LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CC4001NI Programming

COURSEWORK-1

Assessment Weightage & Type

30% Individual Coursework

Semester and Year

Spring 2021

Student Name:

Group: N3

London Met ID: 20049202

College ID: NP01NT4S210071

Assignment Due Date: 23rd may

Assignment Submission Date: 23rd may

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

Contents

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Introduction to the topic

Java is a high-level programming language. It was created with the intention of developing programs for set-top boxes and handheld devices, but it quickly gained popularity as a platform for developing web applications.

The Java programming language has a syntax close to C++, but it is purely an object-oriented programming language. Most Java programs, for example, contain classes that describe objects and methods that are assigned to specific classes. Java is also known for being stricter than C++, requiring precise definitions of variables and functions. This means that Java source code is more likely than other languages to create errors or "exceptions," but it also restricts the types of errors that can be caused by unknown variables or unassigned types (MCKenzie, april 2019).

This is the first coursework of the module “Programming”. This coursework is done by using the software ‘bluej’. The task of this coursework is to create a class named course which consists of two child classes academic course and nonacademic course. The program consists of particular methods like constructors, accessors and mutators which allows us to choose particular courses, lecturers and courseleader. The courses is an parent class and academic and nonacademic courses are the child of the parent class. The accessor methods are used to return the value and mutator methods are used to assign the new values.

The constructors of the classes are assigned with the parameters which are to be accepted and the attributes are also assigned with different values. Each of the attributes of all classes have the getter and setter method that helps to return the value and assign the values to the attributes.

2. CLASS DIAGRAM

2.1. Introduction

A Class diagram is a model for creating an object or collection of objects. What an object can do is described by its class. It's a template for making different artifacts and putting their actions into the scheme. A class diagram is represented by a rectangle with rows containing class names, attributes, and operations. The Class Diagram depicts the various types of objects in the structure as well as the various types of relationships between them. It provides a high-level overview of a program. Almost all Object-Oriented Methods can be used for this modeling tool. Another class may be referred to by a class. (guru99, 2019)

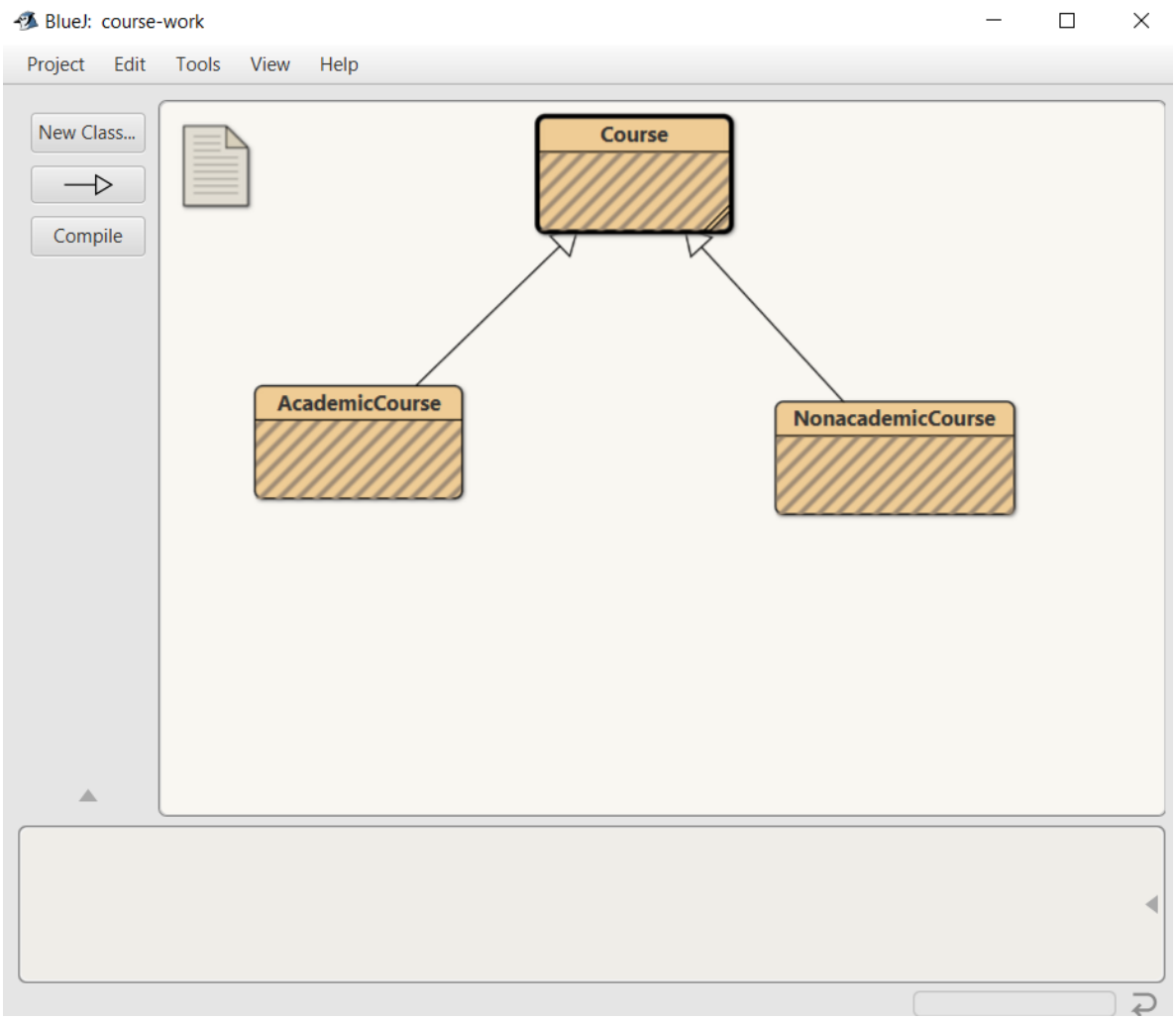

Figure 1: Class Diagram

- Courses

Course
- courseId : String - courseName : String - duration : int - courseLeader : String
+ course(String courseId, String courseName, int duration) + getCourseId() : String + getCourseName() : String + getDuration() : int + getLeader() : String + setLeader(String courseLeader) : void + display : void

Table 1: Table of course class

- **AcademicCourse**

AcademicCourse
- lecturerName : String - level : String - credit : String - startingDate : String - completionDate : String - numberOfAssesments : int - isRegistered : boolean - isRemoved : boolean
+ AcademicCourse(String courseId, String courseName, int duration, String level, String credit, int numberOfAssesments) + getLecturerName() : String + getLevel() : String + getCredit() : String + getStartingDate() : String + getCompletionDate() : String + getnumberOfAssesments() : int + getIsRegistered() : boolean + getIsRemoved() : boolean + setLecturerName(String lecturerName) : void + setNumberOfAssesments(int numberOfAssesments) : void + registerCourse(String courseLeader, String instructorName, String startngDate, String completionDate) : void + display() : void

Table 2: Table of AcademicCourse class

- NonAcademicCourse

NonAcademicCourse
- instructorName : String - startingDate : String - completionDate : String - examDate : String - prerequisite : String - duration : int - isRegistered : boolean - isRemoved : boolean
+ NonacademicCourse(String courseId, String courseName, int duration, String prerequisite) : + getInstructorName() : String + getDuration() : int + getStartingDate() : String + getCompletionDate() : String + getExamDate() : String + getPrereuisite() : String + getIsRegistered() : boolean + getIsRemoved() : boolean + setInstructorName(String instructorName) : void + registerCourse(String courseLeader, String instructorName, String startngDate, String completionDate, String examDate): void + removeCourse() : void + display() : void

Table 3: Table of NonAcademicCourse class

Figure 2: Class Diagram of table of classes

3. Pseudocode

Pseudocode is an easy way to write English programming code. Pseudocode is not a programming language in the traditional sense. Short phrases are used to write code for applications until they are written in a particular language. You can use pseudocode to generate statements to produce the desired outcomes for your program until you know what it's about and how it'll work. (Airth, 23 january, 2020)

3.1. Class: Course

```
DEFINE class course

DEFINE four instant variables as String courseId, String courseName, String courseLeader and int duration.

CREATE a constructor course and initialize variables

    INITIALIZE courseId as String type

    INITIALIZE courseName as String type

    INITIALIZE courseLeader as String type

    INITIALIZE duration as int type

DEFINE method getCourseId() as String type

    IF courseId == “”

        RETURN “empty”

    IF END

        RETURN courseId

DEFINE method getCourseName() as String type

    DO

        EXTRACT the value of courseName
```

```
                RETURN courseName
        END DO
        DEFINE method getduration() as int type
            DO
                EXTRACT the value of duration
                RETURN duration
            END DO
        DEFINE method setLeader() as String type
            DO
                SET the courseLeader
            END DO
        DEFINE method getLeader() as Sting type
            DO
                EXTRACT the value of courseLeader
                RETURN courseLeader
            END DO
        DEFINE method display ()
            DO
                PRINT “Course Details”
                PRINT “CourseID:” + couseId
                PRINT “CourseName:”+ courseId
            IF courseLeader == “”
                PRINT “CourseLeader:” + courseLeader
            IF END
                PRINT “Duration:” + duration
            END DO
```

3.2. Class: AcademicCourse

DEFINE class AcademicCourse

DEFINE eight instant variables as String lecturerName, String level, String credit, String startingdate, String completionDate, int numberOfAssesments, Boolean is registered and Boolean is removed

CREATE a constructor AcademicCourse and initialize variables (String courseId, String courseName, int duration, String level, int numberOfAssesments)

DECLARE variable of parent class (String courseId, String courseName, int duration)

INITIALIZE level as String type
INITIALIZE credit as String type
INITIALIZE numberOfAssesments as int type
INITIALIZE lecturerName as empty String
INITIALIZE startingDate as empty String
INITIALIZE completionDate as empty String
INITIALIZE isRegistered as false value
INITIALIZE isRemoved as true value

DEFINE method setLecturerName () as String type

DO
SET lecturer name
END DO

DEFINE method getlecturerName () as String type

DO
EXTRACT the value of lecturer name set
RETURN lecturerName
END DO

DEFINE method setNumberOfAssesments () as int type

DO

```
                SET number of assesments to be given
        END DO
DEFINE method getNumberOfAssesments () as int type
        DO
                EXTRACT the value of numberOfAssesments
                RETURN numberOfAssesments
        END DO
DEFINE method getLevel () as String type
        DO
                EXTRACT the value of level
                RETURN level
        END DO
DEFINE method getCredit () as String type
        DO
                EXTRACT the value of credit
                RETURN credit
        END DO
DEFINE method getStartingDate () as String type
        DO
                EXTRACT the value of starting Date
                RETURN startingDate
        END DO
DEFINE method getCompletionDate () as String type
        DO
                EXTRACT the value of completion date
                RETURN completionDate
        END DO
DEFINE method getIsRegistered () as boolean

        DO

                RETURN isRegistered

        END DO
```

DEFINE method getIsRemoved () as boolean

DO

RETURN isRemoved

END DO

DEFINE method registerCourse(String courseLeader, String lecturerName, String startingDate, String completionDate)

IF IsResistered is false

DO

SET Leader to instant variable courseLeader

SET lecturerName to instant variable

lecturerName

SET startingDate to instant variable to staringDate

SET completion date to instant variable to completionDate

isResistered = true

isRemoved = false

END DO

ELSE

DO

PRINT “This course is being registered to” + lecturerName+ “to complete within” + startingDate + “to” + completionDate

END DO

DEFINE method display ()

IF isResistered is true

```
        DO
            Super.display ()
            PRINT “Lecturer Name:” + lecturerName
            PRINT “Level:” + level
            PRINT “Credit:” + credit
            PRINT “Starting Date:” + startingDate
            PRINT “Completion Date:” + completionDate
        END DO
    ELSE
        DO
            Super.display()
        END DO
```

3.3. Class: NonAcademicCourse

DEFINE class nonAcademicCourse

DEFINE eight instant variables (String instructorName, int duration, String startingDate, String completionDate, String examDate, String prerequisite,

boolean isResistered, boolean isRemoved)

DEFINE constructor NonAcademicCourse and initialize variables (String courseId, String courseName, int duration, String prerequisite)

DECLARE variable of parent class (String courseId, String courseName, int duration)

INITIALIZE prerequisite as String type

INITIALIZE instructorName as empty String

INITIALIZE startingName as empty String

INITIALIZE completionDate as empty String

```
        INITIALIZE examDate as empty String
        INITIALIZE isRegistered as false value
        INITIALIZE isRemoved as false value
    DEFINE method setInstructorName() as String type
        DO
            IF isRegistered is false
                SET instructorName to instant variable instructorName
            ELSE
                PRINT "This course is registered so instructor name
    cannot be updated"
        END DO
    DEFINE method getInstructorName () as String type
        DO
            EXTRACT the value of instructor
            RETURN instructorName
        END DO
    DEFINE method getduration () as int type
        DO
            EXTRACT the value of duration
            RETURN duration
        END DO
    DEFINE method getStartingDate () as String type
        DO
            EXTRACT the value of starting date
```

```
                RETURN StartingDate
        END DO
    DEFINE method getCompletionDate () as String type
        DO
            EXTRACT the value of completion date
            RETURN completionDate
        END DO
    DEFINE method getExamDate () as String type
        DO
            EXTRACT the value of exam date
            RETURN examDate
        END DO
    DEFINE method getPrerequisite () as String type
        DO
            EXTRACT the value of prerequisite
            RETURN prerequisite
        END DO
    DEFINE method getIsResistered () as Boolean type
        DO
            RETURN isResistered
        END DO
    DEFINE method getIsRemoved () as Boolean type
        DO
            RETURN isRemoved
```

```
        END DO

    DEFINE method registerCourse (String courseLeader, String
    instructorName, String startingDate, String completionDate, String
    examDate)

            IF isRegistered is false

                DO

                    SET setleader to initial variable courseLeader

                    SET instructorName to initial variable

                instructorName

                    SET startingDate to initial variable startingDate

                    SET completionDate to initial variable
                completionDate

                    SET examDate to initial variable examDate

                        isRegistered = true

            END DO

        ELSE

            DO

                PRINT “This course has been registered”

            END DO

    DEFINE method removeCourse ()

        IF isRemoved is true

            DO

                PRINT “This course has been removed”

            END DO

        ELSE
```

```
            DO
                SET leader as empty string
                SET instructorName as empty String
                SET startingDate as empty String
                SET completionDate as empty string
                SET examDate as empty String
                IsRegistered = false
                IsRegistered = true
            END DO
    DEFINE method display ()
        IF isRegistered is true
            DO
                Super.display ()
                PRINT “Instructor Name:” + instructorName
                PRINT “Starting Date:” + startingDate
                PRINT “Completion Date:” + completionDate
                PRINT “ExaminationDate:” + examDate
            END DO
        ELSE
            DO
                Super.display ()
            END DO
```

4

4. Method Description

4.1. Course Class

- **Public Course(String courseId, String courseName, int duration):**
  This method initializes all the variables passed in the parameters.
- **Public String getCourseId():**
  This method returns the value of courseId as String type.
- **Public String getCourseName():**
  This method returns the value of courseName as String type.
- **Public int getDuration():**
  This method returns the value of duration as int type.
- **Public String getLeader():**
  This method returns the value of Leader as String type.
- **Public void setLeader():**
  This method sets the leader and assign input to the specific parameter.
- **Public void display():**
  This method displays the courseId, courseName, courseLeader and duration of the course to be taken.

4.2. AcademicCourse

- **Public AcademicCourse(String courseId, String courseName, int duration, String level, String credit, int number of assesments):**
  This method initializes all the variables passed in the parameters.
- **Public String getLecturerName():**
  This method returns the value of lecturerName as String type.
- **public String getLevel():**
  This method returns the value of level as String type.
- **public String getCredit():**
  This method returns the value of credit as String type.
- **public String getStartingDate():**

This method returns the value of startingDate as String type.

- **public String getCompletionDate():**

  This method returns the value of completionDate as String type.
- **public int getNumberOfAssesments():**

  This method retrurns the value of numberOfAssesments as int type.
- **public boolean getIsRegistered():**

  This method returns boolean value whether it isRegistered or not.
- **public boolean getIsRemoved():**

  This method returns boolean value whether it isRemoved or not.
- **public void setLecturerName(String lecturerName):**

  This method sets the lecturerName and assigns input to specific parameter.
- **public void setNumberOfAssesments(int numberOfAssesments):**

  This method sets the numberOfAssesments and assigns input to specific parameters.
- **public void registerCourse(String courseLeader, String lecturerName, String startingDate, String completionDate):**
- This method registers the value to the specific parameter of the course to be chosen and display courseLeader, lecturerName, StartingDate and completionDate.

- **public void display():**

  This method display the information of the lecturer Name, Level, Credit, Starting Date and Completion Date or else display registerCourse.

4.3. NonAcademicCourse Class

- **Public NonacademicCourse(String courseId, String courseName, int duration, String prerequisite):**

This method initializes all the variables passed in the parameter.

- **Public String getInstructorName():**

  This method returns the value of instructorName as String type.

- **Public int getDuration():**

  This method returns the value of duration as int type.

- **Public String getStartingDate():**

  This method returns the value of startingDate as String type.

- **Public String getCompletionDate():**

  This method returns the value of completionDate as String type.

- **Public String getExamDate():**

  This method returns the value of examDate as String type.

- **Public String getPrerequisite():**

  This method returns the value of prerequisite as String type.

- **Public boolean getIsRegistered():**

  This method returns boolean value whether isRegistered or not.

- **Public boolean getIsRemoved():**

  This method returns boolean value whether isRemoved or not.

- **Public void setInstructorName(String instructorName):**

  This method sets the instructorName and assigns input to specific parameter.

- **Public void registerCourse(String courseLeader, String instructorName, String startingDate, String completionDate, String examDate):**

  This method registers the value to the specific parameter of the course to be chosen and display course Leader, instructorName, starting Date, completionDate and examDate.

- **Public void removeCourse():**

  This method removes the value of the course been input.

- **Public void display():**

  This method display the information of the instructor Name, Starting Date, CompletionDate, ExaminationDate or else displays registered course.

5. Testing

5.1. Test1:

Inspect AcademicCourse class, registor an academic course and re-inspect the AcademicCourse class

Objective	To register academic course
Action	The AcademicCourse is called with following arguments. course ID = “2” courseName = “programming” duration = 4 level = “1” credit = “30” number Of Assessments = 5 Inspection of the AcademicCourse class. Void registerAcademicCourse is called with the following arguments: courseLeader = “Bibek” lecturerName = “Raj” startingDate = “21 may 2020” completionDate = “20 march 2021” Reinspection of the AcademicCourse class.
Expected Result	The course will be registered.
Actual Result	The course has been registered.
Conclusion	Test was successful

Table 4: Table of Test1

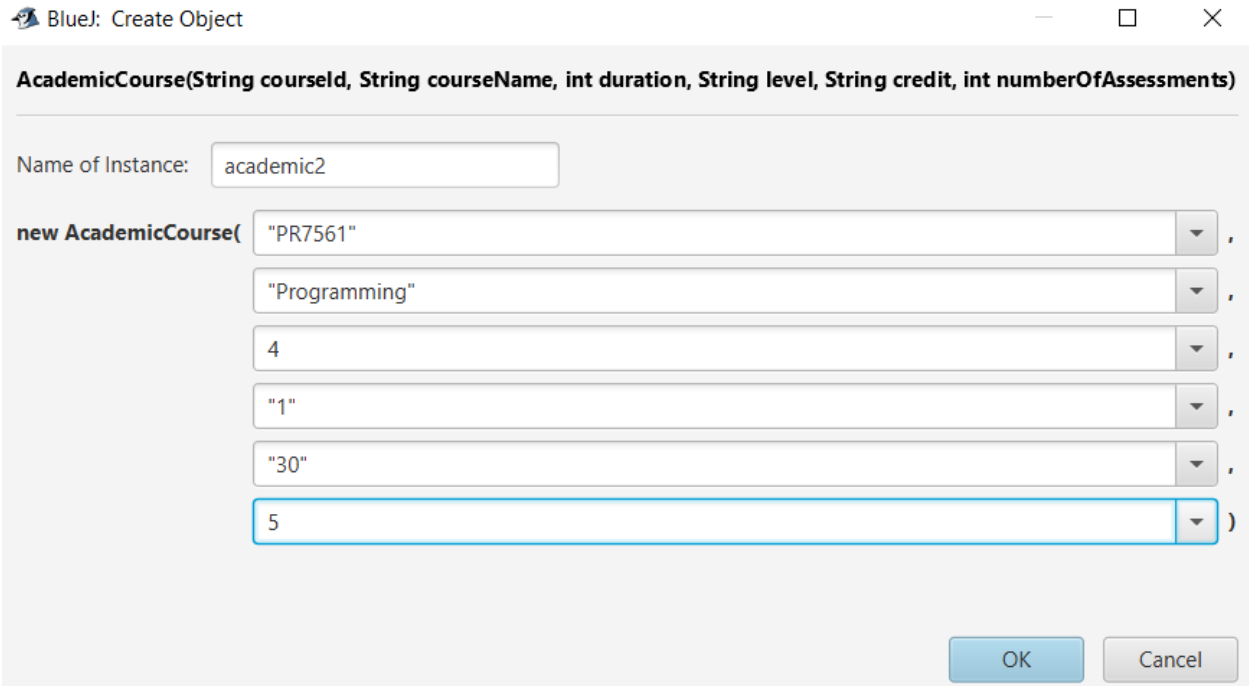

Figure 3: Figure of AcademicCourse

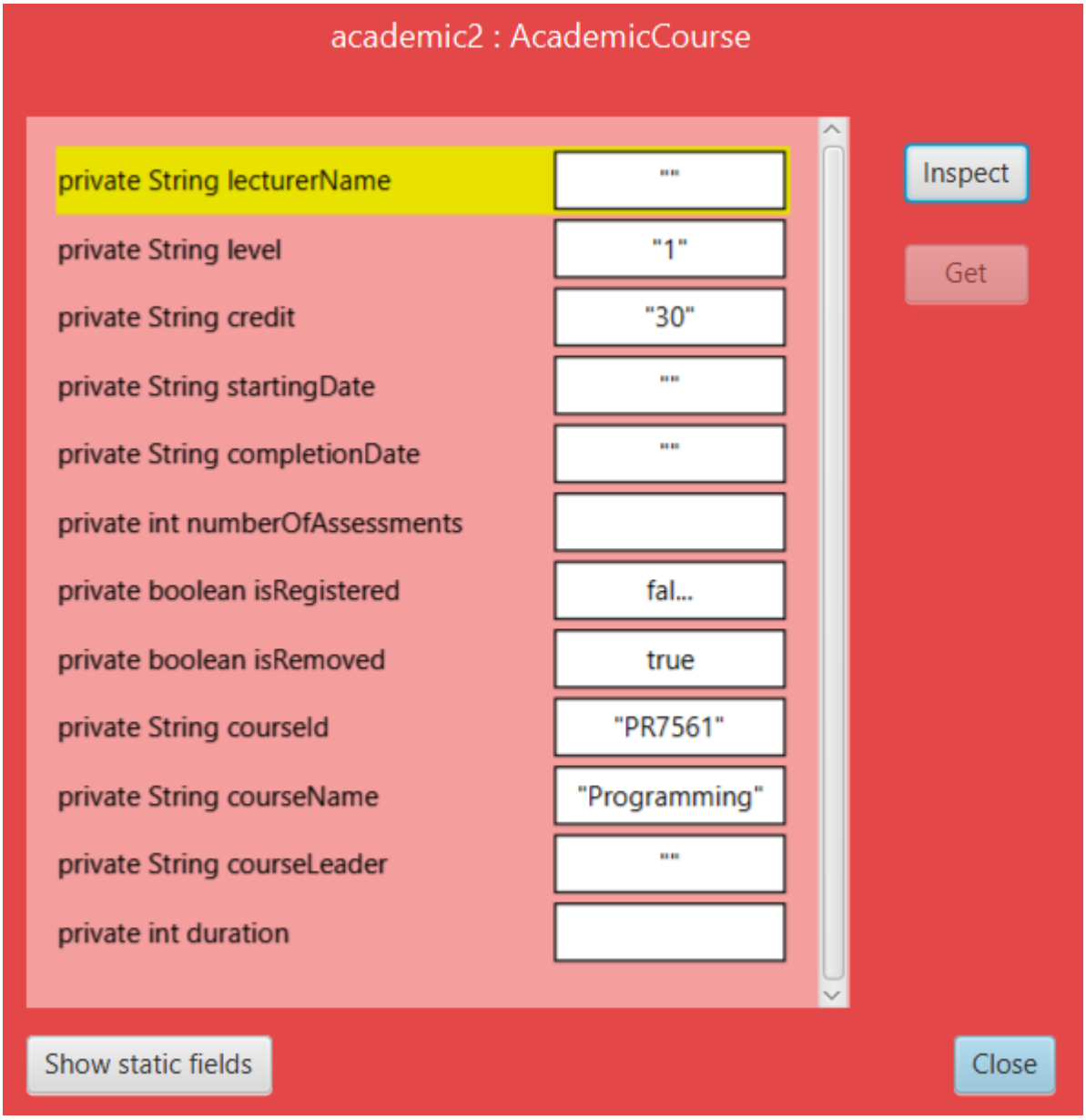

Figure 4: Figure of AcademicCourse class inserted value

Figure 5: Figure of method call of AcademicCourse class

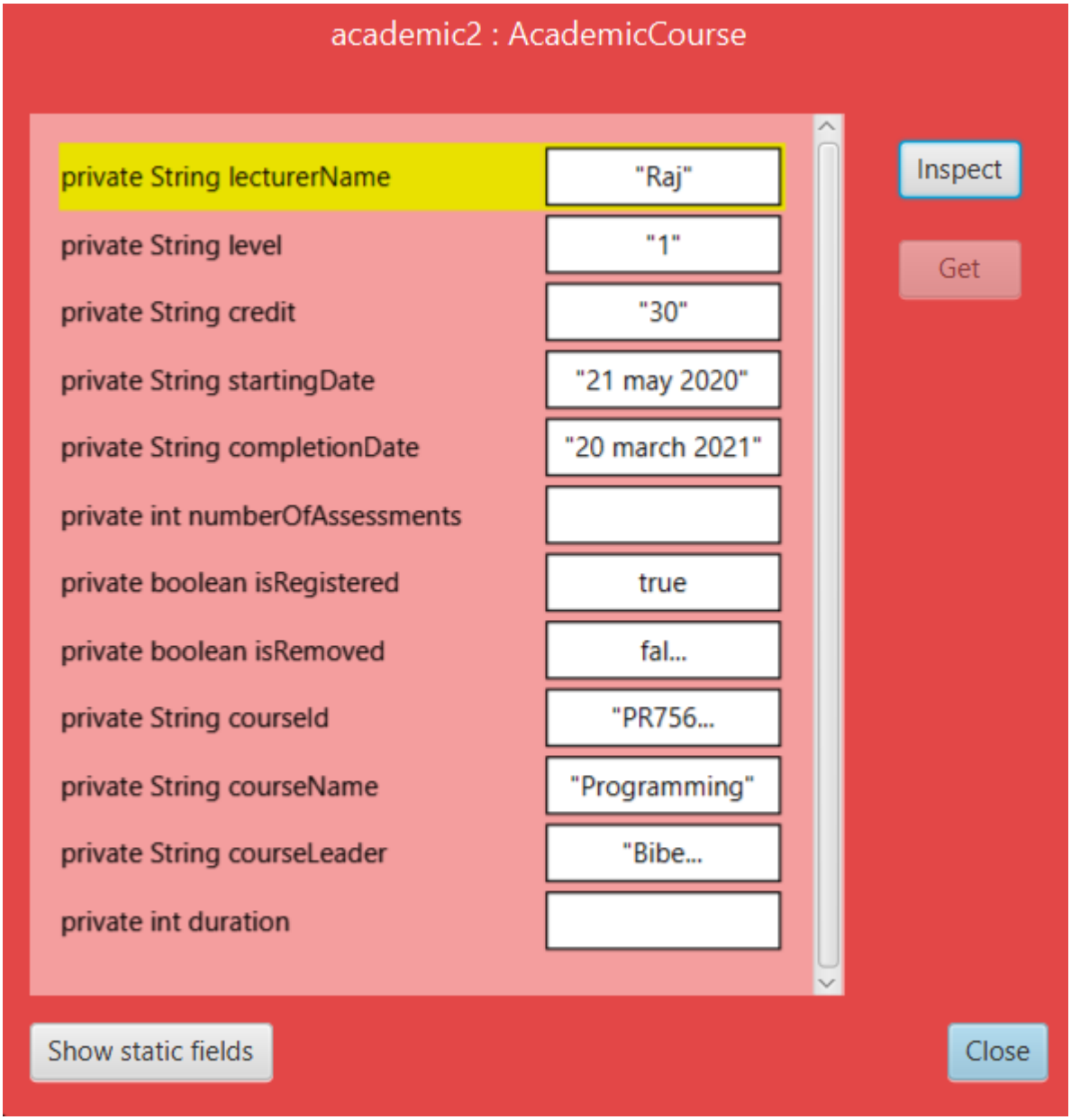

Figure 6: Figure of AcademicCourse class method

5.2. Test2:

Inspect NonAcademicCourse class, registor an nonacademic course and re-inspect the AcademicCourse class

Objective	To register nonacademic course
Action	The NonAcademicCourse is called with following arguments. course ID = “PH0051” courseName = “physics” duration = 4 prerequisite = “practical” Inspection of the NonAcademicCourse class. Void register NonAcademicCourse is called with the following arguments: courseLeader = “Susan” InstructorName = “Binam” startingDate = “23 April 2020” completionDate = “25 May 2021” ExamDate = “29 may 2021” Reinspection of the NonAcademicCourse class.
Expected Result	The course will be registered.
Actual Result	The course has been registered.
Conclusion	Test was successful

Table 5: Table of Test2

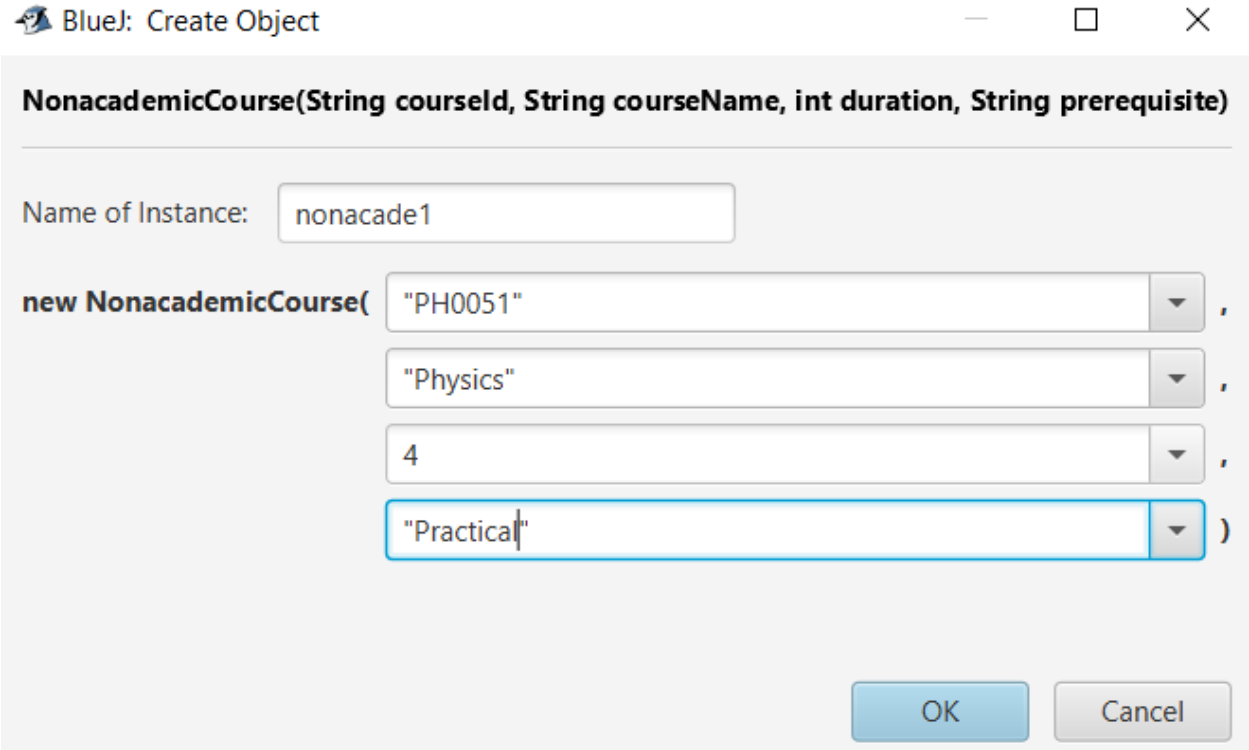

Figure 7: Figure of NonAcademicCourse inserting values

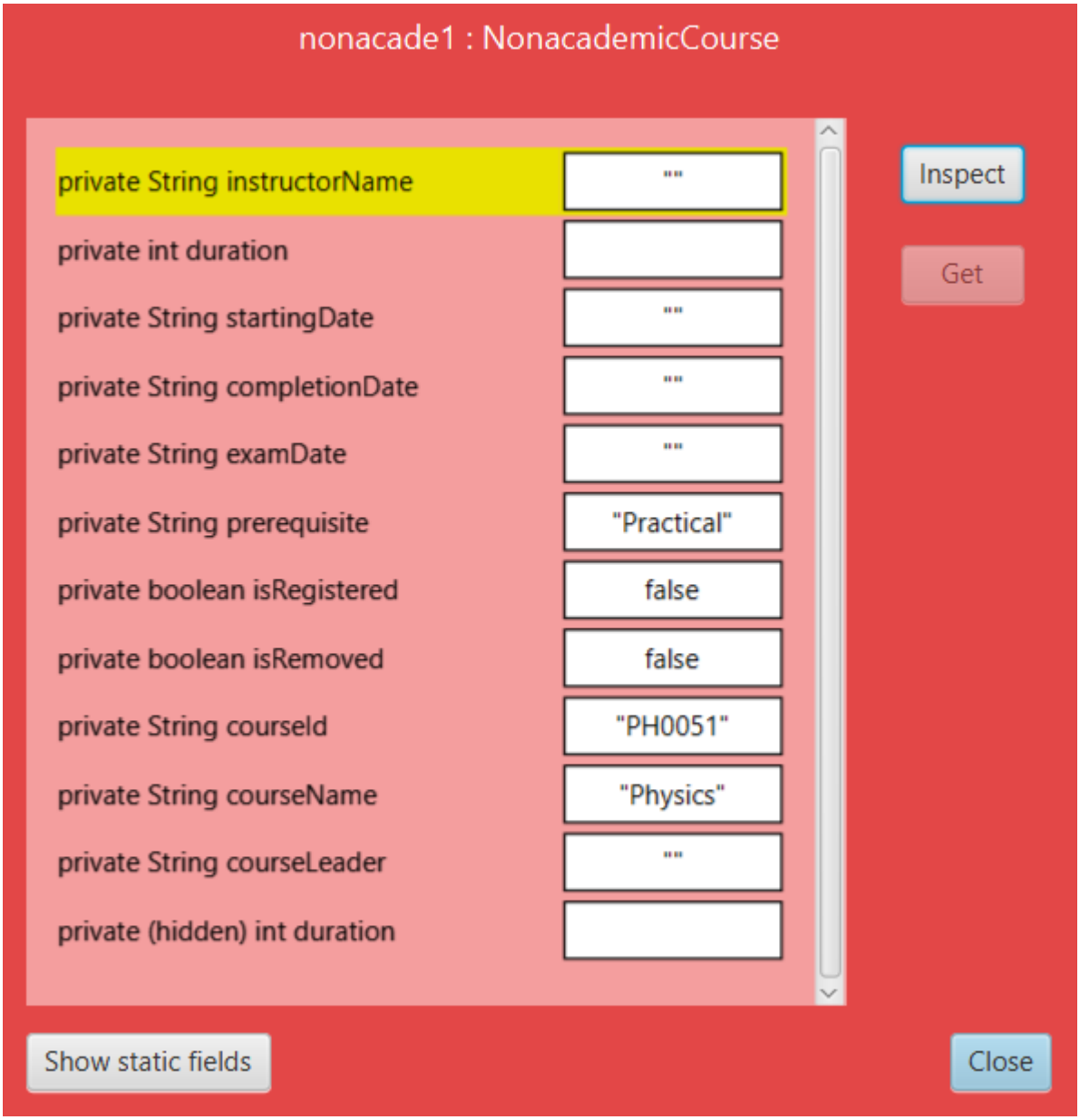

Figure 8: Figure of NonAcademicCourse Class inserted value

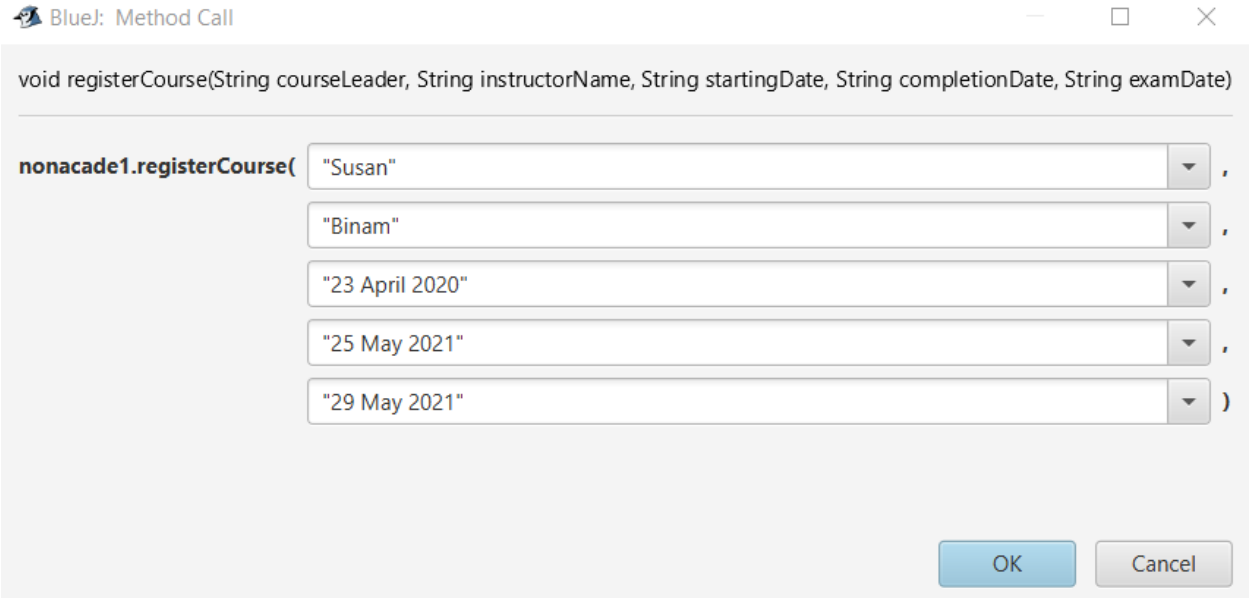

Figure 9: Figure of method call of NonAcademicCourse class

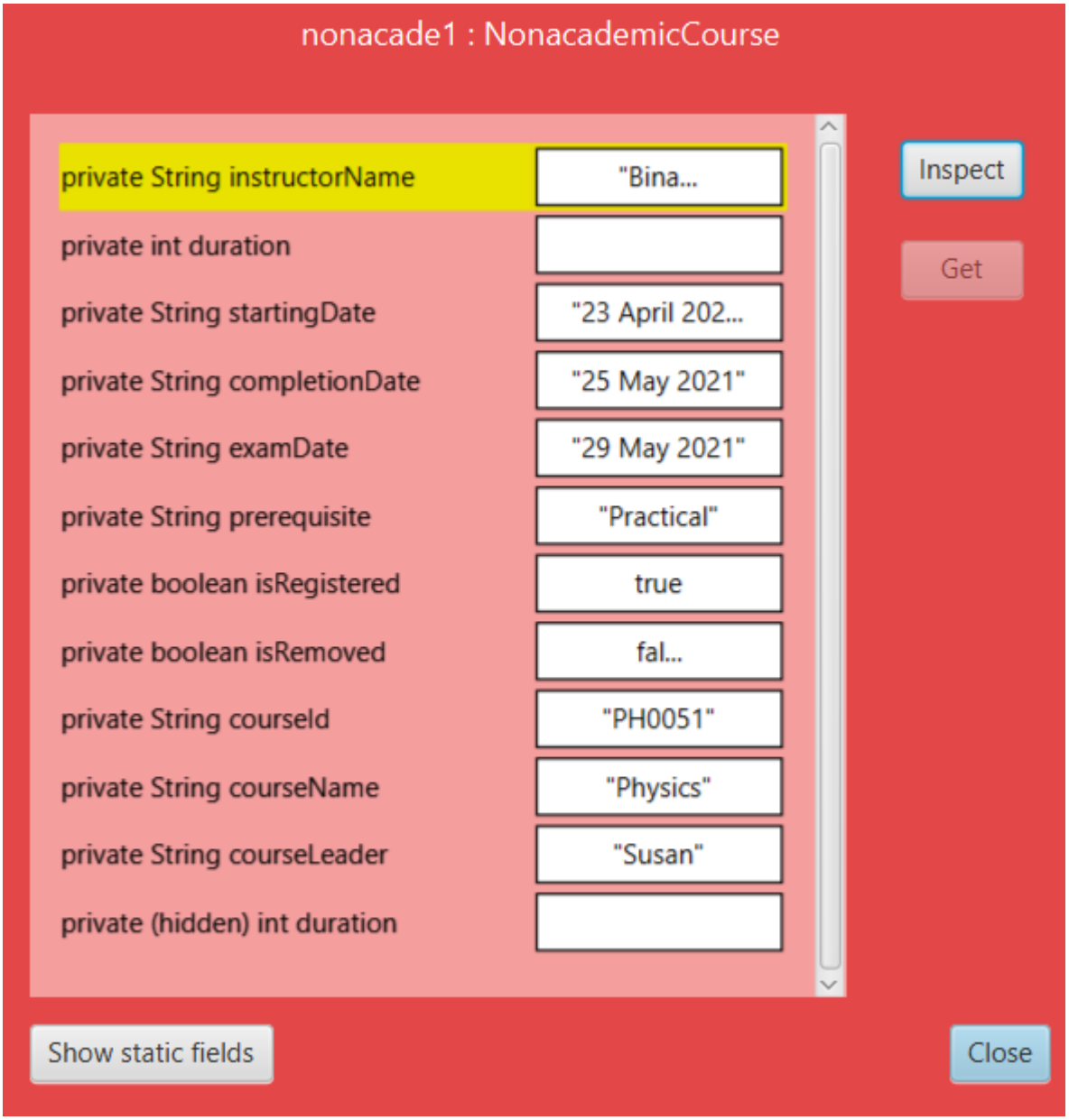

Figure 10: Figure of method call of academicCourse class inserted value

5.3. TEST3:

Inspect NonAcademicCourse class again, remove the non-academic course and re-inspect the NonAcademicCourse class.

Objective	Inspect NonAcademicCourse class, remove non-academic course and re-inspect NonAcademicCourse class.
Action	Inspection of the NonAcademicCourse class after test 2. b) Void remove is called. c) Re-inspection of the AcademicCourse class after the void remove is called.
Expected Result	The NonAcademicCourse will be removed.
Actual Result	The NonAcademicCourse has been removed.
Conclusion	Test was successful

Table 6: Table of test3

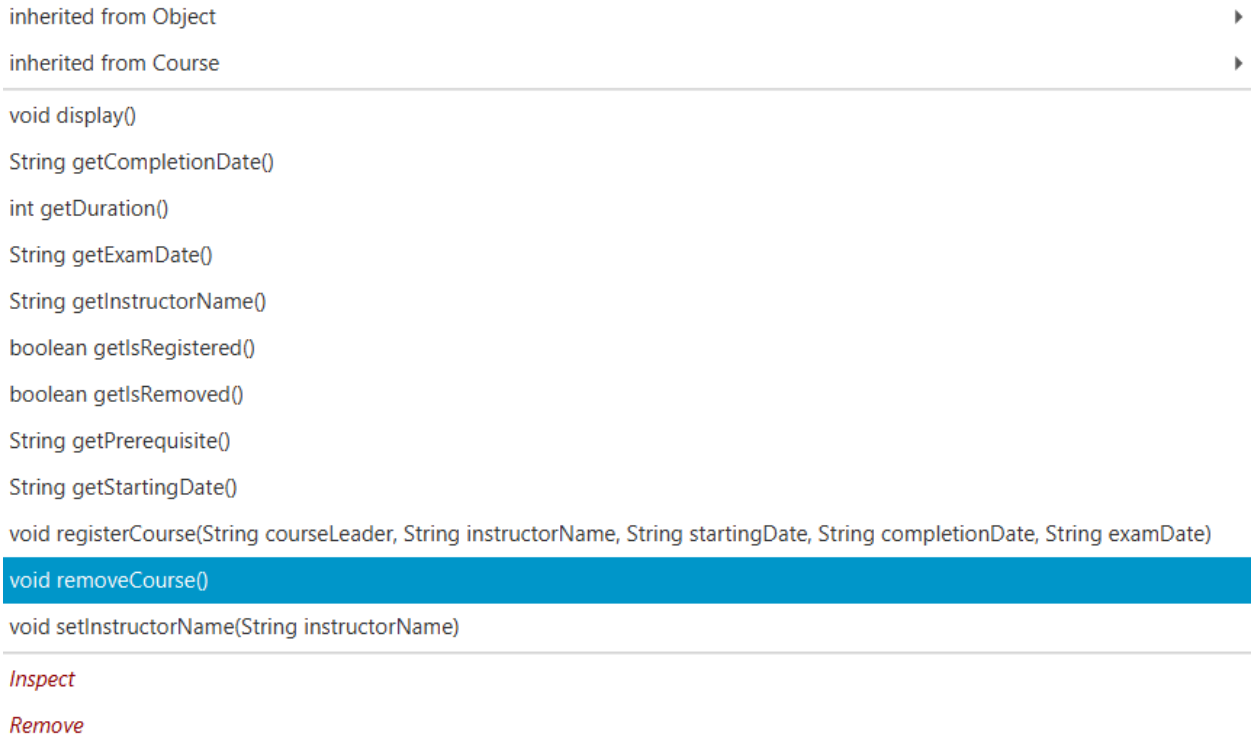

Figure 11: Figure of remove course method

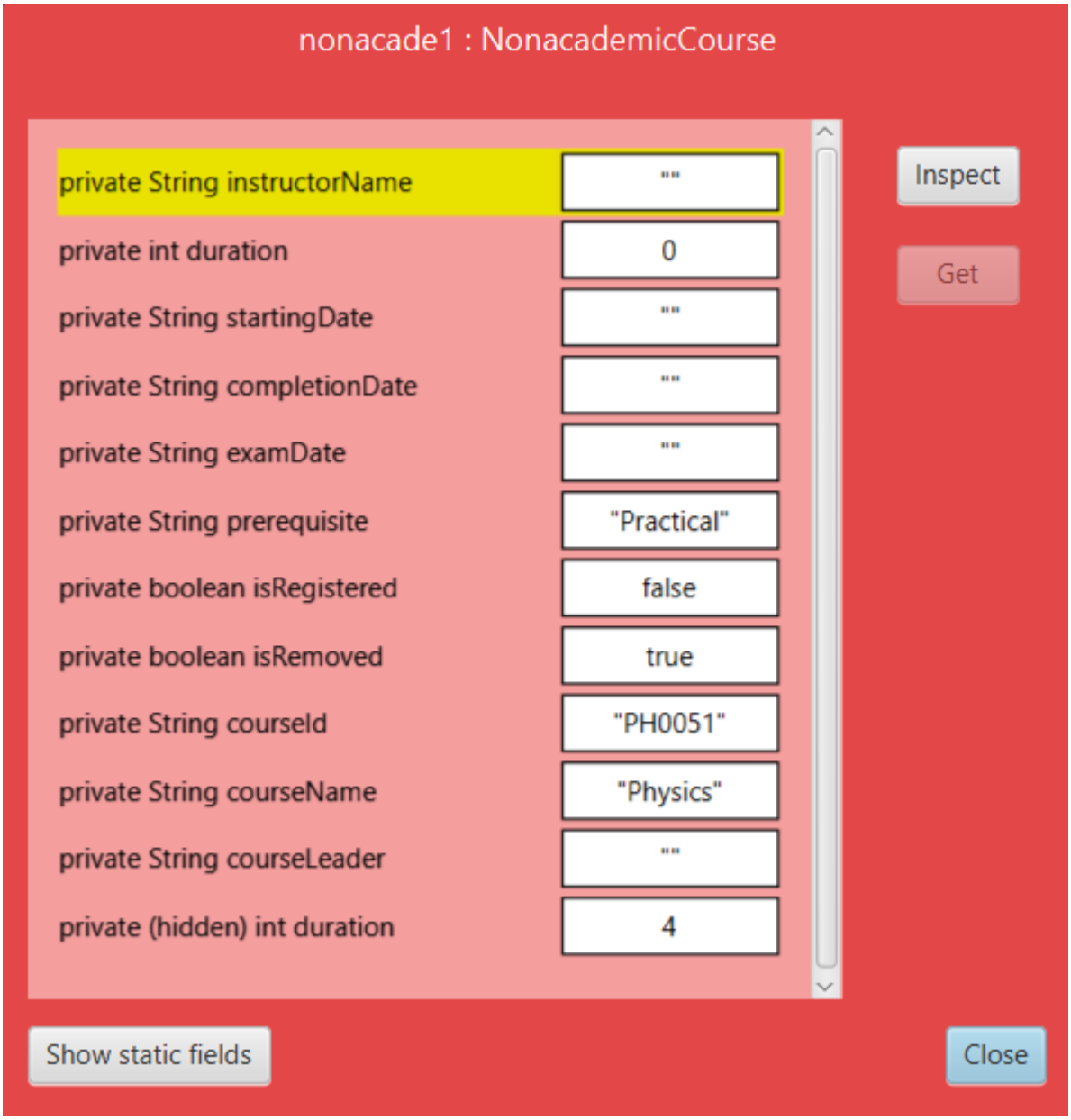

Figure 12: Figure of NonAcademicCourse method remove course

5.4. Test4:

To display the details of the AcademicCourse and NonAcademicCourse classes.

Objective	To display the details of the AcademicCourse and NonAcademicCourse classes.
Action	After performing test 1, void display method was called After performing test 2, void display method was called.
Expected Result	Display the details of AcademicCourse and NonAcademicCourse.
Actual Result	The details of AcademicCourse and NonAcademicCourse has been displayed.
Conclusion	Test was successful

Table 7: Table of test4

```
Course Details
CourseID: PR7561
CourseName: Programming
Duration: 4
Lecturer Name: Raj
Level: 1
Credit: 30
Starting Date: 21 may 2020
Completion Date: 20 march 2021
```

Figure 13: Figure of method display class academicCourse

```
Course Details
CourseID: PH0051
CourseName: Physics
Duration: 4
Instructor Name: Binam
Starting Date: 23 April 2020
Completion Date: 25 May 2021
Examination Date: 29 May 2021
```

Figure 14: Figure of method display of class nonAcademicCourse

6. Error Detection And Correction

During this assignment I went through a lot of mistakes. As the java is case sensitive language most of the time I faced compile time error. It is because of brackets, missing letters, missing colons, etc. Sometimes I get errors in calling methods. This is because of the values called in parameters.

In very first, BlueJ is the wonderful java compiler which allows user to find the errors in different lines. With the help of this editor my task has been easier to solve errors. On the other hand checking the run time error and inspection the methods errors were solved.

6.1. Semantic error

A semantic error occurs when a sentence is syntactically correct but does not perform the function intended by the programmer. This may often cause your program to crash, such as when dividing by zero: 1. 2.

Error:

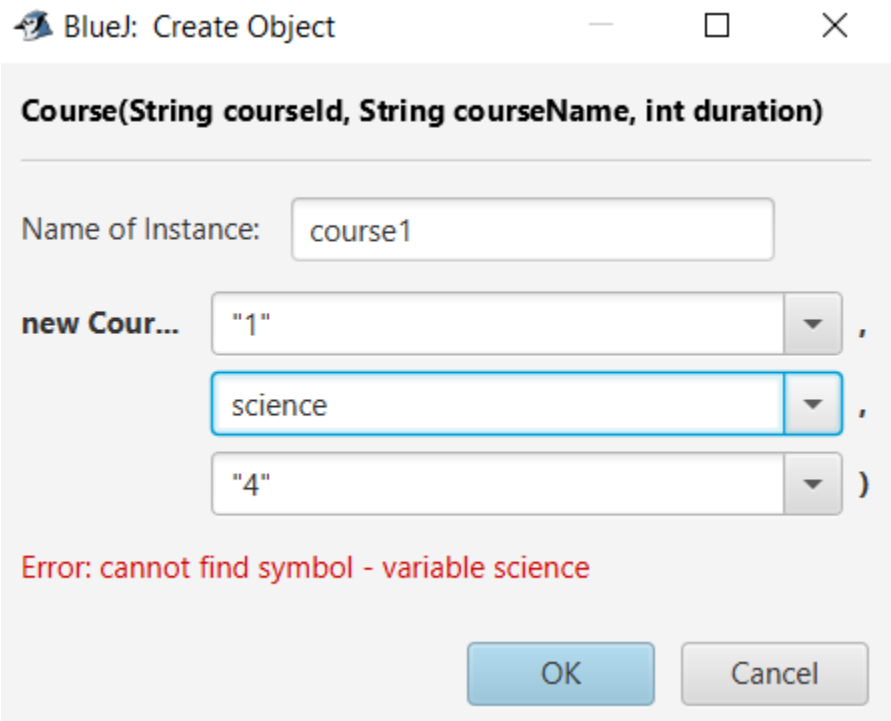

Figure 15: Figure of Semantic error

Correction:

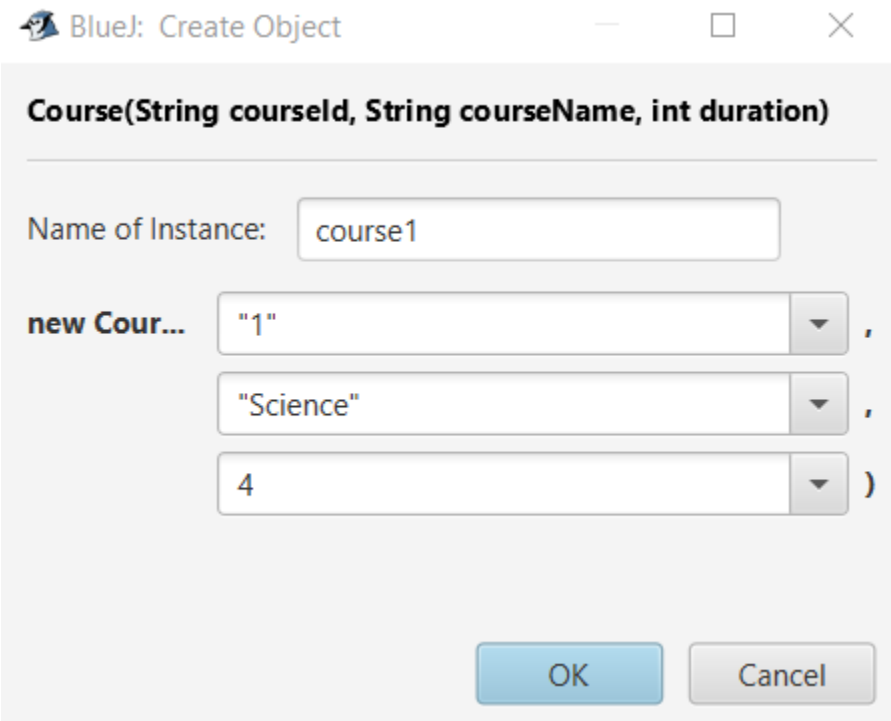

Figure 16: Correction of Semantic error

6.2. Syntax error

A syntax error is a mistake in a program's source code. Since computer programs must adhere to strict syntax in order to compile correctly, any parts of the code that do not follow the programming language's syntax will result in a syntax error.

Error:

```
public void setLeader(String courseLeader)
{
    this.courseLeader = courseLeader:
}
```

Figure 17: figure of syntax error

Correction:

```
public void setLeader(String courseLeader)
{
    this.courseLeader = courseLeader;
}
```

Figure 18: Figure of correction of syntax error

6.3. Logical error

A logic error (or logical error) is a mistake in the source code of a program that causes it to behave incorrectly or unexpectedly. It's a form of runtime error that can cause a program to crash or simply produce the incorrect performance. Logic errors can be caused by a variety of programming errors.

Error:

```
public void display()
{
    if(isRegistered == true); {
        super.display();
        System.out.println("Instructor Name: " + instructorName);
        System.out.println("Starting Date: " + startingDate);
        System.out.println("Completion Date: " + completionDate);
        System.out.println("Examination Date: " + examDate);
    }

    else {
        super.display();
    }
}
```

Correction:

```
public void display()
{
    if(isRegistered == true) {
        super.display();
        System.out.println("Instructor Name: " + instructorName);
        System.out.println("Starting Date: " + startingDate);
        System.out.println("Completion Date: " + completionDate);
        System.out.println("Examination Date: " + examDate);
    }

    else {
        super.display();
    }
}
```

7. Conclusion

Completing this assignment not only helps to understand the concept of inheritance, but also helps to understand the instance variables, constructor, methods, and many more. Lots of difficulties are there during this project. It helps to learn and minimize the errors in program in coming future. It also helps to assign the values, know the concept of parameters. Finally, with the lots of guidance and assistance from our teachers and friends, finally we are able to complete this assignment with the every suggested tasks in the question. Further in future experience from this project is going to be worthy.

8. References

Airth, M., 23 january, 2020. *Study.com.* [Online]
Available at: https://study.com/academy/lesson/pseudocode-definition-examples-quiz.html

Airth, M., n.d. *study.com.* [Online]
Available at: https://study.com/academy/lesson/pseudocode-definition-examples-quiz.html

Anon., april 19, 2012. *TechTerms.* [Online]
Available at: https://techterms.com/definition/java
[Accessed 2021].

guru99, 2019. *guru99.* [Online]
Available at: https://www.guru99.com/uml-class-diagram.html

helsinki, n.d. *mooc.* [Online]
Available at: https://java-programming.mooc.fi/

MCKenzie, C., april 2019. *TheServerSide.* [Online]
Available at: https://www.theserverside.com/definition/Java

9. Code

9.1. Course

```
public class Course
{
```

```
    //ivars
    private String courseId;
    private String courseName;
    private String courseLeader;
    private int duration;

    public Course(String courseId, String courseName, int duration)
    {
        this.courseId = courseId;
        this.courseName = courseName;
        this.duration = duration;
        this.courseLeader = "";
    }

    //Accessors Method
    public String getCourseId() //Allow user to access private courseId
    {
        if(this.courseId.equals("")){
        return "is empty";
        }
        return courseId;
    }

    public String getCourseName()   //Allow user to access private courseName
    {
        return courseName;
    }

    public int getDuration()    //Allow user to access private duration
    {
```

```
        return duration;
    }

    public String getLeader()   //Allow user to access private courseLeader
    {
        return courseLeader;
    }

    //Mutators Method
    public void setLeader(String courseLeader)//Allows user to set the value of
courseLeader
    {
        this.courseLeader = courseLeader;
    }

    public void display()//Displays all
    {
        System.out.println("Course Details");
        System.out.println("CourseID: " + courseId);
        System.out.println("CourseName: " + courseName);

        if(courseLeader.isEmpty()) {
            System.out.println("CourseLeader: " + courseLeader);
        }

        System.out.println("Duration: " + this.duration);
    }

}
```

9.2. AcademicCourse

```
public class AcademicCourse extends Course
{
    //ivars
    private String lecturerName;
    private String level;
    private String credit;
    private String startingDate;
    private String completionDate;
    private int numberOfAssessments;
    private boolean isRegistered, isRemoved;

    public AcademicCourse(String courseId, String courseName, int duration, String
level, String credit, int numberOfAssessments)
    {
        super(courseId, courseName, duration);

        this.level = level;
        this.credit = credit;
        this.numberOfAssessments = numberOfAssessments;
        lecturerName = "";
        startingDate = "";
        completionDate = "";
        isRegistered = false;
```

```
        isRemoved = true;
    }

    //Accessors Method
    public String getLecturerName() //Allow us to access the value of private instant
variable lecturerName
    {
        return lecturerName;
    }

    public String getLevel()    //Allow us to access the value of private instant variable
level
    {
        return level;
    }

    public String getCredit()   //Allow us to access the value of private instant variable
credit
    {
        return credit;
    }

    public String getStartingDate() //Allow us to access the value of private instant
variable startingDate
    {
```

```
        return startingDate;
    }

    public String getCompletionDate()   //Allow us to access the value of private
instant variable completionDate
    {
        return completionDate;
    }

    public int getNumberOfAssessments() //Allow us to access the value of private
instant variable numberOfAssessments
    {
        return numberOfAssessments;
    }

    public boolean getIsRegistered()    //Allow us to access the value of private instant
variable isRegistered
    {
        return isRegistered;
    }

    public boolean getIsRemoved()   //Allow us to extract the value of private instant
variable isRemoved
    {
        return isRemoved;
```

```
    }

    //Mutators Method
    public void setLecturerName(String lecturerName)//allows us to set the value of
instant variable lecturerName
    {
        this.lecturerName = lecturerName;
    }

    public void setNumberOfAssessments(int numberOfAssessments)//allows us to
set the value of instant variable numbetOfAssesments
    {
        this.numberOfAssessments = numberOfAssessments;
    }

    public void registerCourse(String courseLeader, String lecturerName, String
startingDate, String completionDate)
    {
        if(isRegistered == false) {
            super.setLeader(courseLeader);
            this.lecturerName = lecturerName;
            this.startingDate = startingDate;
            this.completionDate = completionDate;
            isRegistered = true;
            isRemoved = false;
```

```
        }


        else {
            System.out.println("This course is being registerd to " +lecturerName + " to
    complete within " + startingDate + " to " + completionDate);
        }
    }


    public void display()
    {
        if(isRegistered == true) {
            super.display();
            System.out.println("Lecturer Name: " + lecturerName);
            System.out.println("Level: " + level);
            System.out.println("Credit: " + credit);
            System.out.println("Starting Date: " + startingDate);
            System.out.println("Completion Date: " + completionDate);
        }


        else {
            super.display();
        }
    }
```

```
    }
```

9.3. NonAcademicCourse

```
public class NonacademicCourse extends Course
{
    private String instructorName;
    private int duration;
    private String startingDate;
    private String completionDate;
    private String examDate;
    private String prerequisite;
    private boolean isRegistered, isRemoved;

    public NonacademicCourse(String courseId, String courseName, int duration, String
prerequisite)
    {
        super(courseId,courseName,duration);

        this.prerequisite = prerequisite;
        instructorName = "";
        startingDate = "";
        completionDate = "";
        examDate = "";
        isRegistered = false;
        isRemoved = false;
    }

    /*Accessors(getter) Method
```

```
      Allows user to access all the instant variables

      */

     public String getInstructorName()   //Allow us to access the value of private instant
variable instructorName

    {

        return instructorName;

    }


    public int getDuration()    //Allow us to access the value of private instant variable
duration

    {

        return duration;

    }


    public String getStartingDate()  //Allow us to access the value of private instant
variable startingDate

    {

        return startingDate;

    }


    public String getCompletionDate()   //Allow us to access the value of private instant
variable completionDate

    {

        return completionDate;

    }


    public String getExamDate() //Allow us to access the value of private instant variable
examDate

    {

        return examDate;

    }
```

```
    public String getPrerequisite() //Allow us to access the value of private instant
variable prerequisite
    {
        return prerequisite;
    }

    public boolean getIsRegistered()//Allow us to access the value of private instant
variable isRegistered
    {
        return isRegistered;
    }

    public boolean getIsRemoved()   //Allow us to access the value of private instant
variable isRemoved
    {
        return isRemoved;
    }

    /*Mutators(setter) Method
       it sets the value of the instant variable
       */
    public void setInstructorName(String instructorName)//Allow us to set the value of
instant variable instrucorName
    {
        if(isRegistered == false) {
            this.instructorName = instructorName;
        }

        else {
            System.out.println("This course is registered therefore Instructor Name can't be
updated");
        }
```

```
    }

    public void registerCourse(String courseLeader, String instructorName, String
startingDate, String completionDate, String examDate)
    {
        if(isRegistered == false) {
            super.setLeader(courseLeader);

            setInstructorName(instructorName);
            this.startingDate = startingDate;
            this.completionDate = completionDate;
            this.examDate = examDate;
            isRegistered = true;
        }

        else {
            System.out.println("This course has been Registered!");
        }
    }

    public void removeCourse()
    {
        if(isRemoved == true) {
            System.out.println("This course has been removed!");
        }

        else {
            super.setLeader("");

            instructorName = "";
```

```
            startingDate = "";
            completionDate = "";
            examDate = "";
            isRegistered = false;
            isRemoved = true;
        }
    }

    public void display()
    {
        if(isRegistered == true) {
            super.display();
            System.out.println("Instructor Name: " + instructorName);
            System.out.println("Starting Date: " + startingDate);
            System.out.println("Completion Date: " + completionDate);
            System.out.println("Examination Date: " + examDate);
        }

        else {
            super.display();
        }
    }
}
```