CS4001NI Programming

30% Individual Coursework

2023-24 Autumn

Student Name: Sujal Parajuli

London Met ID: 23050262

College ID: np01cp4a230257

Group: C10

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

Table of Figures

Table of Table

1.Introduction

Java is a programming language which is one of the most popular languages which was developed in 1995 AD by James Gosling at Sun Micro Systems which was later acquired by Oracle Corporation. Java is a class-based OOP (Object Oriented Programing Language) which aims to reduce implementation dependencies. Java is mainly used for developing web applications, desktop applications and mobile applications. This programming language grew its popularity due to its simplicity, user friendly interface, Platform Independence and security features (What is Java technology and why do I need it?, n.d.).

Java codes are compiled into byte codes that can be run on any java virtual machine. It comes with Java Development Kit (JDK), Java Virtual Machine (JVM), Java Run time Environment (JRM) which allows java to run in all operating systems by converting java code into byte code. Java is a statically typed language as it allows developers to check whether a program will work or not without running it.

2.Bluej

BlueJ is an IDE (Integrated Development Environment)

- It is an Editor
- It is a Debugger
- It is a Viewer

In bluej to display something we need to create a main function. Main function is the starting point of the program. Class name in bluej should be of one word with no spacing, it can have underscore when it has two words. Bluej is a free java development environment which allows students and other beginner developers to learn java programs quickly and easily. I used this development environment to complete my coursework (About BlueJ, n.d.).

3.Class Diagram

3.1. Teacher Class Diagram

Table 1 Table of Class Diagram of Teacher Class

Table 1 Teacher

Teacher
- teacherID: int - teacherName: String - addres: String - workingType: String - employmentStatus: String - WorkingHours: int
<<constructor>> Teacher (teacherID: Int, teacherName: String, address: String, workingType: String, employment Status: String workingHours: int) +getTeacherID(): int +getTeacherName(): String +getAddress(): String +getEmploymentStatus +getWorkingType() : String +getWorkingHours() : int + setWorkingHours(newWorkingHours: int) +display(): void

3.2 Lecturer Class Diagram

Table 2 Table of Class Diagram of Lecturer Class

Table 2 Lecturer

Lecturer
- department: String - YearsOfExperience: int - gradedScore: int - hasGraded: boolean
<<constructor>> Leacturer (teacherID: Int, teacherName: String, address: String, workingType: String, employmentStatus: String workingHours: int, String YearsOfExprience: int) +getDepartment(): String +getYearsOfExprience(): int +getgradedScore(): int +setgradedScore(gradedScore: int): void +gradeAssignment(gradedScore: int, Department: String, YearsOfExprience: int):void +display(): void

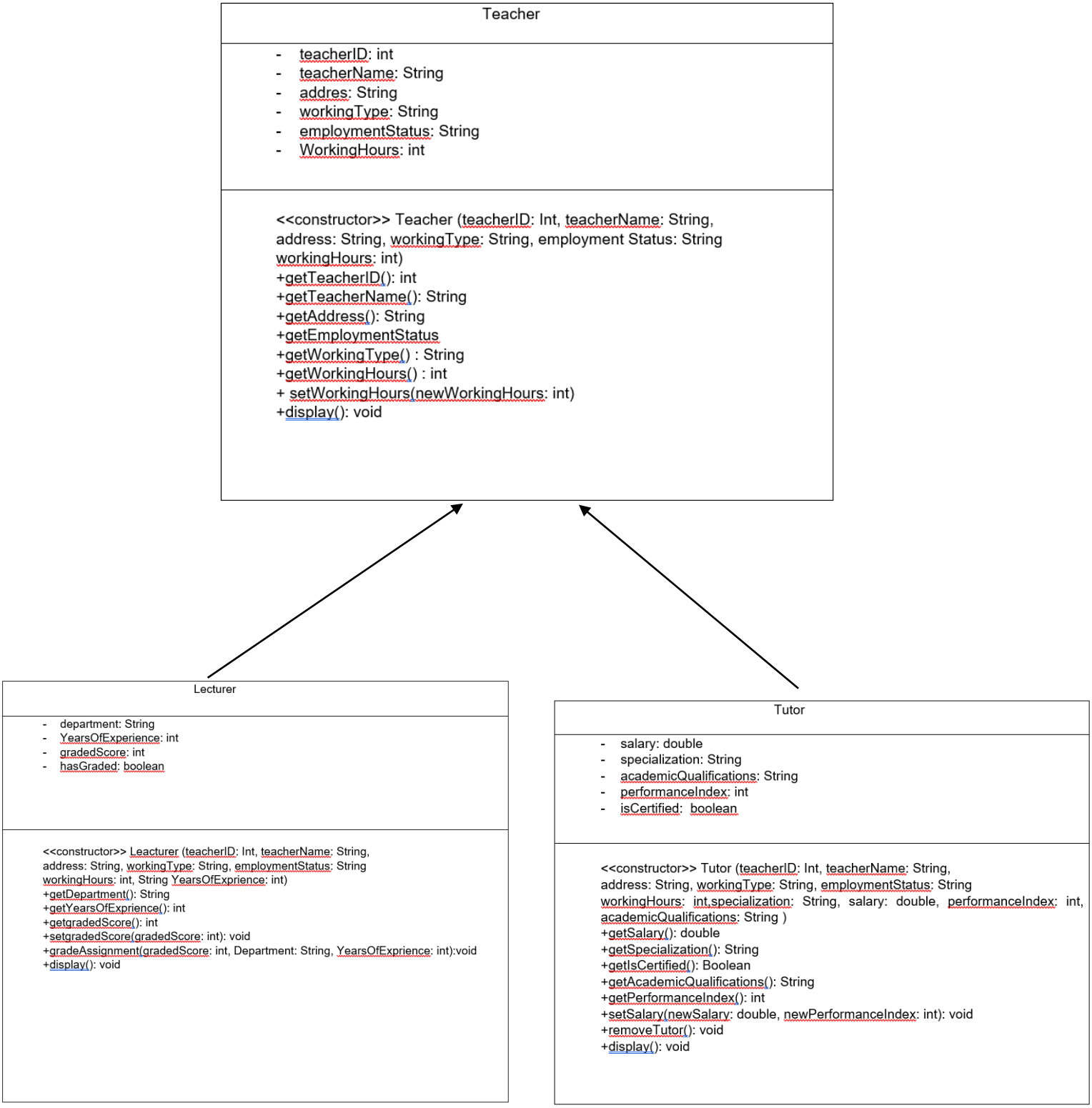
3.3 Tutor Class Diagram

Table 3 Table of Class Diagram of Tutor Class

Table 3 Tutor

Tutor
- salary: double - specialization: String - academicQualifications: String performanceIndex: int - isCertified: boolean
<<constructor>> Tutor (teacherID: Int, teacherName: String, address: String, workingType: String, employmentStatus: String workingHours: int,specialization: String, salary: double, performanceIndex: int, academicQualifications: String) +getSalary(): double +getSpecialization(): String +getIsCertified(): Boolean +getAcademicQualifications(): String +getPerformanceIndex(): int +setSalary(newSalary: double, newPerformanceIndex: int): void +removeTutor(): void +display(): void

3.4 Connection Class Diagrams



4. Pseudocode

4.1 Pseudocode of Teacher

```
CREATE class Teacher

Do

        DECLARE private integer teacherID

        DECLARE private integer workingHours

        DECLARE private String teacherName

        DECLARE private String workingType

        DECLARE private String employmentStatus

        DECLARE private String address

END DO

CREATE a constructor which name is same as the class name i.e because we are
using constructor overloading method (integer teacherID,String teacherName, String
workingType,String employmentStatus, String address)

DO

     CREATE VARIABLE teacherID

     CREATE VARIABLE teacherName

     CREATE VARIABLE workingType

     CREATE VARIABLE employmentStatus

     CREATE VARIABLE address

END DO

ACCESSOR METHOD getTeacherID()

   DO

        return this.teacherID

END DO
```

```
ACCESSOR METHOD getTeacherName ()

 DO
        return this.teacherName
END DO
ACCESSOR METHOD getAddress ()
   DO
        return this.address
END DO

ACCESSOR METHOD getWorkingType ()
   DO
        return this.workingType
END DO

ACCESSOR METHOD getWorkingHours()
   DO
        return this.workingHours
END DO

ACCESSOR METHOD getEmploymentStatus ()
   DO
        return this.employmentStatus
END DO
DO
```

```
ACCESSOR METHOD setWorkingHours(integer newWorkingHours)
        this.workingHours: newWorkingHours
END DO
ACCESSOR METHOD display()
DO
    PRINT TeacherID
    PRINT Address
    PRINT WorkingType
    PRINT TeacherName
    PRINT WorkingHours
    PRINT EmploymentStatus

END DO

IF working hour is greater than 0
DO
PRINT workingHours
END DO

ELSE
DO
PRINT “Value not assigned”
END DO
```

4.2 Pseudocode of Lecturer

```
CREATE class Lecturer EXTENDS Teacher
DO
    DECLARE String Department
    DECLARE integer YearOfExperience
    DECLARE integer gradeScore
    DECLARE Boolean hasGraded
END DO
```

CREATE a constructor which name is same as the class name i.e because we are using constructor overloading method (integer teacherID, String teacherName,String address,String workingType,String employmentStatus, String department, int YearsOfExprience,int workingHours)

CALLING constructor of Teacher class (integer teacherID, String teacherName, String workingType, String employmentStatus, String address)

```
DO
        CREATE VARIABLE department
        CREATE VARIABLE YearsOFExperience
        SET VARIABLE gradedScore as 0
        SET VARIABLE hasGraded as 0
END DO

ACCESSOR METHOD getDepartment()
DO
    return this.department
END DO
```

```
ACCESSOR METHOD getgradedScore ()



DO
    return this.gradedScore
END DO

ACCESSOR METHOD getYearsOfExeprience ()
DO
     return this.YearsOfExperience
END DO



ACCESSOR METHOD gethasGraded()
DO
    return this.hasGraded
END DO

ACCESSOR METHOD setGradedScore (int gradedScore)
DO
      this.gradedScore : gradedScore
END DO
```

```
ACCESSOR METHOD gradeAssignment(integer gradedScore, String Department,
integer YearsOfExperience)
DO
    IF YearsOfExperience is greater then or equals to 5 and this.department.equals to
department
DO
IF gradedScore is greater than or equals to 70
DO
    PRINT A
END DO
ELSE IF gradedScore is greater than or equals to 60
DO
    PRINT B
END DO
ELSE IF gradedScore is greater than or equals to 50
DO
    PRINT C
END DO
ELSE IF gradedScore is greater than or equals to 40
DO
    PRINT D
END DO
ELSE IF gradedScore is greater than or equals to 30
DO
    PRINT E
END DO
```

```
ELSE
DO
   PRINT Grade Not Assigned
END DO
ACCESSOR METHOD display()
DO
CALL display of teacher class()
PRINT department
PRINT YearsOfExperience
PRINT gradedScore
IF hasGraded then
   PRINT hasGraded
ELSE
PRINT Grade is not Assigned

END DO
```

4.3 Pseudocode of Tutor Class

```
CREATE class TUTOR EXTENDS Teacher
DO
    DECLARE double salary
    DECLARE String specialization
    DECLARE String academicQualifications
    DECLARE int performanceIndex
    DECLARE Boolean isCertified
END DO
CREATE a constructor which name is same as the class name i.e because we are
using constructor overloading method (integer teacherID,String teacherName, String
address, integer workingHours, String workingType, String employmentStatus, String
specialization, double salary, integer performanceIndex, String academicQualifications)

CALLING constructor of Teacher class (int teacherID, String teacherName, String
workingType, String employmentStatus, String address)
        CREATE VARIABLE salary
        CREATE VARIABLE specialization
        CREATE VARIABLE academicQualification
        CREATE VARIABLE performanceIndex
        SET isCertified as false

ACCESSOR METHOD getSalary()
DO
    Return this.salary
END DO
```

```
ACCESSOR METHOD getSpecialization()
DO
    return this.specialization
END DO
ACCESSOR METHOD getIsCertified()
DO
    return this.isCertified
END DO

ACCESSOR METHOD getAcademicQualifications()
DO
    return this.academicQualifications
END DO

ACCESSOR METHOD getPerformanceIndex()
DO
return this.performanceIndex
END DO

ACCESSOR METHOD setSalary (double newsalary,integer newPerformanceIndex)
IF isCertified is false
DO
Update appraisalPercentage
IF newPerformanceIndex is greater than equals to 5 and getWorkingHours is greater
than 20
DO
UPDATE appraisalPercentage=5
END DO
```

```
ELSE IF newPerformanceIndex is greater than equals to 8 and newPerformanceIndex
is less than equals to 9
DO
    UPDATE appraisalPercentage=10
END DO
ELSE
DO
   UPDATE appraisalPercentage=20
END DO
SET salary = appraisalPercentage
SET double  appraisal = (appraisalPercentage/100)* newsalary
UPDATE isCERTIFIED =true
PRINT getSalary
PRINT The Tutor doesn't meet the qualification so, the Salary can't be approved
END DO
ELSE
DO
    PRINT Salary can't be approved because the Teacher is Already Certified
END DO

METHOD removeTutor()
 IF isCertified is false
DO
   SET salary=0
  SET specialization=” ”
  SET academicQualification=” “
  SET performanceIndex=0
  SET isCERTIFIED = false
END DO
```

```
ELSE
DO
  PRINT Tutor is certified, Removal cannot be taken place
END DO

DISPLAY METHOD display()
DO
IF isCertified is false
CALL display of Teacher class ()
PRINT salary
PRINT specialization
PRINT academicQualifications
PRINT performanceIndex
END DO
```

5 Method Description

5.1 Teacher Method Description

Table 5 Table of Method Description of Teacher Class

Table 4 Teacher Method Description

Teacher (Constructor)	Constructors are used to create the object of a class.
getTeacherID	This method helps in retrieving the value of TeacherID whose access is private.
getTeacherName	This method helps in retrieving the value of teacherName whose access is private.
getAddress	This method helps in retrieving the value of Address whose access is private.
getWorkingType	This method helps in retrieving the value of workingType whose access is private.
getWorkingHours	This method helps in retrieving the value of workingHours whose access is private.
getEmploymentStatus	This method helps in retrieving the value of employmentStatus whose access is private.
setWorkingHours	This method sets the workingHours attribute with the value of newWorkingHours parameter.
display	This method is used for displaying information about the objects of teacher class. It also checks if the working hour has been assigned or not.

5.2 Lecturer Method Description

Table 6 Table of Method Description of Lecturer Class

Table 5 Method Description of Lecturer

Lecturer (Constructor)	Constructors are used to create the object of a class.
getDepartment	This method helps in retrieving the value of department whose access is private.
getgradedScore	This method helps in retrieving the value of gradedScore whose access is private.
getYearsOfExprience	This method helps in retrieving the value of YearsOfExprience whose access is private.
gethasGraded	This method helps in retrieving the value of workingType whose access is private.
setGradedScore	This method helps in set and update the value of gradeScore attribute with the value of integer gradedScore parameter.
gradeAssignment	This method sets the grade score based on the parameters(integer gradedScore, String Department, integer YearsOfExperience. This grading is conducted it the leacturer has five years of experience then the method grades the assignment based on the given conditions in the nested if, else if and else conditions.
display	This method is used for displaying information about the objects of Lecturer class. This method shows the information about Department, YearsOfExperience, gradedScore.

5.3 Tutor Method Description

Table 7 Table of Method Description of Tutor Class

Table 6 Method Description of Tutor

Tutor (Constructor)	Constructors are used to create the object of a class.
getSalary	This method helps in retrieving the value of salary whose access is private.
getSpecialization	This method helps in retrieving the value of specialization whose access is private.
getIsCertified	This method helps in retrieving the value of iscertified whose access is private.
getAcademicQualifications	This method helps in retrieving the value of academicQualifications whose access is private.
getPerformanceIndex	This method helps in retrieving the value of performanceIndex whose access is private.
setSalary	This method helps in set and update salary of Tutor based on the condition.
removeTutor	This method is helps in removal of tutor in case tutor is not certified.
display	This method is used for displaying information about the Tutor if it is not certified.

6. Testing

6.1 Inspect the Lecturer class, grade the assignment, and re-inspect the Lecturer Class

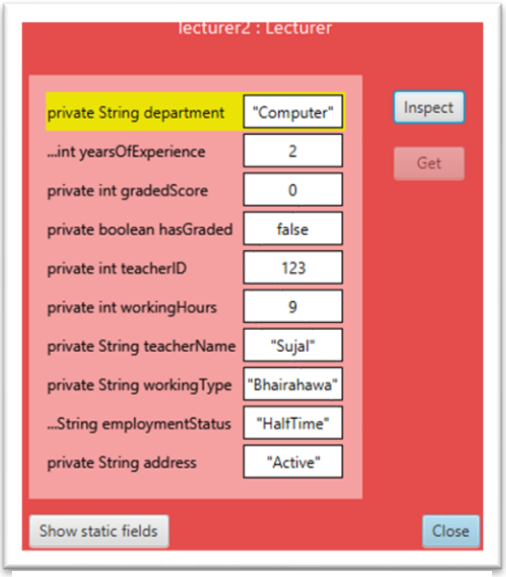


Figure 1

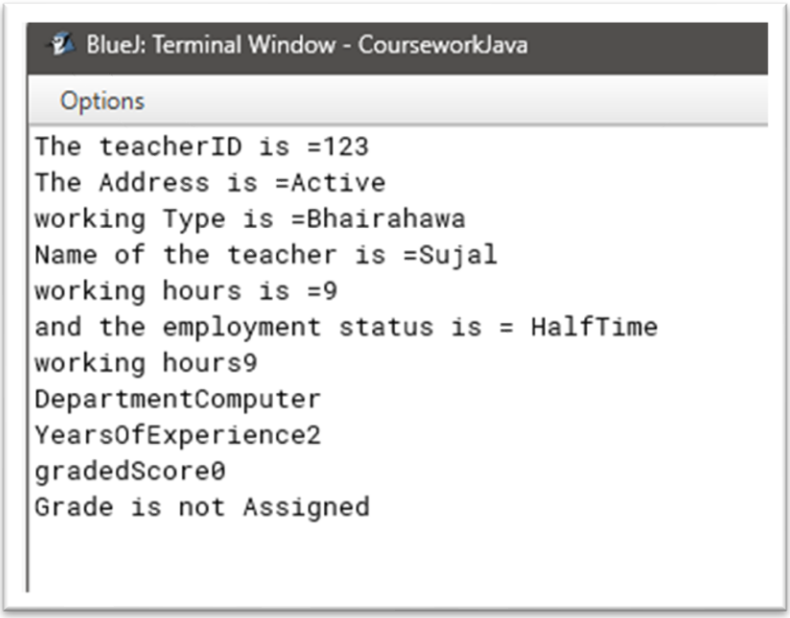


Figure 2 Inspecting Lecturer Class through Terminal

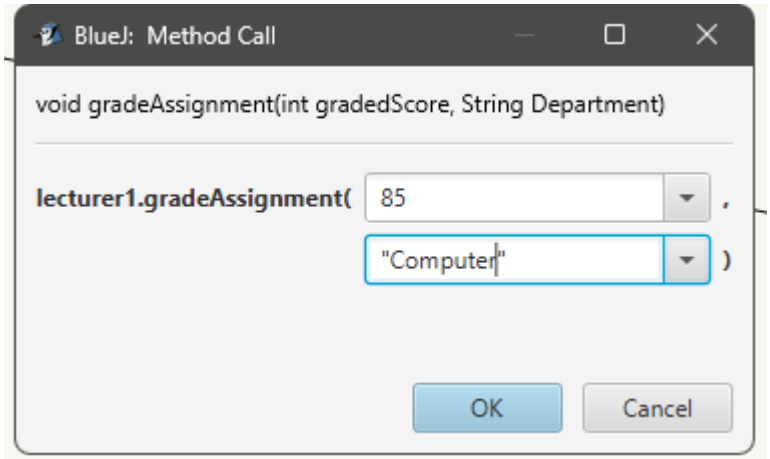


Figure 3 Adding Grade to the Assignment

Figure 4 Output after Grading

6.1.1 Table of testing of Lecturer class

Table 8 Testing table of Lecturer class

Table 7 Testing of Lecturer

Test	Lecturer Class
Action	**The lecturer is called with the following arguments: -** Teacher ID = 123 The Address is = Active Working Type= Bhairahawa Name of the teacher is =Sujal Employment status = Halftime Department = Computer Years Of Experience= 6 **Grading the Assignment: -** Graded Score= 85 Grade= A
Expected Result	When we input the grade then the program determines the grade score according to the statements that we had in our class and displays the message accordingly.
Actual Result	When we input the grade then the program determines the grade score according to the statements that we had in our class and displays the message accordingly
Conclusion	Our Test was a Success. There are no errors to be found.

6.1.2 Inspecting Tutor class, setting salary and reinspect the class.

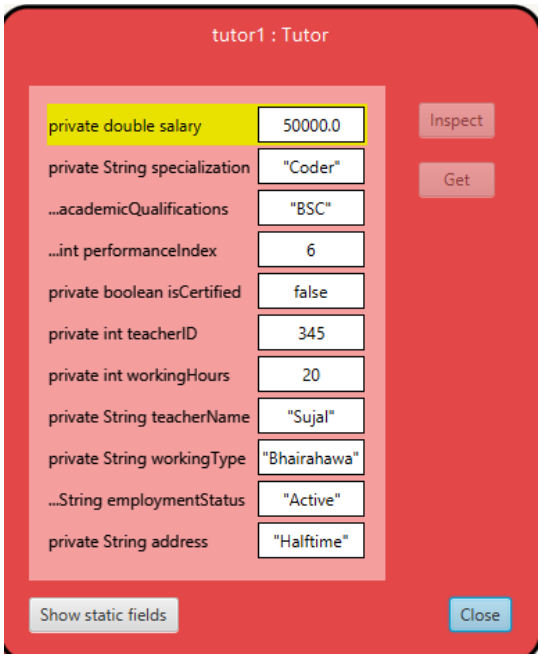


Figure 5 Inspecting Tutor Class's Salary

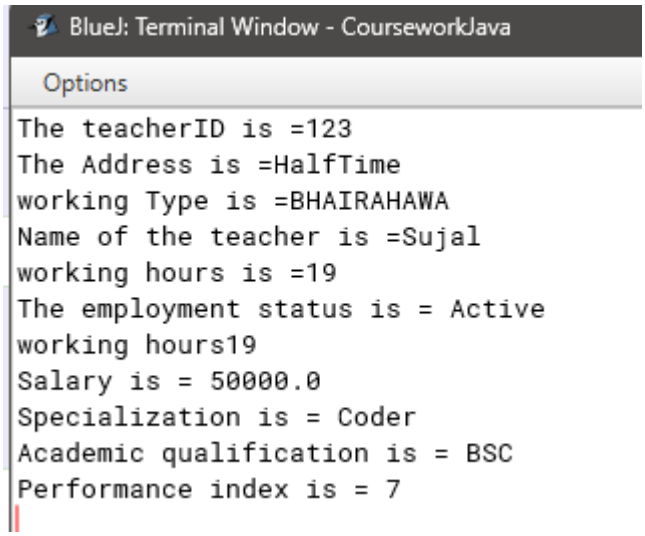


Figure 6 Terminal View of Tutor Class

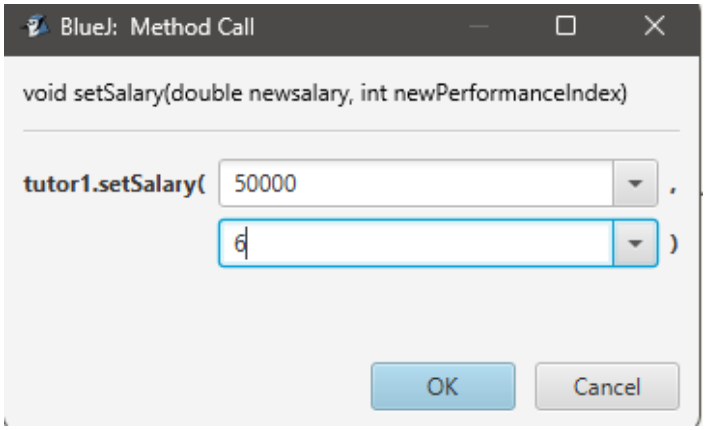


Figure 8 setting the salary and performance index.

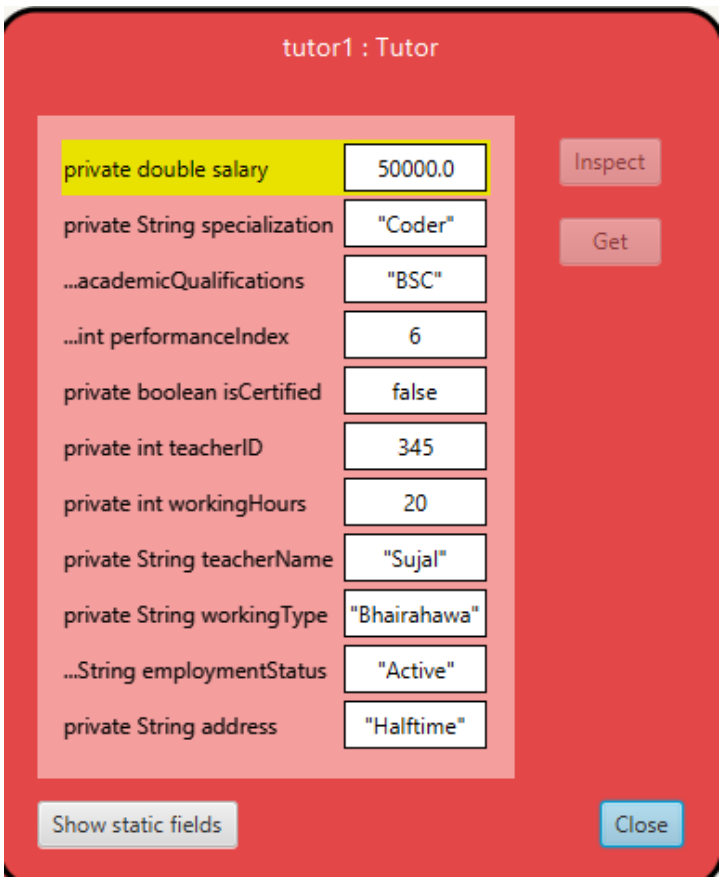


Figure 7 Reinspecting the Tutor class.

6.2 Table of Test of Tutor class

Table 9 Table of testing of Tutor class

Table 8 Testing of Tutor

Test	Tutor Class
Action	**The Tutor is called with the following arguments: -** Teacher ID = 345 The Address is = Active Working Type= Bhairahawa Name of the teacher is =Sujal Employment status = Halftime Salary is = 0 Specialization is = Coder Academic qualification is = BSC Performance index is = 7 **After Salary Calculation: -** Graded Score= 50000 Grade= A
Expected Result	After providing performance index and working hours then the salary is calculated according to the appraisal Percentage. Then it shows suitable message.
Actual Result	After providing performance index and working hours then the salary is calculated according to the appraisal Percentage. Then it shows suitable message.
Conclusion	Our Test was a Success. There are no errors to be found.

6.3 Inspecting Tutor class again after removing the Tutor.

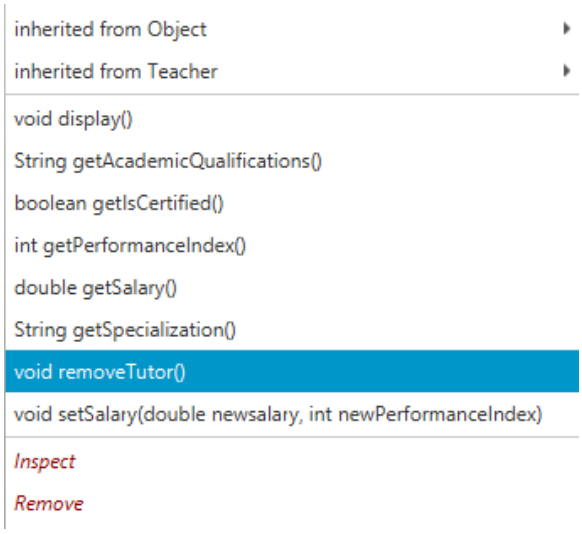


Figure 9 Process of Tutor removal

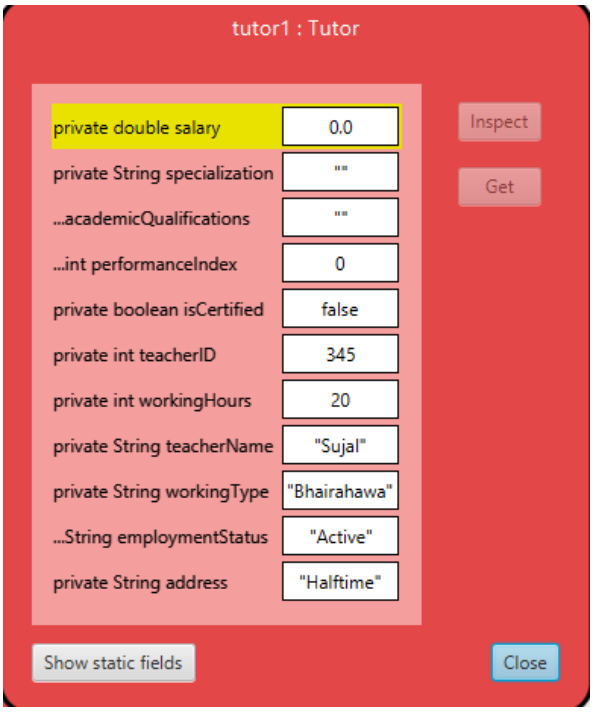


Figure 10 Reinspecting Tutor after its removal

6.4 Displaying the details of Lecturer and Tutor class.

```
BlueJ: Terminal Window - CourseworkJava
Options
The teacherID is =123
The Address is =Active
working Type is =Bhairahawa
Name of the teacher is =Sujal
working hours is =9
and the employment status is = HalfTime
working hours9
DepartmentComputer
YearsOfExperience2
gradedScore0
Grade is not Assigned
```

Figure 11 Terminal view of Lecturer class

```
BlueJ: Terminal Window - CourseworkJava
Options
The teacherID is =123
The Address is =HalfTime
working Type is =BHAIRAHAWA
Name of the teacher is =Sujal
working hours is =19
The employment status is = Active
working hours19
Salary is = 50000.0
Specialization is = Coder
Academic qualification is = BSC
Performance index is = 7
```

Figure 12 Terminal view of Tutor class

6.4.1 Table of test details of Lecturer and Tutor classes

Table 9 Testing details of Lecturer class and Tutor classes

Table 9 Testing of Lecturer and Tutor

Test	Lecturer and Teacher classes
Action	**The lecturer is called with the following arguments: -** Teacher ID = 123 The Address is = Active Working Type= Bhairahawa Name of the teacher is =Sujal Employment status = Halftime Department = Computer Years Of Experience= 6 **The Tutor is called with the following arguments: -** Teacher ID = 345 The Address is = Active Working Type= Bhairahawa Name of the teacher is =Sujal Employment status = Halftime Salary is = 0 Specialization is = Coder Academic qualification is = BSC Performance index is = 7
Expected Result	The Program entered will show the details of Lecturer and Tutor class
Actual Result	The Program entered will show the details of Lecturer and Tutor class
Conclusion	Our Test was a Success. There are no errors to be found.

7. Error

There are very types of error in programming some of them as follows: -

1. Syntex Error
2. Logical Error
3. Semantics Error

7.1 Syntex Error

It refers to an error in the syntax of the program. It is one of the most common errors that happens when we code as a beginner.

Some of the common examples of syntax error includes:

- Incorrect spelling of keywords
- Missing semicolon
- Improper use of quotes
- Missing close brackets

7.1.1 Syntex Error Detection (Spelling Incorrect)

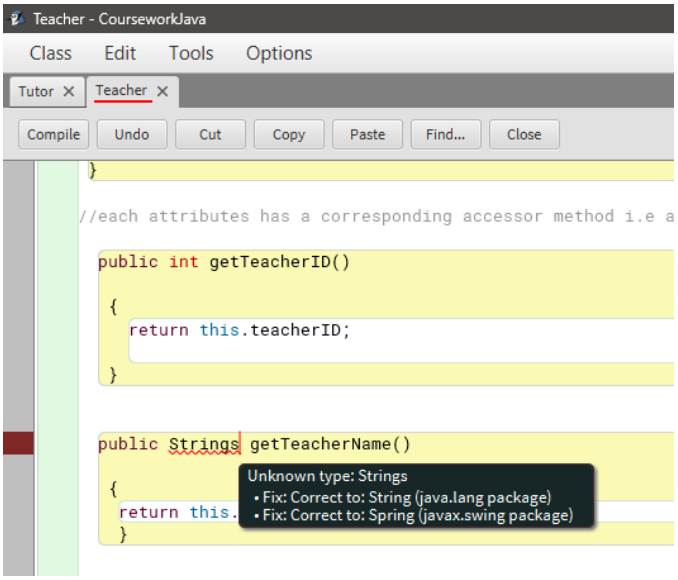


Figure 13 (i) Syntex Error Detection

7.1.2 Syntex Error Detection (Missing bracket)

```
public int getWorkingHours()

 {
  return this.workingHours;


public String getEmploymentStatus()
 illegal start of expression
```

Figure 14 (ii) Syntex Error Detection

7.2 Logical Error

It refers to an error in the logic of a program. It is also one of the most difficult problems to find a solution.

7.2.1 Logical Error Detection

```
      double appraisalPercentage;

if(newPerformanceIndex <= 5 && getWorkingHours()>20)


 {
     if (newPerformanceIndex <=7)

{
    appraisalPercentage=5;

  }
```

Figure 15 (i) Logical error detection

7.2.2 Logical Error Solving

```
if(newPerformanceIndex >= 5 && getWorkingHours()>20)

 {
     if (newPerformanceIndex <=7)

{
    appraisalPercentage=5;
```

Figure 16 Solving the Error by correcting the Logic.

7.3 Semantic Error

It is an error that occurs when the code written doesn't make any sense even though it has corrected syntaxes.

Examples: -

- Using Incorrect formula's
- Using Incorrect datatype

7.3.1 Semantic Error Detection

```
public void setWorkingHours(String newWorkingHours)

 {
  this.workingHours=newWorkingHours;

  }
```

Figure 17 Error: Wrong Datatype

7.3.2 Semantic Error Correction

```
public void setWorkingHours(int newWorkingHours)

 {
  this.workingHours=newWorkingHours;

  }
```

Figure 18 Solution: Correction Datatype

7.3.3 Semantic Error Detection

```
public void display()

{
 if( isCertified )

    {



    super.display();

    System.out.println("Salary is = "+this.salary);

    System.out.println("Specialization is = "+this.specialization);

    System.out.println("Academic qualification is = "+this.academicQualifications);

    System.out.println("Performance index is = "+this.performanceIndex);
}
```

Figure 19 Error: Incorrect Formula

7.3.4 Semantic Error Solution

```
public void display()

{
 if( !isCertified )

    {



    super.display();

    System.out.println("Salary is = "+this.salary);

    System.out.println("Specialization is = "+this.specialization);

    System.out.println("Academic qualification is = "+this.academicQualifications);

    System.out.println("Performance index is = "+this.performanceIndex);
 }
```

Figure 20 Solution: Correcting Formula

8. Conclusion

To sum it all up, this java coursework has been a worthwhile educational experience that has really improved my programming abilities. The process of creating and troubleshooting the code has improved my problem-solving ability. By completing this project, I have gained a deeper understanding of Java. Working on a large project on my own has made me an independent learner. This project helped me to put my academic understanding into practice.

All things considered, this project has greatly aided my academic and future professional understanding by giving me the tools I need to take on challenging programming tasks.

Additionally, the coursework reinforced the importance of software design principles and best practices. I was able to create a maintainable and scalable solution. The experience that I gained from this coursework has not only improved my technical skills but also enhanced my real-world problem-solving skills. This coursework has laid a solid foundation for my future projects and has fostered a greater amount of knowledge for me to use in future.

9. Bibliography

References

About BlueJ. (n.d.). Retrieved from BlueJ: https://www.bluej.org/about.html

What is Java technology and why do I need it? (n.d.). Retrieved from Java || oracle: https://www.java.com/en/download/help/whatis_java.html

10. Appendix

10.1 Teacher Codes

```
public class Teacher//this class is the parent class

{
//creating 6 attributes of teacher class according to que

 private int teacherID;

 private int workingHours;//applying datatypes according to the que

 private String teacherName;

 private String workingType;

 private String employmentStatus;//applying datatypes according to the que

 private String address;

 /*aboves are all instance variable which have private
 as an access specifier*/
```

```
    public Teacher(int teacherID,String teacherName,String workingType,String
employmentStatus, String address)//making constructor accept 5 parameters according
to the que


    //creating Teacher as a Constructor with above parameters,it is mainly used to
initialize instance variable in a class


    //we are using constructor overloading method because we are using the same
name in constructor which is also our class name i.e "Teacher"


    {


        this.teacherID=teacherID;


        this.teacherName=teacherName;


        this.address=address;


        this.workingType=workingType;


        this.employmentStatus=employmentStatus;


        /*above is "This keyword" which is assigning varibable with parameter values, it is
used inside the constructor


       to call another overloaded constructor in the same class which is "Teacher" in our
case*/



    }
```

//each attributes has a corresponding accessor method i.e a getter method according to que so below we applied getter method

```
public int getTeacherID()

{
  return this.teacherID;

}


public String getTeacherName()

{
 return this.teacherName;
 }


public String getAddress()

 {
 return this.address;
 }


public String getWorkingType()

{
```

```
  return this.workingType;

  }


public int getWorkingHours()

 {
  return this.workingHours;

 }


public String getEmploymentStatus()

 {
  return this.employmentStatus;

  //getter method is used to view/access and return the value of data/attributes
 }

 /*method to set the workinghrs which accepts a new workinghrs as a parameter

 i.e we need to create a setter method(mutator method) */

public void setWorkingHours(int newWorkingHours)

 {
```

```
this.workingHours=newWorkingHours;

}

//display method to display the output

public void display()
{
 System.out.println("The teacherID is =" +getTeacherID() +"\nThe Address is ="
 +this.getAddress()+ "\nworking Type is =" + this.getWorkingType() + "\nName of the teacher is ="
 + this.getTeacherName()+  "\nworking hours is =" + this.getWorkingHours() +
 "\nthe employment status is = " + this.getEmploymentStatus());



//using if and else condition to display a suitable message when working hour is not assigned and when it is assigned

if(workingHours>0)

{
    System.out.println("working hours" + workingHours); //which it is assigned
}

else

{
    System.out.println("Value not assigned");  //when not assigned
}}}
```

10.2 Lecturer Codes

```
public class Lecturer extends Teacher
/*here Lecturer is a sub class (child class) of Teacher which is a parent class,
To achieve inheritance we use the keyword extends*/

{

  private String department;

  private int yearsOfExperience;

  private int gradedScore;

  private boolean hasGraded;

  //these are all instance variable which there respective datatypes according to the
que

   public Lecturer(int teacherID, String teacherName,String address,String
workingType,String employmentStatus, String department,int yearsOfExperience,int
workingHours)

  /*creating Leacturer as a Constructor with above parameters,it is mainly used to
initialize instance variable in a class.
   *
  we are using constructor overloading method because we are using the same name
in constructor which is also our class name.*/
```

```
    {
        super(teacherID,teacherName,address,workingType,employmentStatus);

        this.department=department;

        this.yearsOfExperience=yearsOfExperience;

        this.gradedScore=0;

        this.hasGraded=false;

        //instance varibale to parameter
    }

    //each attributes has a corresponding accessor method i.e a getter method according
to que so below we applied getter method

    public String getDepartment()

    {
        return this.department;
    }

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

```
public int getYearsOfExperience()
{
    return this.yearsOfExperience;
}

 public boolean getHasGraded()
{
    return this.hasGraded;

     //getter method is used to view/access and return the value of data/attributes
}

   //creating a mutator method for attribute gradedScore i.e a setter method

   public void setGradedScore(int gradedScore)

   {
       this.gradedScore=gradedScore;

   }

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

   {
       //if ,else if and else statement to print the output which is shown in the question

       if(yearsOfExperience >=5 && this.department.equals(department))
```

```
{
    if (gradedScore >=70)

    {
        System.out.println("A");
    }

    else if(gradedScore >=60)

    {
        System.out.println("B");

    }

    else if(gradedScore >=50)

    {
        System.out.println("C");

    }

    else if(gradedScore >=40)

    {
        System.out.println("D");

    }
```

```
        else if(gradedScore >=30)

        {
            System.out.println("E");

        }

        else

        {
            System.out.println("Grade Not Assigned");

        }

        //overriding

    }
}

//using diplay method to dislay the output

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

        //here parent class is called from display method to override

        System.out.println("Department" +this.department);
```

```
        System.out.println("YearsOfExperience" +this.yearsOfExperience);

        System.out.println("gradedScore" +this.gradedScore);

        if(hasGraded)

        {
            System.out.println(this.hasGraded);
            System.out.println("Grading was a success");

        }

        else

        {
           System.out.println("Grade is not Assigned");

        }

    }

}
```

10.3 Tutor Codes

```
//here Tutor is also the sub class of teacher

public class Tutor extends Teacher
{
    //creating 5 attributes of teacher class according to que

    private double salary;    //applying datatypes according to the que

    private String specialization;

    private String academicQualifications;   //applying datatypes according to the que

    private int performanceIndex;    //applying datatypes according to the que

    private boolean isCertified;   //applying datatypes according to the que

    /*aboves are all instance variable which have private
    as an access specifier*/

    public Tutor(int teacherID,String teacherName, String address, int workingHours,

    String workingType,String employmentStatus, String specialization,
```

double salary, int performanceIndex, String academicQualifications) //making constructor accept 10 parameters according to the que

```
{
    super (teacherID,teacherName,address,employmentStatus, workingType);

    setWorkingHours(workingHours);

    this.salary=salary;

    this.specialization=specialization;

    this.academicQualifications=academicQualifications;

    this.isCertified=false;  //according to que it is set to false

    this.performanceIndex=performanceIndex;
```

/*above is "This keyword" which is assigning varibable with parameter values, it is used inside the constructor

to call another overloaded constructor in the same class which is "Teacher" in our case*/

```
}
```

//each attributes has a corresponding accessor method i.e a getter method according to que so below we applied getter method

```
    public double getSalary()
{
```

```
    return this.salary;
}

 public String getSpecialization()
  {
    return this.specialization;
 }

 public boolean getIsCertified()
 {
     return this.isCertified;
 }
  public String getAcademicQualifications()
 {
    return this.academicQualifications;
```

```
}

  public int getPerformanceIndex()

{
  return this.performanceIndex;

 //getter method is used to view/access and return the value of data/attributes

}

//above is the method to set salary
public void setSalary(double newsalary,int newPerformanceIndex)

{
    //below we used nested if condition because we use nested if condition when
execution of one condition depends upon other condition
 if(!isCertified)

 {
      double appraisalPercentage;

 if(newPerformanceIndex >= 5 && getWorkingHours()>20)



  {
     if (newPerformanceIndex <=7)
```

```
{
    appraisalPercentage=5;

  }

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

{

  appraisalPercentage=10;

}

else

 {

  appraisalPercentage=20;

}

   this.salary += appraisalPercentage;
   double appraisal = (appraisalPercentage/100) * newsalary;
   this.isCertified=true;
   System.out.println("Salary" + this.getSalary());
   System.out.println("The Tutor doesn't meet the qualification so, the Salary
can't be approved ");

 }
```

```
            else

             {
                System.out.println("Salary can't be approved because the Teacher is
Already Certified");



             }



         }
        }



        //below is a method which will remove the tutor (only if the tutor has not been
certified yet)

           public void removeTutor()

           {
              if( !isCertified )

             {
                this.salary=0;

                this.specialization="";
```

```
      this.academicQualifications="";

      this.performanceIndex=0;

      this.isCertified=false;
    }



     else

    {
          System.out.println("Tutor is certified, Removal cannot be taken place");
    }
}

  //display method to display the output

 public void display()

{
  if( !isCertified )

     {




     super.display();
```

```
        System.out.println("Salary is = "+this.salary);

        System.out.println("Specialization is = "+this.specialization);

        System.out.println("Academic qualification is = "+this.academicQualifications);

        System.out.println("Performance index is = "+this.performanceIndex);
      }

    }

}
```