStatus	"Full-Time"

Inspect Get

Show static fields Close

Figure 11: Screenshot for inspecting of Tutor Class.

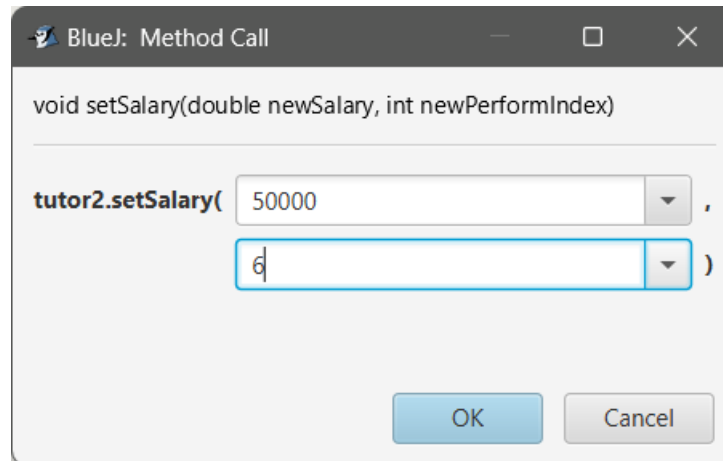


Figure 12: Screenshot of assigning the data of setSalary Method.

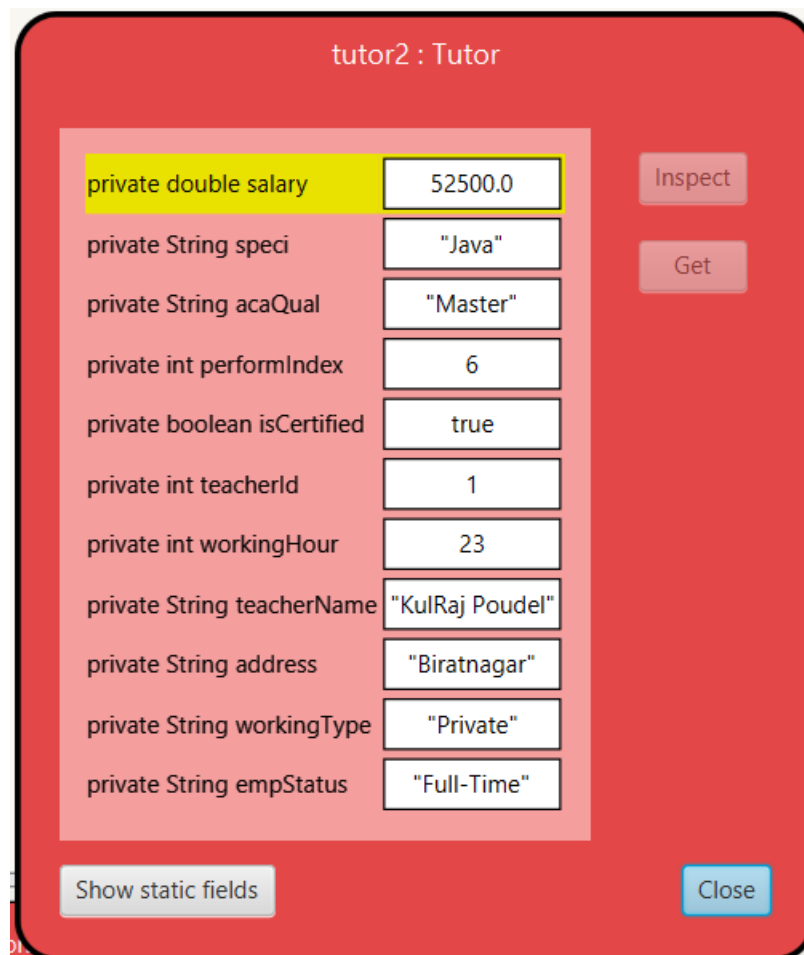


Figure 13: Screenshot for re-inspecting of Tutor Class.

Test-3:Inspect Tutor Class again after removing the tutor.

Test No:	3
Objective:	To inspect Tutor Class again after removing the tutor.
Action:	<ul style="list-style-type: none">➤ The Tutor is Called with the following arguments: teacherId=1 teacherName ="Kul Raj Poudel" address ="Biratnagar" workingType ="Private" empStatus ="Full-Time" workingHour=20 salary=50000 speci="Java" acaQuali="Master" performIndex=3➤ Inspect of the Tutor Class.➤ Void removeTutor is called.➤ Re-inspect of the Tutor Class
Expected Result:	The salary ,performance index is set to zero ,specification and academic qualification is set to null and is certified is set to false.
Actual Result:	The salary ,performance index is set to zero ,specification and academic qualification is set to null and is certified is set to false.
Conclusion:	This test is successful.

Table 3:Inspect the Tutor Class again after removing the Tutor.

Output Result:

BlueJ: Create Object

Tutor(int teacherId, String teacherName, String address, String workingType, String empStatus, int workingHour, double salary, String speci, String acaQual, int performIndex)

Name of Instance:

new Tutor(

<input type="text" value="1"/>	,
<input type="text" value="Kul Raj Poudel"/>	,
<input type="text" value="Biratnagar"/>	,
<input type="text" value="private"/>	,
<input type="text" value="Full-Time"/>	,
<input type="text" value="20"/>	,
<input type="text" value="50000"/>	,
<input type="text" value="Java"/>	,
<input type="text" value="Master"/>	,
<input type="text" value="3"/>)

OK Cancel

Figure 14: Screenshot of assigning the data of Tutor Class.

tutor1 : Tutor

private double salary	50000.0
private String speci	"Java"
private String acaQual	"Master"
private int performIndex	3
private boolean isCertified	false
private int teacherId	1
private int workingHour	20
private String teacherName	"Kul Raj Poudel"
private String address	"Biratnagar"
private String workingType	"private"
private String empStatus	"Full-Time"

Inspect Get

Show static fields Close

Figure 15: Inspecting the Tutor Class.

tutor1 : Tutor

private double salary	0.0
private String speci	""
private String acaQual	""
private int performIndex	0
private boolean isCertified	false
private int teacherId	1
private int workingHour	20
private String teacherName	"Kul Raj Poudel"
private String address	"Biratnagar"
private String workingType	"private"
private String empStatus	"Full-Time"

Inspect Get

Show static fields Close

Figure 16: Re-inspecting the Tutor Class after removing The Tutor..

TEST-4:Display The Details Of Lecturer and Tutor Class.

Test No:	4
Objective:	Display the details of Lecturer and Tutor Class.
Action:	<ul style="list-style-type: none">➤ The Lecturer is Called with the following arguments: teacherId=1 teacherName ="KulRaj Poudel" address ="Biratnagar" workingType ="Private" empStatus ="Full-Time" department="Computing" yearOfExp=7 workingHour=23➤ Void setGradedScore is called with following argument: gradedScore=85➤ String gradeAssignment is called with following argument: gradedScore=85 department="Computing" yearOfExp=7➤ Void Display is called.➤ The Tutor is called with the following arguments: teacherId=1 teacherName ="Kul Raj Poudel" address ="Biratnagar" workingType ="Private" empStatus ="Full-Time" workingHour=26 salary=75000 speci="Java" acaQuali="Master" performIndex=8

	<ul style="list-style-type: none"> ➤ Void setSalary is called with following argument: newSalary=80000 newPerfomanceIndex=8 ➤ Void Display is called.
Expected Result:	All the detail of Lecturer class and Tutor Class is displayed..
Actual Result:	All the detail of Lecturer and Tutor Class is displayed.
Conclusion:	This test is successful.

Table 4:Display detail of Lecturer and Tutor Class.

Result Output:

BlueJ: Create Object

Lecturer(int teacherId, String teacherName, String address, String workingType, String empStatus, String department, int yearOfExp, int workingHour)

Name of Instance: lecturer1

new Lecturer(

- 1
- "Kul Raj Poudel"
- "Biratnagar"
- "Private"
- "Full-Time"
- "Computing"
- 7
- 23

OK Cancel

Figure 17:Screenshot of assigning the data of Lecturer Class.

BlueJ: Method Call

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

lecturer1.gradeAssignment(

- 85
- "Computing"
- 7

OK Cancel

Figure 18:Screenshot of assigning the data of gradeAssignment Method.

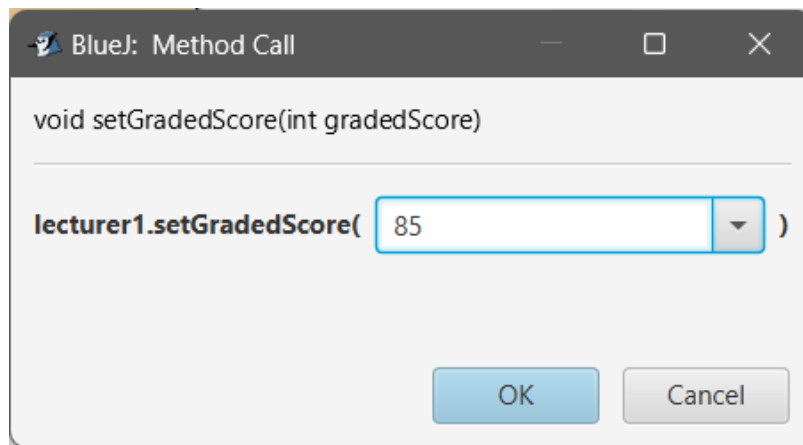


Figure 19: Screenshot of assigning the data of setGradedScore Method.

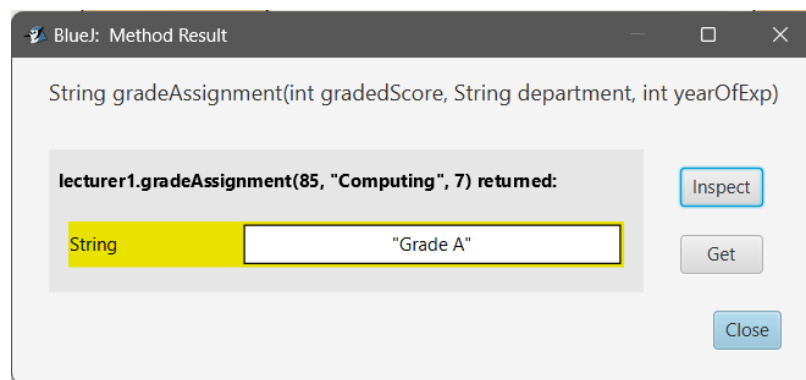


Figure 20: Screenshot of graded assignment.

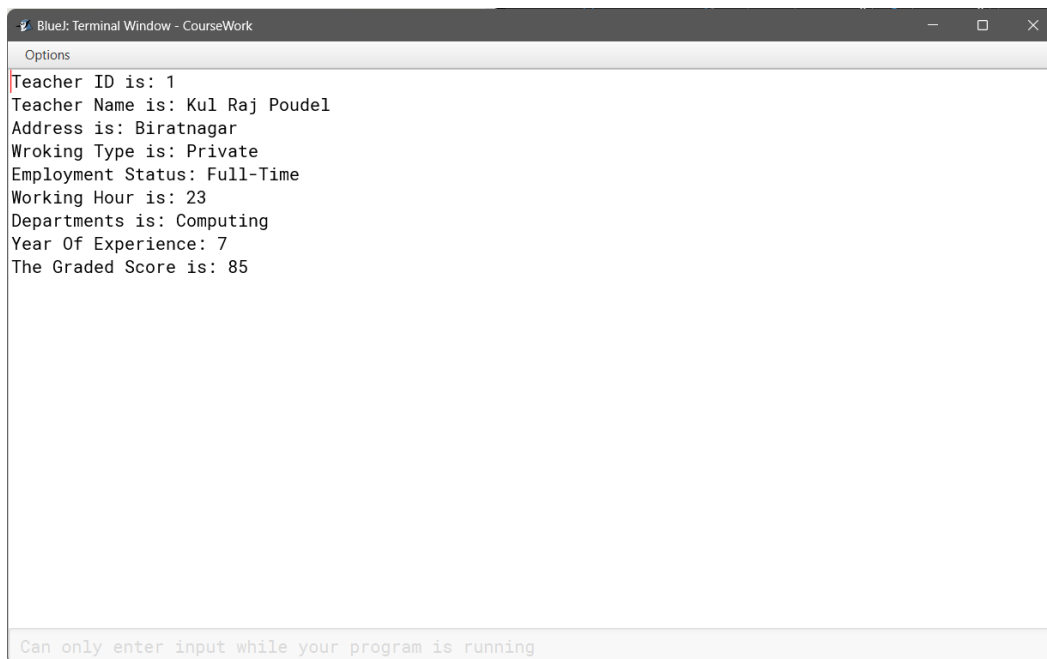


Figure 21: Screenshot of displayed Detail of Lecturer Class.

BlueJ: Create Object

Tutor(int teacherId, String teacherName, String address, String workingType, String empStatus, int workingHour, double salary, String speci, String acaQual, int performIndex)

Name of Instance:

new Tutor(,

)

OK Cancel

Figure 23: Screenshot of assigning the data of Tutor Class

BlueJ: Method Call

void setSalary(double newSalary, int newPerformIndex)

tutor1.setSalary(,
)

OK Cancel

Figure 22: Screenshot of assigning the data of setSalary Method

BlueJ: Terminal Window - CourseWork

Options

```
Teacher ID is: 1
Teacher Name is: Kul Raj Poudel
Address is: Biratnagar
Working Type is: Private
Employment Status: Full-Time
Working Hour is: 26
The Salary of Tutor is: 88000.0
The Specilization is: Java
The Academic Qualification is: Master
The Performance Index is:8
```

Can only enter input while your program is running

Figure 24: Screenshot of displayed the detail of Tutor class.

ERROR DETECTION AND CORRECTION

Syntax Error:

Syntax error is an error that occurs when a compiler or interpreter cannot understand the source code statement in order to generate machine code. In other words syntax error occur when there is syntactical error occur in java program due to incorrect use of Java syntax.

In coursework ,there was an syntax error in teacher class which was caused dur to use semicolon after definig the method Display()

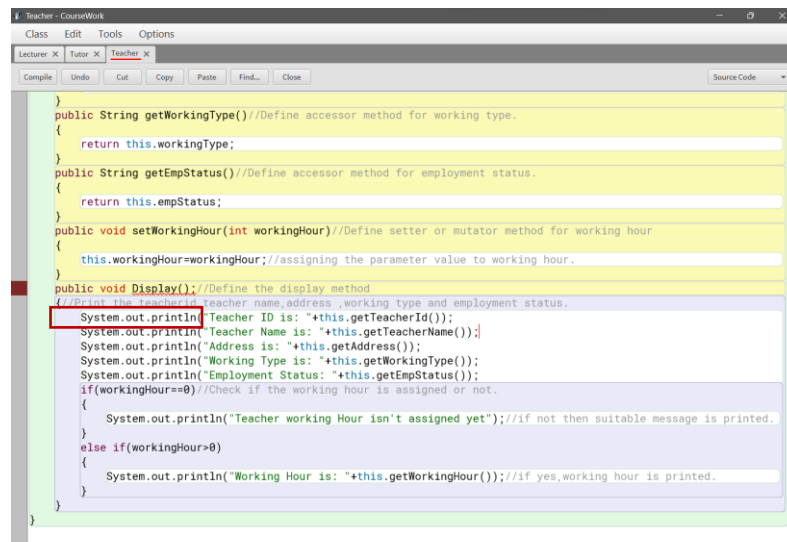


Figure 25:Screenshot of Syntax Error

Correction Of Syntax Error:

The error was corrected simply by removing the semicolon after the `Display()` method and error was solved.

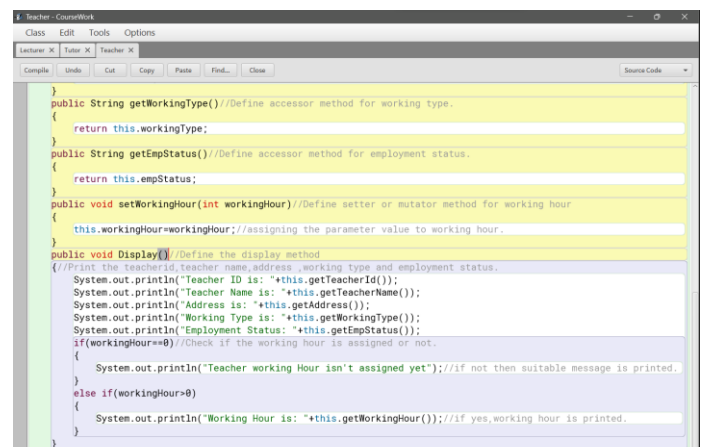
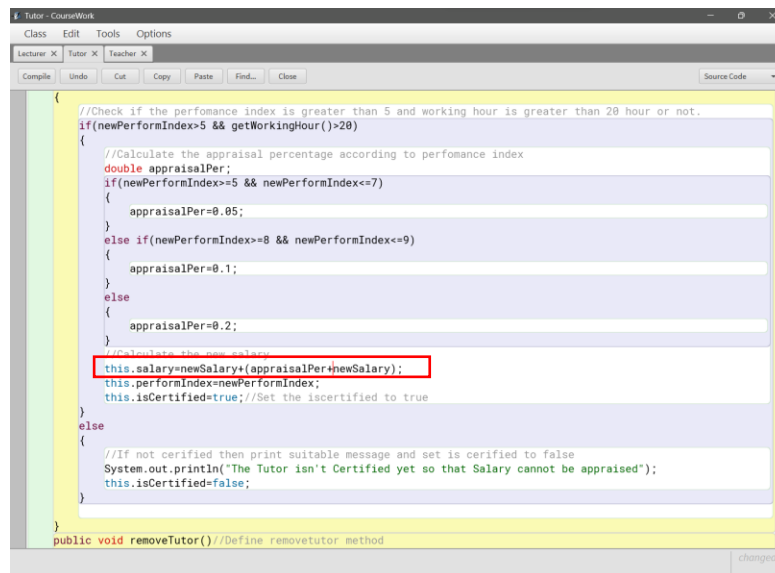


Figure 26:Correction of syntax Error

Semantics Error:

A semantic error is a type of error that occurs when the meaning or logic of a program is incorrect. A semantic error does not prevent the program from running, but it causes the program to produce an unexpected or wrong output.

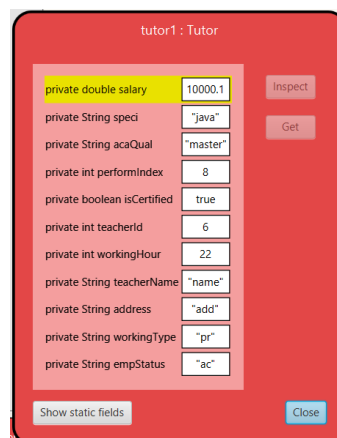
In coursework, there was a semantic error found in the Tutor class while calculating the appraisal of salary. The error caused the incorrect output; the output should come by appraising 10% of salary but instead it came double. This was due to the use of an addition sign instead of multiplication while calculating salary appraisal.



```
//Check if the performance index is greater than 5 and working hour is greater than 20 hour or not.
if(newPerformIndex>5 && getWorkingHour(>)>20)
{
    //Calculate the appraisal percentage according to performance index
    double appraisalPer;
    if(newPerformIndex==5 && newPerformIndex<=7)
    {
        appraisalPer=0.05;
    }
    else if(newPerformIndex==8 && newPerformIndex<=9)
    {
        appraisalPer=0.1;
    }
    else
    {
        appraisalPer=0.2;
    }
    //Calculate the new salary
    this.salary=newSalary+(appraisalPer+newSalary);
    this.performIndex=newPerformIndex;
    this.isCertified=true; //Set the isCertified to true
}
else
{
    //If not certified then print suitable message and set is certified to false
    System.out.println("The Tutor isn't Certified yet so that Salary cannot be appraised");
    this.isCertified=false;
}
}

public void removeTutor()//Define removeTutor method
```

Figure 27: Screenshot of Semantic error



Field	Value
private double salary	10000.1
private String speci	"java"
private String acaQual	"master"
private int performIndex	8
private boolean isCertified	true
private int teacherId	6
private int workingHour	22
private String teacherName	"name"
private String address	"add"
private String workingType	"pr"
private String empStatus	"ac"

Figure 28: Screenshot of Output of Semantic error.

Correction Of Semantics Error:

In coursework ,the semantic error was solved by replacing the addition sign of calculation with multiplication arithmetic sign ,which leads to correct output.

```
11 (newPerformIndex>5 && getWorkingHour())>20)
{
    //Calculate the appraisal percentage according to performance index
    double appraisalPer;
    if(newPerformIndex>=5 && newPerformIndex<=7)
    {
        appraisalPer=0.05;
    }
    else if(newPerformIndex>=8 && newPerformIndex<=9)
    {
        appraisalPer=0.1;
    }
    else
    {
        appraisalPer=0.2;
    }
    //Calculate the new salary
    this.salary=newSalary+(appraisalPer*newSalary);
    this.performIndex=newPerformIndex;
    this.isCertified=true;//Set the iscertified to true
}
else
{
    //If not certified then print suitable message and set is certified to false
    System.out.println("The Tutor isn't Certified yet so that Salary cannot be appraised");
    this.isCertified=false;
}
```

Figure 29: Screenshot of correction of semantic error

Field	Value
private double salary	5500.0
private String speci	"java"
private String acaQual	"master"
private int performIndex	8
private boolean isCertified	true
private int teacherId	1
private int workingHour	22
private String teacherName	"name"
private String address	"add"
private String workingType	"pr"
private String empStatus	"ac"

Figure 30: Screenshot of Output of Corrected Semantic error

Logical Error:

A logical error in java is a mistake in the logic or meaning of the code that causes the program to behave incorrectly or produce an unexpected output. Logical error are not detected by the java compiler, but they can be found by testing and debugging the code.

In coursework ,there was logical error found in lecturer class while assigning the grade. The error always shows the same result i.e Grade A even the score is less than 40 or 50.

```
public String gradeAssignment(int gradedScore,String department,int yearOfExp)
{
    //Check if the lecturer has enough experience and same department
    if(yearOfExp>=5 && this.getDepartment()==department)
    {
        this.hasGraded=true; //Set the hasGraded to true
        this.gradedScore=gradedScore;
        this.yearOfExp=yearOfExp;
        //Grade the assignments according to criteria given
        String grade;
        if(gradedScore<=70)
        {
            grade="Grade A";
            return grade;
        }
        else if(gradedScore>=60)
        {
            grade="Grade B";
            return grade;
        }
        else if(gradedScore>=50)
        {
            grade="Grade C";
            return grade;
        }
        else if(gradedScore>=40)
        {
            grade="Grade D";
            return grade;
        }
    }
}
```

Figure 31: Screenshot of Logical Error

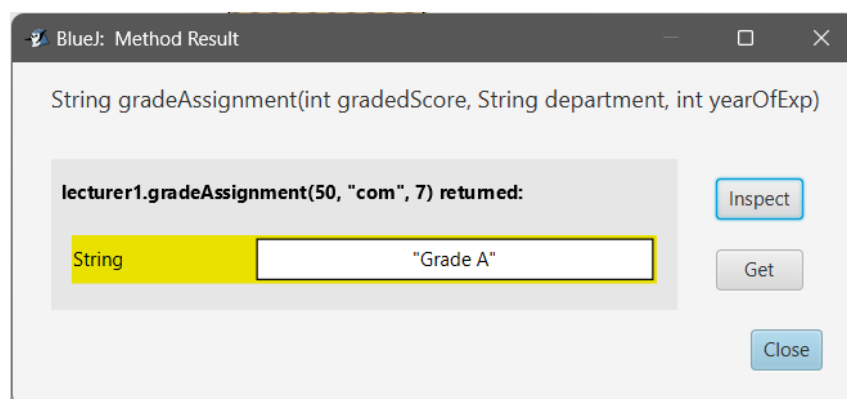


Figure 32: Screenshot of Output of Logical Error

Correction Of Logical Error:

In coursework ,the logical error was solved by replacing the less than sign into greater than in first statement of gadeAssignment() method and result in correct output.

```
public String gradeAssignment(int gradedScore, String department, int yearOfExp)
{
    //Check if the lecturer has enough experience and same department
    if(yearOfExp>=5 && this.getDepartment()==department)
    {
        this.hasGraded=true; //Set the hasGraded to true
        this.gradedScore=gradedScore;
        this.yearOfExp=yearOfExp;
        //Grade the assignments according to criteria given
        String grade;
        if(gradedScore>=70)
        {
            grade="Grade A";
            return grade;
        }
        else if(gradedScore>=60)
        {
            grade="Grade B";
            return grade;
        }
        else if(gradedScore>=50)
        {
            grade="Grade C";
            return grade;
        }
        else if(gradedScore>=40)
        {
            grade="Grade D";
            return grade;
        }
    }
}
```

Figure 33: Screenshot of Corrected Logical Error

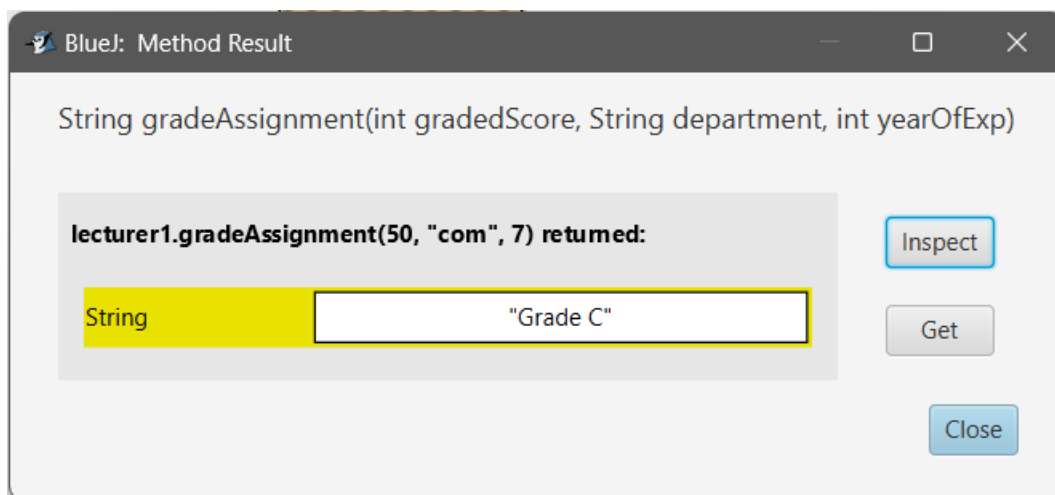


Figure 34: Screenshot of Corrected Logical Error Output.

CONCLUSION

In this coursework, I have acquired a solid understanding of the fundamental concepts and principles of object-oriented programming (OOP) in Java. These include objects, classes, inheritance. Additionally, I have become proficient in utilizing BlueJ, a Java development environment that facilitates OOP, to construct and interact with objects and classes.

To apply the OOP concept effectively, I have implemented a real-world problem scenario involving teachers, lecturers, and tutors. This involved creating a class to represent a teacher, along with two subclasses to represent a lecturer and a tutor respectively. Furthermore, I have compiled a comprehensive report that encompasses various aspects of my program, such as the problem statement, class design, implementation details, testing results, and conclusion.

Throughout the coursework, I encountered several difficulties and challenges. These included comprehending the syntax and semantics of Java, ensuring error-free code, designing and implementing classes and methods that meet the problem scenario's requirements and specifications, and thoroughly testing and debugging my program to ensure its functionality and accuracy. Additionally, I faced challenges in writing and formatting my report, utilizing MS Word and draw.io to incorporate tables, charts, diagrams, and equations.

To overcome these obstacles, I employed various strategies. Firstly, I extensively studied Java tutorials, documentation, and online resources, utilizing examples to enhance my understanding and practice Java programming. Secondly, I adopted meaningful and consistent variable names and comments to enhance the readability and

comprehensibility of my code. Thirdly, I leveraged the power of inheritance, to minimize code duplication and complexity, while simultaneously improving reusability and modularity. Lastly, I made effective use of BlueJ's features and tools, such as the editor, compiler, debugger, and object inspector, to create, edit, compile, execute, and manipulate objects and classes. Additionally, I utilized MS Word's diverse features and tools, spanning across file management, formatting, and styling, to create, edit, and format my report.

Bibliography

Aslanyan, V., n.d. *Learn Java and Object-Oriented Programme*. 2024 ed. s.l.:FreeCodeCamp Press.

Danny C. C. Poo, D. K. S. A., 2008. *Object-Oriented Programming and Java*. s.l.:Springer.

Hege, S. a. J. E. R., 1998. *W3school*. [Online]
Available at: <https://www.w3schools.com/java/default.asp>
[Accessed 2024].

APPENDIX

Code Of Teacher.java:

```
public class Teacher//Defined a class Teacher.

{

private int teacherId,workingHour;//Declare instance Variables or attributes.

private String teacherName,address,workingType,empStatus;

public Teacher(int teacherId,String teacherName,String address,

String workingType,String empStatus)//define a constructor method

{

this.teacherId=teacherId;//Initilizing the instance variable.

this.teacherName=teacherName;

this.address=address;

this.workingType=workingType;

this.empStatus=empStatus;

this.workingHour=0;//Assigned the parameter of working hour as 0 as it is not assigned yet.

}

public int getTeacherId();//Define accessor method for teacher ID.

{

return this.teacherId;
```

```
}
```

```
public int getWorkingHour()//Define accessor method for working Hour.
```

```
{
```

```
return this.workingHour;
```

```
}
```

```
public String getTeacherName()//Define accessor method for teacher name.
```

```
{
```

```
return this.teacherName;
```

```
}
```

```
public String getAddress()//Define accessor method for address.
```

```
{
```

```
return address;
```

```
}
```

```
public String getWorkingType()//Define accessor method for working type.
```

```
{
```

```
return this.workingType;
```

```
}
```

```
public String getEmpStatus()//Define accessor method for employment status.
```

```

{

return this.empStatus;

}

public void setWorkingHour(int workingHour)//Define setter or mutator method for
working hour

{

this.workingHour=workingHour;//assigning the parameter value to working hour.

}

public void Display()//Define the display method

{//Print the teacherid,teacher name,address ,working type and employment status.

System.out.println("Teacher ID is: "+this.getTeacherId());

System.out.println("Teacher Name is: "+this.getTeacherName());

System.out.println("Address is: "+this.getAddress());

System.out.println("Working Type is: "+this.getWorkingType());

System.out.println("Employment Status: "+this.getEmpStatus());

if(workingHour==0)//Check if the working hour is assigned or not.

{

System.out.println("Teacher working Hour isn't assigned yet");//if not then suitable
message is printed.

```

```
}
```

```
else if(workingHour>0)
```

```
{
```

```
System.out.println("Working Hour is: "+this.getWorkingHour());//if yes,working hour is  
printed.
```

```
}
```

```
}
```

```
}
```

Code of Lecturer.java:

```
public class Lecturer extends Teacher//Define lecturer class as subclass of Teacher  
class.
```

```
{
```

```
private String department;//Declare instance variable or attribute.
```

```
private int yearOfExp,gradedScore;
```

```
private boolean hasGraded;
```

```
//Define a constructor for Lecturer subclass.
```

```
public Lecturer(int teacherId,String teacherName,String address,String workingType
```

```
,String empStatus,String department,int yearOfExp, int workingHour)
```

```
{
```

super(teacherId,teacherName,address,workingType,empStatus);//Call the parent class constructor (teacher) with five parameters.

//Assigning the parameter values for department and year of experience.

this.department=department;

this.yearOfExp=yearOfExp;

this.gradedScore=0;//Assign graded score to 0.

this.hasGraded=false;//Assign hasgraded to false.

this.setWorkingHour(workingHour);//Calling setter method of parent class

}

public String getDepartment();//Define accessor method for department

{

return this.department;

}

public int getYearOfExp();//Define accessor method for year of experience.

{

return this.yearOfExp;

}

public int getGradedScore();//Define accessor method for graded score.

```
{  
  
    return this.gradedScore;  
  
}
```

public boolean getHasGraded()//Define accessor method for hasgraded.

```
{  
  
    return this.hasGraded;  
  
}
```

public void setGradedScore(int gradedScore)//Define getter or mutator method for graded Score.

```
{  
  
    this.gradedScore=gradedScore;  
  
    this.hasGraded=true;//Set the hasgraded to true.  
  
}
```

//Define method for grade assignment.

public String gradeAssignment(int gradedScore,String department,int yearOfExp)

```
{  
  
    //Check if the lecturer has enough experience and same department  
  
    if(yearOfExp>=5 && this.getDepartment()==department)
```

```

{

    this.hasGraded=true;//Set the hasGraded to true

    this.gradedScore=gradedScore;

    this.yearOfExp=yearOfExp;

    //Grade the assignments according to criteria given

    String grade;

    if(gradedScore>=70)

    {

        grade="Grade A";

        return grade;

    }

    else if(gradedScore>=60)

    {

        grade="Grade B";

        return grade;

    }

    else if(gradedScore>=50)

    {

        grade="Grade C";

```

```

        return grade;

    }

    else if(gradedScore>=40)

    {

        grade="Grade D";

        return grade;

    }

    else

    {

        grade="Grade E";

        return grade;

    }

    }

else

{

    this.hasGraded=false;

    //Print the suitable message if condition doesn't match

    String message="The lecturer is not qualified to grade a assignments";

```



```

        return message;

    }

}

public void Display();//Define display method with the help of method overriding

{

    super.Display();//Invoke display from parent class

    //Print the department and year of experience

    System.out.println("Departments is: "+this.getDepartment());

    System.out.println("Year Of Experience: "+this.getYearOfExp());

    //Check if the score is graded or not

    if(hasGraded)

    {

        System.out.println("The Graded Score is: "+this.getGradedScore());

    }

    else

    {

        //If not,print the suitable message

        System.out.println("Score has not been Graded Yet");

    }

}

```

}

}

Code Of Tutor.Java

```
public class Tutor extends Teacher//Define tutor subclass of Parent class Teacher

{

    //Declare attributes

    private double salary;

    private String speci,acaQual;

    private int performIndex;

    private boolean isCertified;

    //Define constructor of tutor class

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

        String empStatus,int workingHour,double salary,String speci,String acaQual,int

        performIndex)

    {

        //Call the teacher class constructor and setter method

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

        this.setWorkingHour(workingHour);

        //Assign the salary ,specilization ,academic qualification,performance index with

        parameter values

        this.salary=salary;

        this.speci=speci;
```

```

        this.acaQual=acaQual;

        this.performIndex=performIndex;

        this.isCertified=false;//Set iscertified to false
    }

    //Define accessor methods for each attributes

    public double getSalary()

    {

        return this.salary;

    }

    public String getSpeci()

    {

        return this.speci;

    }

    public String getAcaQual()

    {

        return this.acaQual;

    }

    public int getPerformIndex()

    {

```

```

        return this.performIndex;
    }

    public boolean getIsCertified()

    {

        return this.isCertified;
    }

    //Define setter method for new salary

    public void setSalary(double newSalary,int newPerformIndex)

    {

        //Check if the performance index is greater than 5 and working hour is greater than
        20 hour or not.

        if(newPerformIndex>5 && getWorkingHour(>20)

        {

            //Calculate the appraisal percentage according to performance index

            double appraisalPer;

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

            {

                appraisalPer=0.05;

            }

```

```

else if(newPerformIndex>=8 && newPerformIndex<=9)

{
    appraisalPer=0.1;

}

else

{

    appraisalPer=0.2;

}

//Calculate the new salary

this.salary=newSalary+(appraisalPer*newSalary);

this.performIndex=newPerformIndex;

this.isCertified=true;//Set the iscertified to true

}

else

{

    //If not cerified then print suitable message and set is cerified to false

    System.out.println("The Tutor isn't Certified yet so that Salary cannot be
appraised");

    this.isCertified=false;

```

```

    }

}

public void removeTutor()//Define removetutor method

{

    if(!isCertified)//Check if the tutor is certified yet or not

    {

        /*If not certified then set salary,performance to zero and specification,
        academic qualification to null and is certified to false*/

        this.salary=0;

        this.speci="";

        this.acaQual="";

        this.performIndex=0;

        this.isCertified=false;

    }

    else

    {

        //Print suitable message

        System.out.println("The Tutor is Certified and Salary is Appraised");

    }

```

```

}

//Define display method

public void Display();//method overriding

{

    //Check if the tutor is certified then print all the detail from parent class and subclass

    if(isCertified){

        super.Display();//Call the display method from teacher class

        //Print salary,specification,academic qualification,performance index

        System.out.println("The Salary of Tutor is: "+this.getSalary());

        System.out.println("The Specilization is: "+this.getSpeci());

        System.out.println("The Academic Qualification is: "+this.getAcaQual());

        System.out.println("The Performance Index is:"+this.getPerformIndex());

    }

    //If not certified then call the display method of parent class only

    else {

        super.Display();

    }

}

}

```