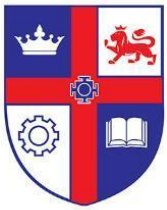

islington college
(इस्लिङ्टन कलेज)

CS4001NI Programming

30% Individual Coursework

2023-24 Autumn

Student Name: Bibek Kumar Sahani

London Met ID: 23047566

College ID: NP01NT4A230229

Group: N8

Assignment Due Date: Friday, January 26, 2024

Assignment Submission Date: Friday, January 26, 2024

I confirm that I understand my coursework needs to be submitted online via My Second Teacher 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 mark of zero will be awarded.

TABLE OF CONTENT

TABLE OF FIGURES

TABLE OF TABLES

INTRODUCTION

The overall goal of this course is to create a program that stores Teacher data, and all attributes. This project is designed to know or view the information of the teacher. It is based on the creation of a reference tool called Blue J, created by Michael Kolling in the late 1990's because it helps to see the structure and paths of classes and also provides the understanding of classes for visual motion modification. It has inheritance building properties where the lecturer and tutor categories extract results from the super category teacher, and this is the target relationship from which the results should be extracted from the supergroup to the child group. The aim of this assignment is to implement a real-world problem scenario using the Object-oriented concept of Java that includes creating a class to represent a teacher, together with its two subclasses to represent a Lecturer and a Tutor respectively.

ABOUT THE COURSE WORK

The Teacher class is like a base model with details such as teacherId, teacherName, address, workingType, employmentStatus, and workingHours. It can do basic things like showing and changing hours. Then, there's the Lecturer class, a kind of teacher with extras like department, yearsOfExperience, gradedScore, and whether they've graded assignments. They can give grades based on experience and department. Finally, the Tutor class adds more details like salary, specialization, academicQualifications, performanceIndex, and a certification status. Tutors can get more money based on performance, but if they have not performed well then they will not certified, and we can remove them. Also, we can see all these details, whether they have been certified or not.

TOOLS USED

a) Blue J: It is a user-friendly IDE (Integrated Development Environment) designed for learning programming in the context of JAVA. It is suitable of beginners in programming.

b) Ms-Word: It is inbuilt user-friendly software tool of MS-Office for PC that is used for word processing applications, offering rich text editing, formatting tools and some extra features.

c) Draw.io: It is a online diagramming tool having user-friendly interface which is used for creating flowcharts, different diagrams and visualization. It supports collaborative work and exports to various formats.

CLASS DIAGRAM

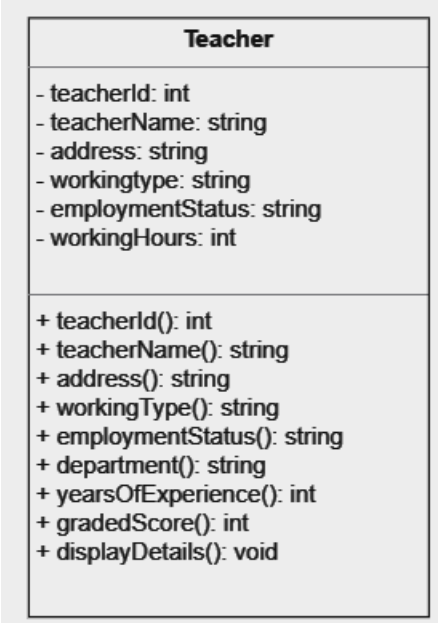

Fig 6.1 :-Teacher Class Diagram

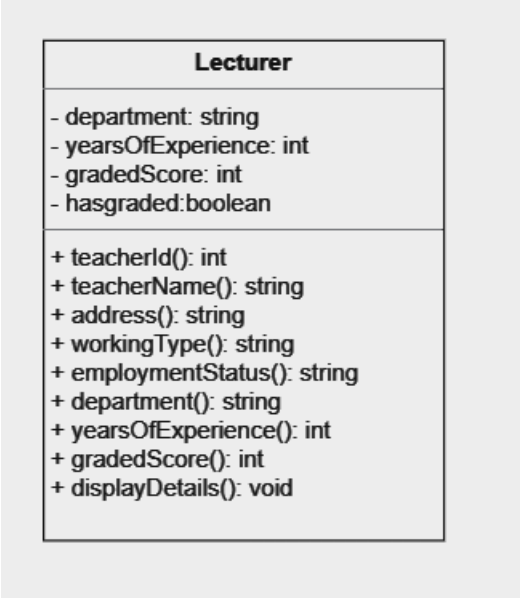

Fig 6.2 :- Lecturer Class Diagram

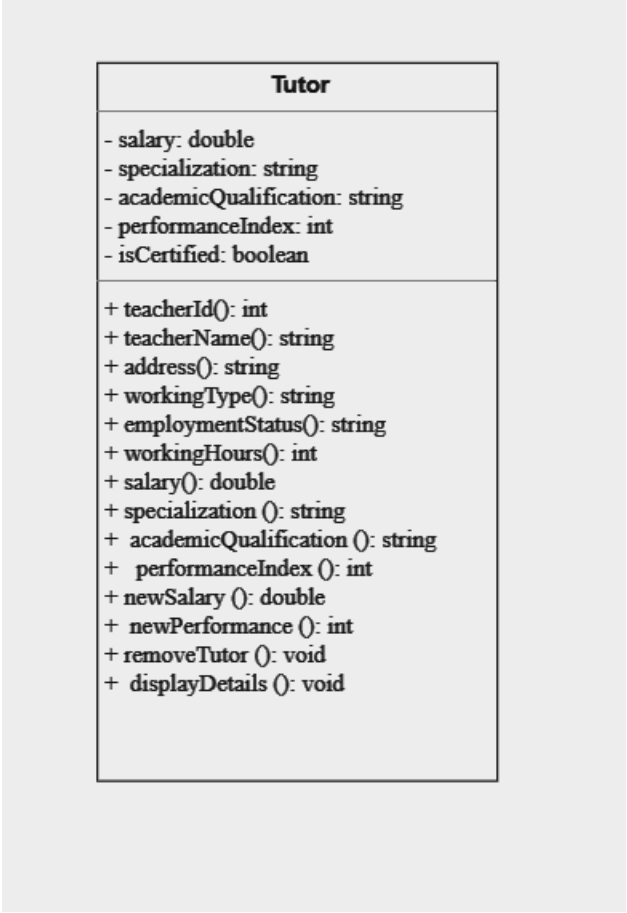

Fig 6.3 :- Tutor Class Diagram

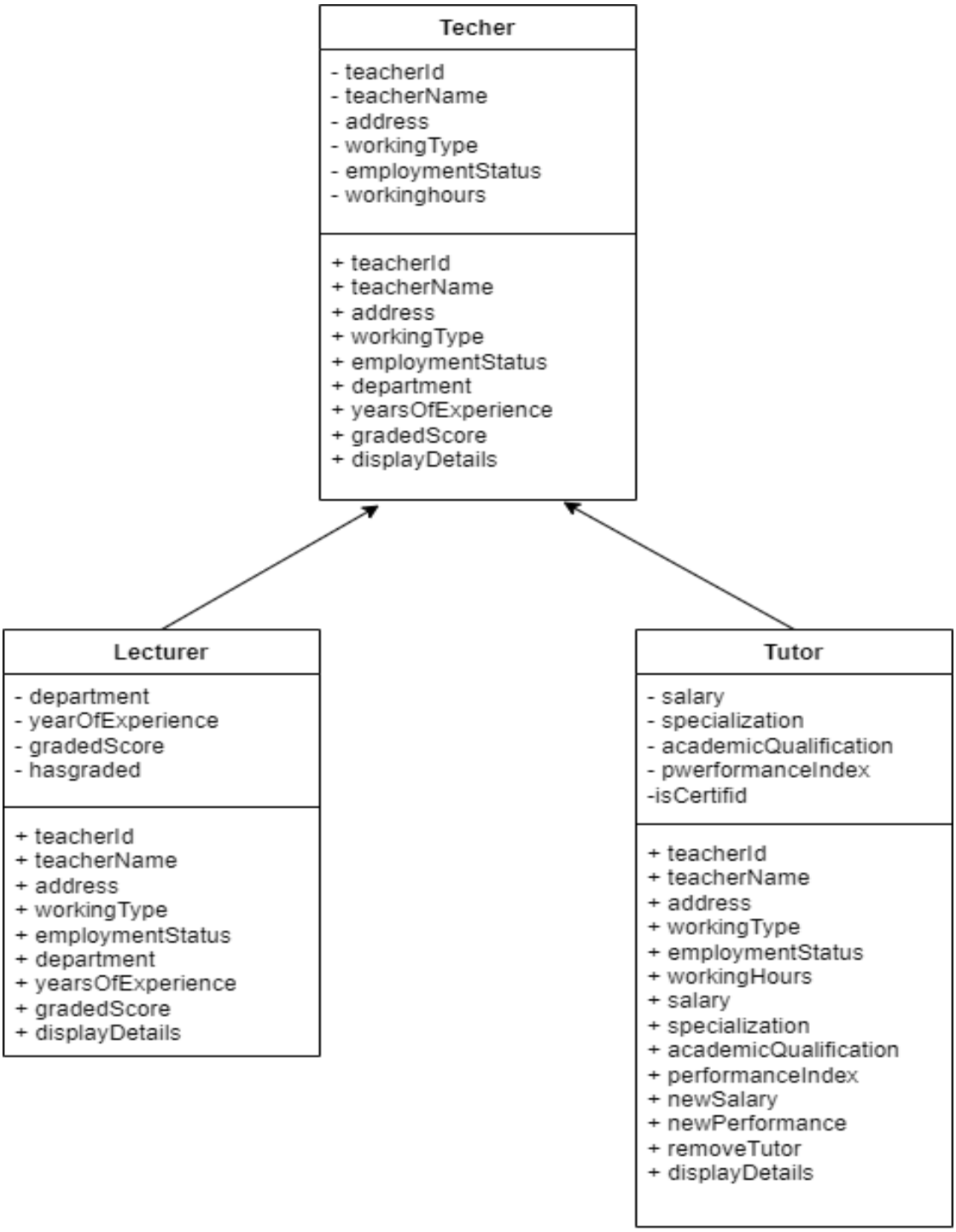

Fig 6.4 :- Relationship Class Diagram

7. PSEUDOCODE

7.1 Pseudocode for class Teacher

CREATE a class Teacher.

DO

DECLARE private teacherId as integer

DECLARE private teacherName; as string

DECLARE private address as string

DECLARE private workingType as string

DECLARE private employmentStatus as string

DECLARE private workingHours as integer

END DO

CREATE a constructor Teacher and initialize the variables with the appropriate vaues

INITIALIZE teacherId to teacherId

INITIALIZE teacherName to teacherName

INITIALIZE address to address

INITIALIZE workingType to workingType

INITIALIZE employmentStatus to employmentStatus

DEFINE accessor method for required attributes

FUNCTION int getTeacherId()

RETURN teacherId

END FUNCTION

FUNCTION String getteacherName()

RETURN teacherName

END FUNCTION

FUNCTION String getAddress()

RETURN address

END FUNCTION

FUNCTION String getWorkingType()

RETURN workingType

END FUNCTION

FUNCTION String getEmploymentStatus()

RETURN employmentStatus

END FUNCTION

DEFINE mutator method for each attribute

FUNCTION set workingHours (int newWorkingHours)

this.workingHours = newWorkingHours

END FUNCTION

DEFINE a method to display the teacher information

FUNCTION displayTeacherInfo()

PRINT “Teacher ID”

PRINT “Teacher Name”

PRINT “Address”

PRINT “Working Type”

PRINT “Employment Status”

END FUNCTION

7.2 Pseudocode for class Lecturer

CREATE a subclass lecturer extends Teacher.

DO

DECLARE private department as string

DECLARE private yearsOfExperience as integer

DECLARE private gradedScore as integer

DECLARE private hasGraded as boolean

END DO

CREATE a constructor Lecturer and initialize the variables with the appropriate values

CALL the super class variables

SUPER (teacherId, teacherName, address, workingType, employmentStatus)

INITIALIZE department to department

INITIALIZE yearsOfExperience to yearsOfExperience

INITIALIZE gradedScore to gradedScore

INITIALIZE hasGraded to false

DEFINE accessor method for required attributes

FUNCTION string getDepartment()

RETURN department

END FUNCTION

FUNCTION int getYearsOfExperience ()

RETURN yearsOfExperience

END FUNCTION

FUNCTION int getGradedScore ()

RETURN gradedScore

END FUNCTION

FUNCTION boolean getHasGraded ()

RETURN hasGraded

END FUNCTION

DEFINE mutator method for each attribute

FUNCTION set GradedScore (int newgradedScore)

this.gradedScore = gradedScore

END FUNCTION

DEFINE a additional method for grading assignments

FUNCTION gradeAssignment as integer

IF (!hasGraded && yearsOfExperience >= 5 &&
department.equals(studentDepartment))

IF (gradedScore >= 70)

gradedScore = score

PRINT “Grade A”

ELSE

IF (gradedScore >= 60)

gradedScore = score

```
        PRINT “Grade B”
ELSE
IF (gradedScore >= 50)
        gradedScore = score
        PRINT “Grade C”
ELSE
IF (gradedScore >= 40)
        gradedScore = score
        PRINT “Grade D”
ELSE
        PRINT “Grade E”

END IF
END FUNCTION


DEFINE a method to display the LecturerInfo

FUNCTION displayLecturerInfo()

PRINT “Lecturer”

PRINT “Department”

PRINT “Years Of Experience”

ELSE
```

PRINT “Graded Score”

END FUNCTION

7.3 Pseudocode for tutor

CREATE a subclass Tutor extends Teacher.

DO

DECLARE private salary as double

DECLARE private specialization as string

DECLARE private academicQualification as string

DECLARE private performanceIndex as integer

DECLARE private isCertified as boolean

END DO

CREATE a constructor Tutor and initialize the variables with the appropriate values

CALL the super class variables

SUPER (teacherId, teacherName, address, workingType, employmentStatus)

INITIALIZE salary to salary

INITIALIZE specialization to specialization

INITIALIZE academicQualification to academicQualification

INITIALIZE performanceIndex to performanceIndex

INITIALIZE isCertified to false

DEFINE accessor method for required attributes

FUNCTION double getSalary()

RETURN salary

END FUNCTION

FUNCTION string getSpecialization()

RETURN specialization

END FUNCTION

FUNCTION string getAcademicQualification ()

RETURN academicQualification

END FUNCTION

FUNCTION int getPerformanceIndex ()

RETURN performanceIndex

END FUNCTION

FUNCTION boolean getIsCertified ()

RETURN isCertified

END FUNCTION

DEFINE a method for calculating salary

IF (newPerformanceIndex > 5 && getWorkingHours() >20)

double appraisal = 0.00

IF (newPerformanceIndex >= 5 && newPerformanceIndex <=7)

appraisal = 0.05

ELSE IF

(newPerformanceIndex >= 8 && newPerformanceIndex <=9)

appraisal = 0.1

```
        ELSE IF
            (newPerformanceIndex == 10)
                appraisal = 0.2

        salary = newSalary + (appraisal * newSalary)
        isCertified = true

            ELSE
                PRINT "Salary is not approved by tutor.”

    END IF

DEFINE a method to remove tutor

FUNCTION removeTutor()
IF (!isCertified)
        Salary = 0
Specialization = “”
academicQualification = “”
performanceIndex = 0
isCertified = false
END FUNCTION

DEFINE a method to display TutorInfo()
Function displayTutorInfo()
IF (isCertified)
PRINT “Tutor Details”
PRINT “Salary”
```

PRINT “Specialization”

PRINT “Academic Qualification”

PRINT “Performance Index”

END FUNCTION

DESCRIPTION OF METHODS

Parameterized Constructor:

It allows you to create object with customized attributes by accepting specific values during object creation.

Accessor Methods (Getters):

It returns the value of private class variables.

Setter Method:

It sets the value of private class variables.

Display Method:

Displays detailed information about the teacher.

Main Method:

It allows the user to input for creating a Teacher object, setting working hours, and displaying teacher details.

Usage of Scanner Class:

It takes users input for teacher details and working hours.

Closing Scanner:

It closes the Scanner to avoid resource leaks.

TESTING

22.1 Testing1

To inspect the Lecturer class, grade the assignment, and re-inspect the Lecturer Class

Test No:	1
Objective:	-To inspect the Lecturer class, grade the assignment, and re-inspect the Lecturer Class
Action:	The lecturer is called with the following arguments: Teacher Id = 1 Teacher Name = Ram Karki Address = Pepsi Working Type = part-time Employment status = Temporary Department = Information Technology Years of Experience = 15 Graded Score = 5 Inspection of the Lecturer class. Void appoint Lecturer Is called with the following arguments: Teacher ID: 1 Teacher Name: Ram Karki Address: Pepsi Working Type: Part-Time Employment Status: Temporary Working Hours: Not assigned Lecturer: Department: Information Technology Years of Experience: 15 Not Graded(NG)
Expected Result:	The lecturer grade assignment will be done
Actual Result:	The lecturer assignment was done
Conclusion:	The test is successful.

Table:-22.1.1 To inspect the Lecturer class, grade the assignment, and re-inspect the Lecturer Class

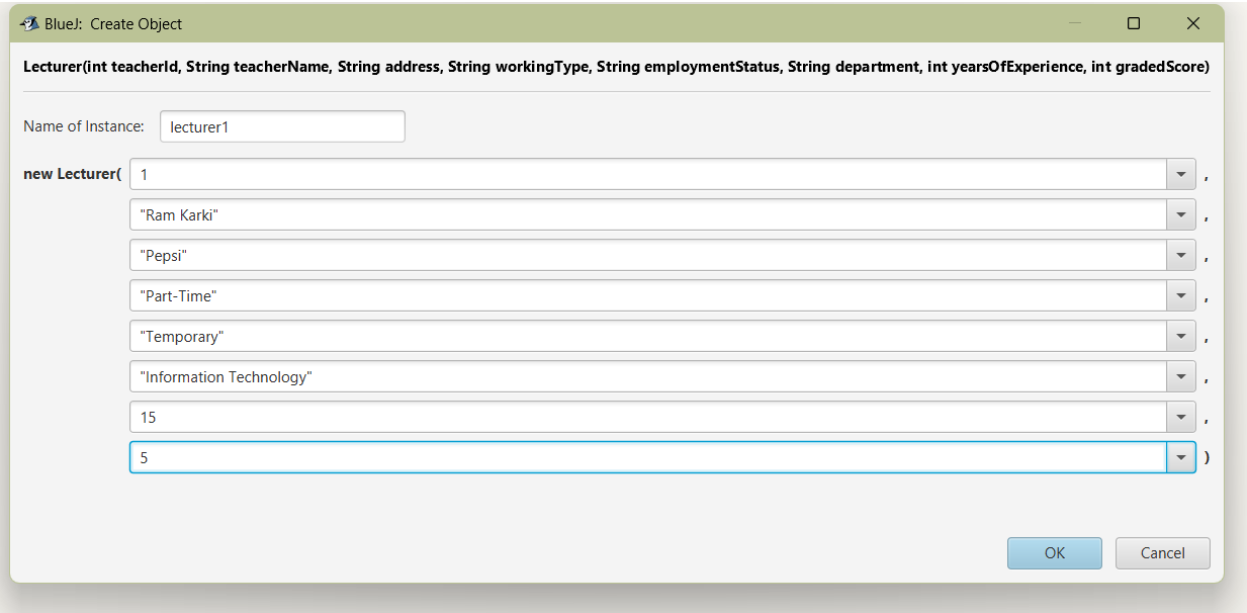

Fig:- 22.1.1.1 (creating object)

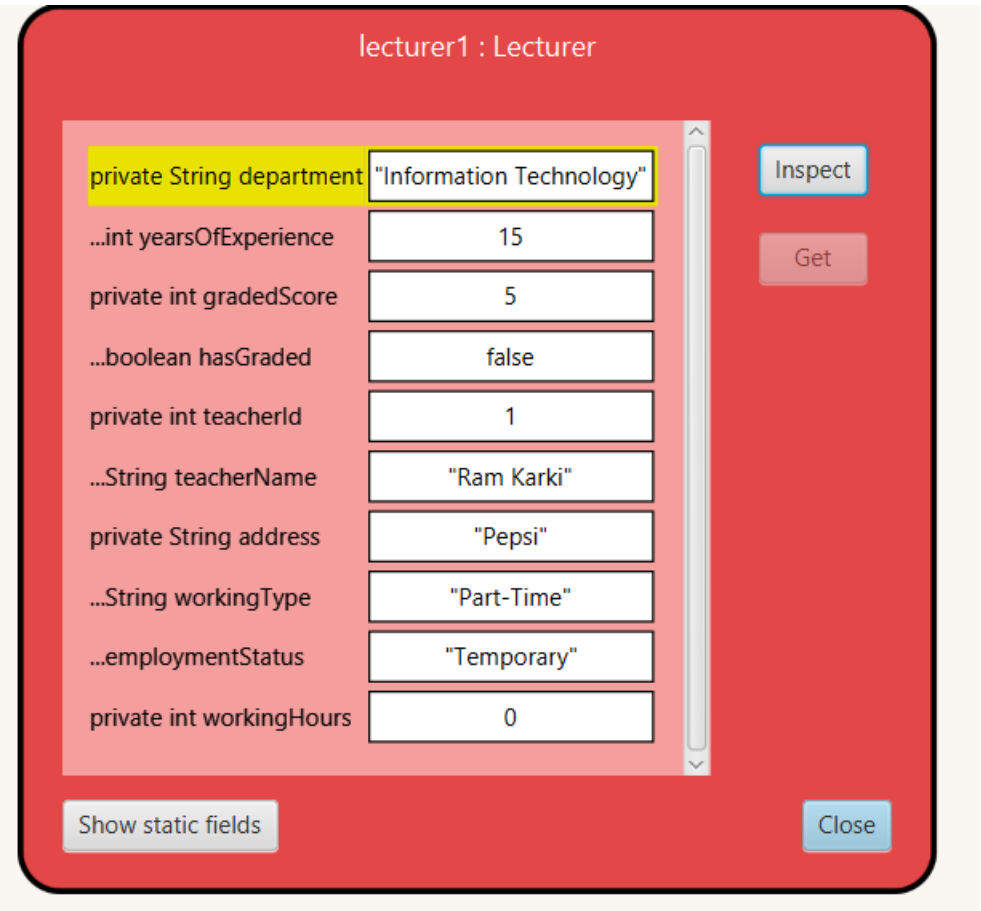

Fig:- 22.1.1.1.1(inspect)

22.2 Testing 2

To Inspect Tutor class, set salary and reinspect the Tutor class

Test No:	2
Objective:	-To Inspect Tutor class, set salary and reinspect the Tutor class
Action:	The tutor class is called with the following arguments: Teacher Id = 1 Teacher Name = Ram Karki Address = Pepsi Working Type = part-time Employment status = Temporary Working Hours = 5 Salary = 50000 Specialization = Lecturer Academic Qualification = MBA Performance Index = 5 Tutor class. Void appoint Tutor Is called with the following arguments: Teacher ID: 1 Teacher Name: Ram Karki Address: Pepsi Working Type: Part-Time Employment Status: Temporary Working Hours: Not assigned
Expected Result:	The Tutor salary will be set.
Actual Result:	The tutor salary was set.
Conclusion:	The test is successful.

Table:-22.2.2 To Inspect Tutor class, set salary and reinspect the Tutor class

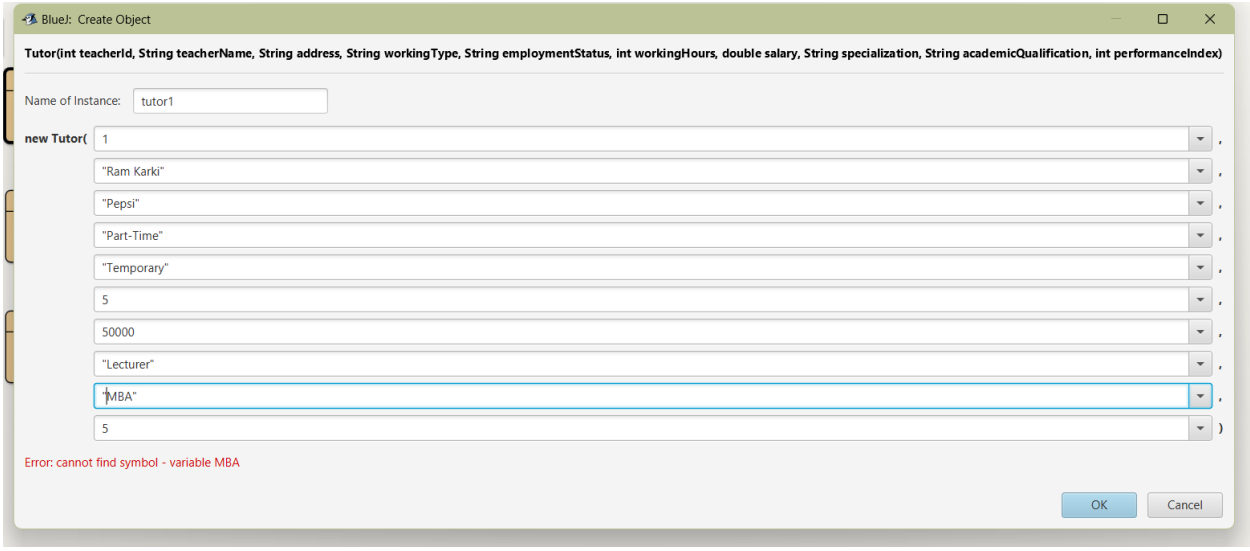

Fig:-22.2.2.2(creating object)

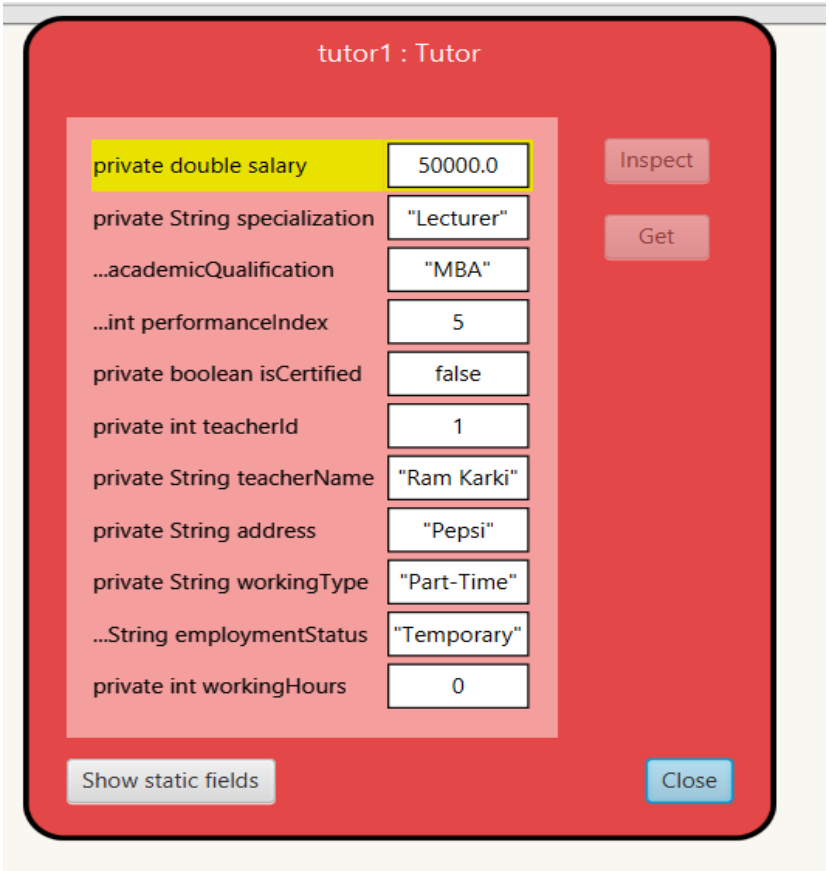

Fig:- 22.2.2.2.2(inspect)

22.3 Testing 3

To Inspect after removing Tutor

Test No:	2
Objective:	-To Inspect after removing Tutor
Action:	Void remove tutor () Reinspection of tutor class
Expected Result:	The Tutor will be removed
Actual Result:	The tutor was removed
Conclusion:	The test is successful.

Table:-22.3.3 To Inspect after removing Tutor

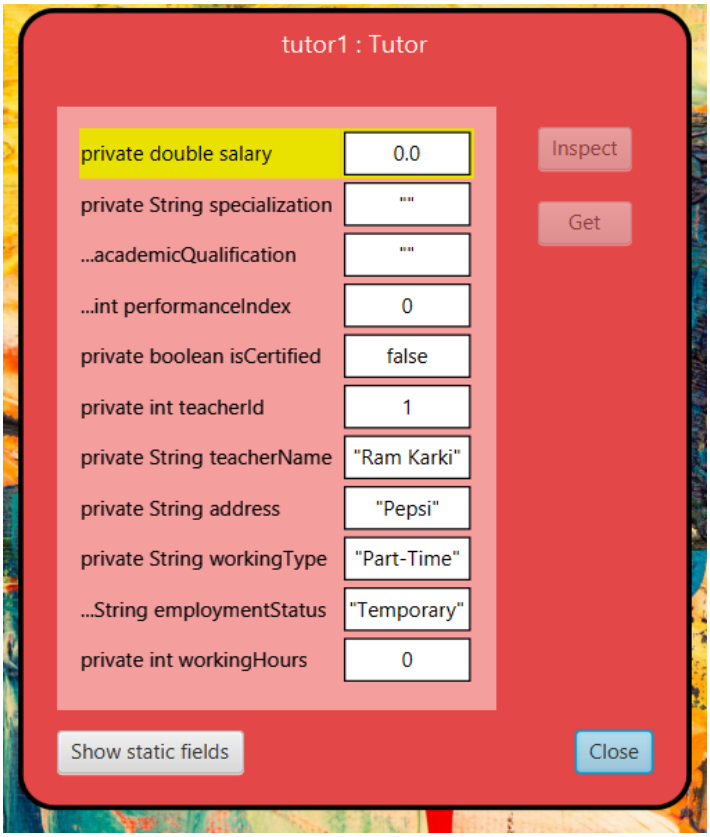

Fig:-22.3.3.3 (inspect)

22.4 Testing 4

To Display the details of Lecturer and Tutor

Test No:	2
Objective:	To Display the details of Lecturer and Tutor
Action:	Void display LecturerInfo() Void display TutorInfo()
Expected Result:	The lecturer and tutor needs to be displayed.
Actual Result:	The lecturer and tutor was displayed.
Conclusion:	The test is successful.

Table:-22.2.2 To Inspect Tutor class, set salary and reinspect the Tutor class

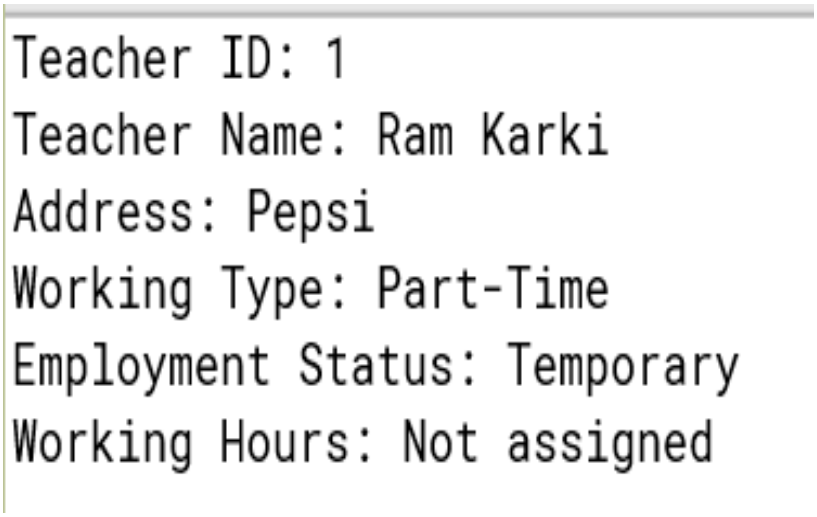

Fig:- 22.2.2.2

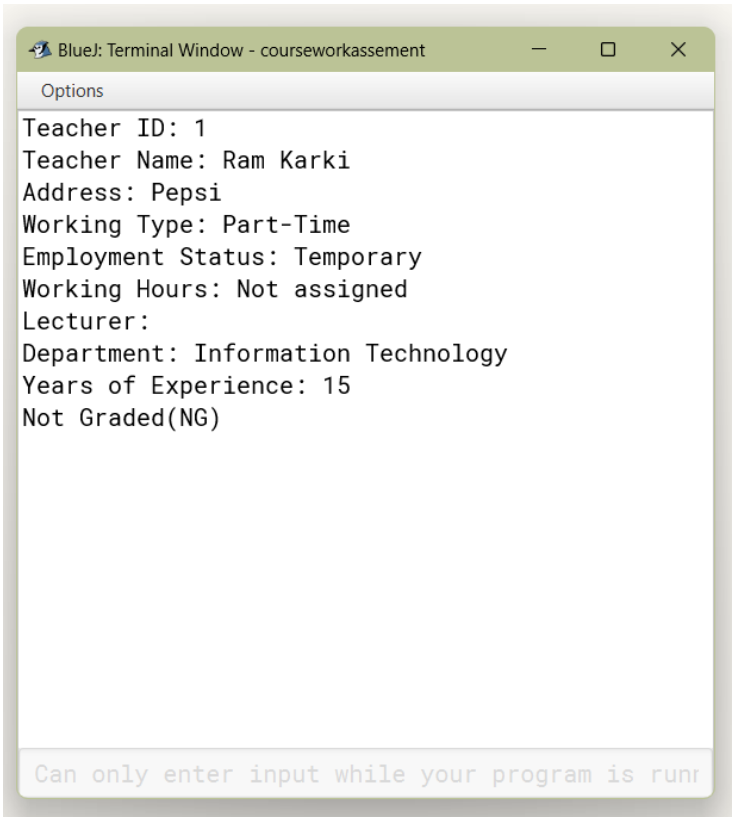

Fig:- 22.2.2.2.2

6. ERROR DETECTION AND CORRECTION

6.1 Syntax Error

```
        return workingHours;
    }

    //method to set working hours (setter method)
    public void setWorkingHours (int newWorkingHours) {
        this.workingHours = newWorkingHours
    }
```

Fig:- 6.1.1

It was corrected by adding semicolon.

6.2 Semantics Error

```
    //setter method
    public void setGradedScore(int gradedScore) {
        this.gradedScore = hasgraded;
    }
```

Fig:- 6.2.1

It was corrected by replacing hasgraded with gradedScore.

6.3 Logical Error

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

Fig:- 6.3.1

It was corrected by interchanging the words.

CONCLUSION

The course work was finally accomplished with lots of hard work and efforts. Different type of errors was occurred while doing the coursework. The coursework was quite difficult but not impossible. Firstly, the coursework was divided into three different parts; Teacher, Lecturer and Tutor. All the values were exported from class to class and documentation of the work was created. The concept about heritance was learned where parameters could be exported from parent class to the children class using a function name called extend. A method named constructor was also learned where the object was created and was also learned about the use of super and this keyword. The difference between get and set method was also learned where the values assigned by users are kept.

The obstacles and problems that were faced by me while during the coursework was extracting the values from parent class and children class and assigning of the get and set method with parameters was also difficult to keep. The problems were solved by asking and visiting to the teacher daily and also researching on various sites about programming and also learning about the ideas and views which were kept on lecture and tutorial slides. It gave me a good knowledge and idea about programming in order to complete the course work.

12. REFERENCES

(Jyothi, 2019)

Bibliography
Jyothi, V., 2019. *Tutorial point.* [Online]
Available at: https://www.tutorialspoint.com/Inheritance-in-Java
[Accessed 21 01 2024].

(Anon., n.d.)

Bibliography
Anon., n.d. *w3 schools.* [Online]
Available at: https://www.w3schools.com/java/java_inheritance.asp
[Accessed 21 01 2024].

(Anon., n.d.)

Bibliography
Anon., n.d. *Java point.* [Online]
Available at: https://www.javatpoint.com/java-constructor
[Accessed 21 01 2024].

(Simpli learn, n.d.)

Bibliography
Simpli learn, n.d. *Simpli learn.* [Online]
Available at: https://www.simplilearn.com/tutorials/java-tutorial/methods-in-java
[Accessed 21 01 2024].

APPENDIX

A. For Teacher

```
import java.util.*:
public class Teacher
{
    private int teacherId;
    private String teacherName;
    private String address;
    private String workingType;
    private String employmentStatus;
    private int workingHours;
    Teacher(int teacherId,String teacherName, String address, String workingType,
   String employmentStatus)
    {
    this.teacherId = teacherId;
    this.teacherName = teacherName;
    this.address = address;
    this.workingType = workingType;
    this.employmentStatus = employmentStatus;
    this.workingHours = 0:
    }

    public int getTeacherId() {
        return teacherId;
    }
```

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

public String getWorkingType() {
    return workingType;
}

public String getEmploymentStatus() {
    return employmentStatus;
}

public int getWorkingHours() {
    return workingHours;
}

public void setWorkingHours (int newWorkingHours) {
    this.workingHours = newWorkingHours;
}

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");
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter Teacher ID:");
        int teacherId = sc.nextInt();
        sc.nextLine();

        System.out.println("Enter Teacher Name:");
        String teacherName = sc.nextLine();

        System.out.println("Enter Address:");
        String address = sc.nextLine();

        System.out.println("Enter Working Type:");
        String workingType = sc.nextLine();

        System.out.println("Enter Employment Status:");
        String employmentStatus = sc.nextLine();

        Teacher teacher = new Teacher(teacherId, teacherName, address, workingType,
employmentStatus);
```

```
        teacher.displayTeacherInfo();
        System.out.println("Enter Working Hours:");
        int newWorkingHours = sc.nextInt();
        teacher.setWorkingHours(newWorkingHours);
            teacher.displayTeacherInfo();
        sc.close();
    }
}
```

B. For Lecturer

```
import java.util.*;

class Lecturer extends Teacher
{
    private String department;
    private int yearsOfExperience;   //camelcase
    private int gradedScore;
    private boolean hasGraded;

    Lecturer(int teacherId, String teacherName, String address, String workingType,
String employmentStatus, String department, int yearsOfExperience, int gradedScore)
    {
        super(teacherId, teacherName, address, workingType, employmentStatus);
        super.setWorkingHours(0);  //Initialize working hours to 0(not assigned)
        this.department = department;
        this.yearsOfExperience = yearsOfExperience;
        this.gradedScore = gradedScore;
        this.hasGraded = false;   //initially, assignments have not been graded yet
    }
```

```
    public String getDepartment() {
    return department;
}

 public int getYearsOfExperience() {
    return yearsOfExperience;
}

 public int getGradedScore() {
    return gradedScore;
}

 public boolean hasGraded() {
    return hasGraded;
}
    public void setGradedScore(int gradedScore) {
    this.gradedScore = gradedScore;
}

   public void gradeAssignment(int gradedScore, String studentDepartment, int
studentYearsOfExperience)
   {
      if (!hasGraded && yearsOfExperience >= 5 &&
department.equals(studentDepartment))
      {
         if (gradedScore >= 70) {
            this.gradedScore = gradedScore;
            System.out.println("Grade: A");
         }
```

```
        else if (gradedScore >= 60) {
            this.gradedScore = gradedScore;
            System.out.println("Grade: B");
        }
        else if (gradedScore >= 50) {
            this.gradedScore = gradedScore;
            System.out.println("Grade: C");
        }
        else if (gradedScore >= 40) {
            this.gradedScore = gradedScore;
            System.out.println("Grade: D");
        }
        else {
            System.out.println("Grade: E");
        }
        this.hasGraded = true;
    }
    else {
        System.out.println("Not Graded(NG)");
    }
}
    public void displayLecturerInfo()
{
    super.displayTeacherInfo

    System.out.println("Lecturer:");
System.out.println("Department: " + department);
    System.out.println("Years of Experience: " + yearsOfExperience);
```

```
    if (hasGraded) {
        System.out.println("Graded Score: " + gradedScore);
    }
    else {
        System.out.println("Not Graded(NG)");
    }
}
     public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter Teacher ID:");
    int teacherId = sc.nextInt();
    sc.nextLine();

    System.out.println("Enter Teacher Name:");
    String teacherName = sc.nextLine();

    System.out.println("Enter Address:");
    String address = sc.nextLine();

    System.out.println("Enter Working Type:");
    String workingType = sc.nextLine();

    System.out.println("Enter Employment Status:");
    String employmentStatus = sc.nextLine();

    System.out.println("Enter Department:");
    String department = sc.nextLine();
```

```
        System.out.println("Enter Years of Experience:");
        int yearsOfExperience = sc.nextInt();
 Lecturer lecturer = new Lecturer(teacherId, teacherName, address, workingType,
employmentStatus, department, yearsOfExperience, 0)
        lecturer.displayLecturerInfo();


        sc.close();
```

C. For Tutor

```
import java.util.*;
public class Tutor extends Teacher
{
    private double salary;
    private String specialization;
    private String academicQualification;   //camelcase
    private int performanceIndex;
    private boolean isCertified;

    public Tutor(int teacherId, String teacherName, String address, String workingType,
String employmentStatus, int workingHours, double salary, String specialization,
                String academicQualification, int performanceIndex) {
        super(teacherId, teacherName, address, workingType, employmentStatus);
        this.salary = salary;
        this.specialization = specialization;
        this.academicQualification = academicQualification;
        this.performanceIndex = performanceIndex;
        this.isCertified = false;  //is set to be false as mentioned in question
    }
        public double getSalary() {
        return salary;
    }

    public String getSpecialized() {
        return specialization;
    }

    public String getAcademicQualification () {
```

```
    return academicQualification;
}

public int getPerformanceIndex () {
    return performanceIndex;
}

public boolean isCertified() {
    return isCertified;
}

public void setSalaryAndCertification(double newSalary, int newPerformanceIndex) {
    if (newPerformanceIndex > 5 && getWorkingHours() >20) {
        double appraisal = 0.00;
        if (newPerformanceIndex >= 5 && newPerformanceIndex <=7) {
            appraisal = 0.05;
        }
        else if
            (newPerformanceIndex >= 8 && newPerformanceIndex <=9) {
                appraisal = 0.1;
            }
            else if
                (newPerformanceIndex == 10) {
                    appraisal = 0.2;
                }
        this.salary = newSalary + (appraisal * newSalary);
        this.isCertified = true;
            }
```

```
            else {
                System.out.println("Salary is not approved by tutor.");
            }
            }

public void removeTutor () {
    if (!isCertified) {
        this.salary = 0;
        this.specialization = "";
        this.academicQualification = "";
        this.performanceIndex = 0;
        this.isCertified = false;
    }
            }

public void displayTutorInfo () {
    if (isCertified) {
        System.out.println("Tutor Details:");
        System.out.println("Salary: " + salary);
        System.out.println("Specialization: " + specialization);
        System.out.println("Academic Qualification:" + academicQualification);
        System.out.println("Performance Index:" + performanceIndex);
        super.displayTeacherInfo();
    }
    else {
        super.displayTeacherInfo();
    }
    }
```

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    System.out.println("Enter Teacher ID:");
    int teacherId = sc.nextInt();
    sc.nextLine(); // use the newline character

    System.out.println("Enter Teacher Name:");
    String teacherName = sc.nextLine();

    System.out.println("Enter Address:");
    String address = sc.nextLine();

    System.out.println("Enter Working Type:");
    String workingType = sc.nextLine();

    System.out.println("Enter Employment Status:");
    String employmentStatus = sc.nextLine();

    System.out.println("Enter Working Hours:");
    int workingHours = sc.nextInt();

    System.out.println("Enter Salary:");
    double salary = sc.nextDouble();

    sc.nextLine();
```

```
        System.out.println("Enter Specialization:");
        String specialization = sc.nextLine();

        System.out.println("Enter Academic Qualifications:");
        String academicQualifications = sc.nextLine();

        System.out.println("Enter Performance Index:");
        int performanceIndex = sc.nextInt();

        Tutor tutor = new Tutor(teacherId, teacherName, address, workingType,
employmentStatus,
                workingHours, salary, specialization, academicQualifications,
performanceIndex);

        tutor.displayTutorInfo();

        System.out.println("Enter New Salary:");
        double newSalary = sc.nextDouble();

        System.out.println("Enter New Performance Index:");
        int newPerformanceIndex = sc.nextInt();

        tutor.setSalaryAndCertification(newSalary, newPerformanceIndex);

        tutor.displayTutorInfo();

        System.out.println("Do you want to remove the tutor? (Enter 'yes' or 'no')");
        sc.nextLine();
        String removeDecision = sc.nextLine().toLowerCase();
```

```
    if (removeDecision.equals("yes")) {
        tutor.removeTutor();
        System.out.println("Tutor has been removed.");
    } else {
        System.out.println("Tutor has not been removed.");
    }
    tutor.displayTutorInfo();
    sc.close();
}
}
```

```
    }
}
```