Module Code & Module Title

CS4001NT Programming

Assessment Weightage & Type

30 % Individual Coursework -1

Year and Semester

2020-21 Spring, Semester

London Met ID: 20048528

Student Name: Sumiran Dahal

College ID:NP05CP4S210117

Assignment Submission Date:23/05/2021

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 submissions will be treated as non-submission and a mark of zero will be awarded.

Table of Contents

Table of Figures:

List of Tables:

1. Introduction:

Java is a most popular programming language. It was released in 1995 by Sun Microsystems. It is created by James Gosling. Java programming language is still popular, widely used. Java has many applications, including software development, mobile applications, large systems can be developed from Java programming language. Java is portable, object-oriented programming language it can run in any platform because of bytecode. Bytecode is executed by JVM. This is a mini project of JAVA programming language. As we all know, Java is an object-oriented programming language. So, this project is ultimately related to it. In this, project we are told to create a project using JAVA's characteristics.

This project main objectives are to get over the core concept of programming language. Playing with methods, creating objects, method overloading or polymorphism, inheritance, method overriding, creating constructors and encapsulation are the main objectives of this project. The main advantage of OOP is reusability of code.

The project is about the concept of inheritance, data encapsulation, polymorphism, method overloading etc. which are required to do this project. For doing, BlueJ will be used. It is development environment which allows user to develop Java programs quickly and easily. For the project some screenshots are attached to clarify the work. So, Microsoft's Snipping tool will be used for snapping screenshots. The project also describes class diagram because, it can model software at a high level of abstraction using class diagrams without having to look at the source code. Draw.io is used for drawing diagrams.

Pseudocode will be provided for each method in each class. A short description of each method also will be implemented in the project. And at last the project will be tested because every project needs to be tested before using it. The project will be inspected with BlueJ and verified it's working condition. The project also contains section of error and correction of code.

2. Class Diagram:

A class diagram is a diagram used to define classes and their relationships in software design and modeling. We can model software at a high level of abstraction using class diagrams without having to look at the source code. A class diagram's classes refer to the source code's classes. The diagram shows the names and attributes of the classes, as well as the relations between them.

Class diagram for this project are as follows:

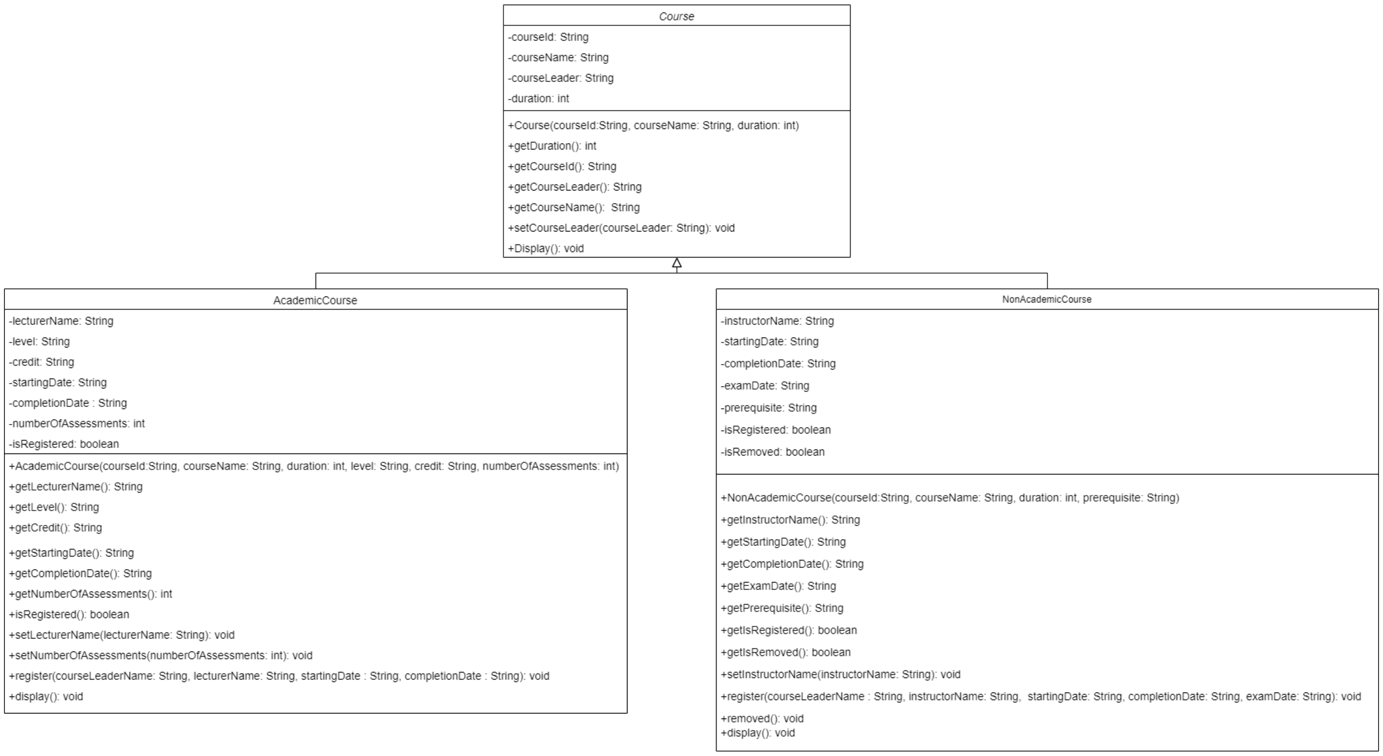

Course
-courseId: String
-courseName: String
-courseLeader: String
-duration: int
+Course(courseId:String, courseName: String, duration: int)
+getDuration(): int
+getCourseId(): String
+getCourseLeader(): String
+getCourseName(): String
+setCourseLeader(courseLeader: String): void
+Display(): void
AcademicCourse
-lecturerName: String
-level: String
-credit: String
-startingDate: String
-completionDate : String
-numberOfAssessments: int
-isRegistered: boolean
+AcademicCourse(courseId:String, courseName: String, duration: int, level: String, credit: String, numberOfAssessments: int)
+getLecturerName(): String
+getLevel(): String
+getCredit(): String
+getStartingDate(): String
+getCompletionDate(): String
+getNumberOfAssessments(): int
+isRegistered(): boolean
+setLecturerName(lecturerName: String): void
+setNumberOfAssessments(numberOfAssessments: int): void
+register(courseLeaderName: String, lecturerName: String, startingDate : String, completionDate : String): void
+display(): void
NonAcademicCourse
-instructorName: String
-startingDate: String
-completionDate: String
-examDate: String
-prerequisite: String
-isRegistered: boolean
-isRemoved: boolean
+NonAcademicCourse(courseId:String, courseName: String, duration: int, prerequisite: String)
+getInstructorName(): String
+getStartingDate(): String
+getCompletionDate(): String
+getExamDate(): String
+getPrerequisite(): String
+getIsRegistered(): boolean
+getIsRemoved(): boolean
+setInstructorName(instructorName: String): void
+register(courseLeaderName : String, instructorName: String, startingDate: String, completionDate: String, examDate: String): void
+removed(): void
+display(): void

Figure 1: Class Diagram

2.1. Describing Class and Class Attribute, Constructors and Methods:

A class is represented in a class diagram by a rectangle with the class name(Course, AcademicCourse, and NonAcademicCourse) is written on top. The name of the class is separated from the list of attributes by a line below the name (names and types of the class variables). The line below of the attribute table are listed(constructors and methods). Each attribute is written on a separate line. In a class diagram, we list constructor and all other methods below attributes. A line below attributes separates methods in class diagram. The private access modifiers are denoted by (-) sign which is presented in the instance variables of each class. The constructors and methods have public access modifiers so it is denoted by (+) sign. In some methods and constructors there are parameters inside parentheses. AcademicCourse and NonAcademicCourse are the child class of Course. The data type of each variable is presented at right side of each of them. There is no return type of constructors but methods do have. The return type of each method is also presented at right side of each of them.

3. Pseudocode:

Pseudocode is the simple form of writing code in English which can be easily understandable. Pseudocode is not a programming language. It is a methodology that allows the programmer to represent the implementation of an algorithm. It uses short phrases to write code for programs before one's actually created it in a specific language. Pseudocode is very useful for large project because, we don't need to see the whole code if we have pseudocode. Pseudocode is the description of the steps of an algorithm or a program.

The pseudocode of this project will be referenced below:

3.1. Pseudocode for Class Course:

3.1.1. Function getDuration:

```
FUNCTION getDuration():
        RETURN duration
END
```

3.1.2. Function getCourseId:

```
FUNCTION getCourseId():
        RETURN courseId
END
```

3.1.3. Function getCourseLeader:

```
FUNCTION getCourseLeader():

        RETURN courseLeader

END
```

3.1.4. Function getCourseName:

```
FUNCTION getCourseName():

        RETURN courseName

END
```

3.1.5. Function setCourseLeader:

```
FUNCTION setCourseLeader(courseLeader):

        UPDATE courseLeader

END
```

3.1.6. Function display:

```
FUNCTION display():

        PRINT "CourseID:" +courseId

        PRINT "Course Name:" +courseName

        PRINT "Duration in minute:" +duration

        IF courseLeader is not equal to empty string then,

                PRINT "Course Leader's Name:" +courseLeader

        END IF

END
```

3.2. Pseudocode for AcademicCourse Class:

3.2.1. Function getLecturerName:

```
FUNCTION getLecturerName():

        RETURN lecturerName

END
```

3.2.2. Function getLevel:

```
FUNCTION getLevel():

        RETURN level

END
```

3.2.3. Function getCredit:

```
FUNCTION getCredit():

        RETURN credit

END
```

3.2.4. Function getStartingDate:

```
FUNCTION getStartingDate():

        RETURN startingDate

END
```

3.2.5. Function getCompletionDate:

```
FUNCTION getCompletionDate():

    RETURN completionDate

END
```

3.2.6. Function getNumberOfAssessments:

```
FUNCTION getNumberOfAssessments():

    RETURN numberOfAssessments

END
```

3.2.7. Function isRegistered:

```
FUNCTION isRegistered():

    RETURN isRegistered

END
```

3.2.8. Function setLecturerName:

```
FUNCTION setLecturerName(lecturerName)

    UPDATE lecturerName

END
```

3.2.9. Function register:

```
FUNCTION register(courseLeaderName, lecturerName, startingDate, completionDate):

    IF isRegistered is false then,

        PRINT "Lecturer's Name is:" +lecturerName

        PRINT "Starting Date is:" +startingDate

        PRINT "Completion Date is:" +completionDate

        RETURN

    END IF

        INVOKE setCourseLeader(courseLeaderName) from parent class

        UPDATE lecturerName

        UPDATE startingDate

        UPDATE completionDate

        UPDATE isRegistered to true

END
```

3.2.10. Function display:

```
FUNCTION display()

    INVOKE display() from parent class

    IF isRegistered is true then,

        PRINT "Lecturer's Name is:" +lecturerName

        PRINT "Level is:" +level

        PRINT "Credit is:" +credit
```

```
            PRINT "Starting Date is:" +startingDate
            PRINT "Completion Date is:" +completionDate
            PRINT "Number of assessments:" +numberOfAssessments
        END IF
    END
```

3.3. Pseudocode for NonAcademicClass:

3.3.1. Function getInstructorName

```
FUNCTION getInstructorName():
    RETURN instructorName
END
```

3.3.2. Function getStartingDate

```
FUNCTION getStartingDate():
    RETURN startingDate
END
```

3.3.3. Function getCompletionDate

```
FUNCTION getCompletionDate():
    RETURN completionDate
END
```

3.3.4. Function getExamDate:

```
FUNCTION getExamDate():

    RETURN examDate

END
```

3.3.5. Function getPrerequisite:

```
FUNCTION getPrerequisite():

    RETURN prerequisite

END
```

3.3.6. Function isRegistered:

```
FUNCTION isRegistered()

    RETURN isRegistered

END
```

3.3.7. Function isRemoved:

```
FUNCTION isRemoved()

    RETURN isRemoved

END
```

3.3.8. Function setInstructorName

```
FUNCTION setInstructorName(instructorName)
    IF isRegistered is false then,
            UPDATE instructorName
    ELSE
            PRINT "The Nonacademic course is already regstered. Unable to set Instrutor's Name"
    END IF
END
```

3.3.9. Function register

```
    FUNCTION register(courseLeaderName, instructorName, startingDate, completionDate, examDate):
            IF isRegistered is false then,
                    INVOKE setInstructorName(instructorName)
                    INVOKE setCourseLeader(courseLeaderName) from parent class
                    UPDATE startingDate
                    UPDATE completionDate
                    UPDATE examDate
                    UPDATE isRegistered to true
            ELSE
                PRINT "The Nonacademic course is already registered. Unable to register again"
            END IF
    END
```

3.3.10. Function remove:

```
FUNCTION remove():
    IF isRemoved is false then,
        INVOKE setCourseLeader to an empty string
        UPDATE instructorName to an empty string
        UPDATE startingDate to an empty string
        UPDATE completionDate to an empty string
        UPDATE examDate to an empty string
        UPDATE isRegistered to false
        UPDATE isRemoved to true
    ELSE
        PRINT "The Nonacademic course has been already removed"
    END IF
END
```

3.3.11. Function display:

```
    INVOKE display() from parent class
        IF isRegistered is true then,
            PRINT "Instructor's name is:" +instructorName
            PRINT "Starting Date is:" +startingDate
            PRINT "Completion Date is:" +completionDate
            PRINT "Exam Date is:" +examDate
        END IF
    END
```

4. Description of each method

4.1. Course Class

4.1.1. getDuration():

This is the getter method of Course class which returns attribute duration value of the calling instance.

4.1.2. getCourseId():

This is the getter method of Course class which returns attribute courseId value of the calling instance.

4.1.3. getCourseLeader():

This is the getter method of Course class which returns attribute courseLeader value of the calling instance.

4.1.4. getCourseName():

This is the getter method of Course class which returns attribute courseName value of the calling instance.

4.1.5. setCourseLeader():

This is the setter method of Course class which takes parameter as courseLeader and assigns or update its attribute courseLeader.

4.1.6. Display():

This is the display method of the Course class which display courseId, courseName and duration. There is a condition if the courseLeader is not an empty string then, then the name of courseLeader is displayed.

4.2. AcademicCourse Class

4.2.1. getLecturerName():

This is the getter method which returns attribute lecturerName value of the calling instance.

4.2.2. getLevel():

This is the getter method which returns attribute level value of the calling instance.

4.2.3. getCredit():

This is the getter method which returns attribute credit value of the calling instance.

4.2.4. getStartingDate():

This is the getter method which returns attribute startingDate value of the calling instance.

4.2.5. getCompletionDate():

This is the getter method which returns attribute completionDate value of the calling instance.

4.2.6. getNumberOfAssessments():

This is the getter method which returns attribute numberOfAssessments value of the calling instance.

4.2.7. isRegistered():

This is also the getter method which returns attribute isRegistered value of the calling instance.

4.2.8. setLecturerName():

This is the setter method which takes parameter as lecturerName and assigns or update its attribute lecturerName.

4.2.9. setNumberOfAssessments():

This is also the setter method which takes parameter as numberOfAssessments and assigns or update its attribute numberOfAssessments.

4.2.10. register():

This is the register method of AcademicCourse class which has parameters: courseLeaderName, lecturerName, startingDate and completionDate. There is a condition, if isRegistered is true then, it displays lecturer name, starting date and completion date and return there. And, the setCourseLeader method with parameter courseLeaderName is called form parent class. And, the lecturerName, startingDate and completionDate are updated with the values and isRegistered is changed to true.

4.2.11. display():

This is the display method inside AcademicCourse class. It has the same values of the as the display method in parent class because it is invoked from parent class. And it checks the condition if isRegistered is true then, it displays lecturer name, level, credit, starting date, completion date and number of assessments.

4.3. NonAcademicCourse Class:

4.3.1. getStartingDate():

This is the getter method which returns attribute startingDate value of the calling instance.

4.3.2. getCompletionDate():

This is the getter method which returns attribute completionDate value of the calling instance.

4.3.3. getExamDate():

This is the getter method which returns attribute examDate value of the calling instance.

4.3.4. getPrerequisite():

This is the getter method which returns attribute prerequisite value of the calling instance.

4.3.5. isRegistered():

This is the getter method which returns attribute isRegistered value of the calling instance.

4.3.6. isRemoved():

This is the getter method which returns attribute isRemoved value of the calling instance.

4.3.7. setInstructorName():

This is the setter method which takes instructorName as parameter. There is a condition inside this method which is if isRegistered is false then, the instructorName will be updated Else the message is displayed.

4.3.8. register():

This is the register method which has courseLeaderName, instructorName, startingDate, completionDate, examDate as parameters. If isRegistered is false then, setter method setInstructorName with parameter instructorName is invoked, setter method setCourseLeader with parameter(courseLeaderName) from parent class is invoked too. startingDate, completionDate and examDate are updated. IsRegistered is changed to true. Else the message is displayed.

4.3.9. remove():

This is the remove method. If isRemoved is false then, setCourseLeader method with empty string parameter is called from parent class. instructorName, startingDate, completionDate, examDate are updated to empty string. isRegistered is updated to false and isRemoved is updated to true. Else the message is displayed.

4.3.10. display():

This is the display method. display method is called from super class. If isRegistered is true then, the instructor name, starting date, completion date and exam date is displayed.

5. Testing/Inspection:

The code will be inspected using BlueJ. For the evidence, screenshots will be attached in the project.

5.1. Test 1:Inspect AcademicCourse class, register an academic course, and re-inspect the AcademicCourse Class

- **Testing Table 1:**

Objective	Inspect AcademicCourse class, register an academic course and re-inspect AcademicCourse

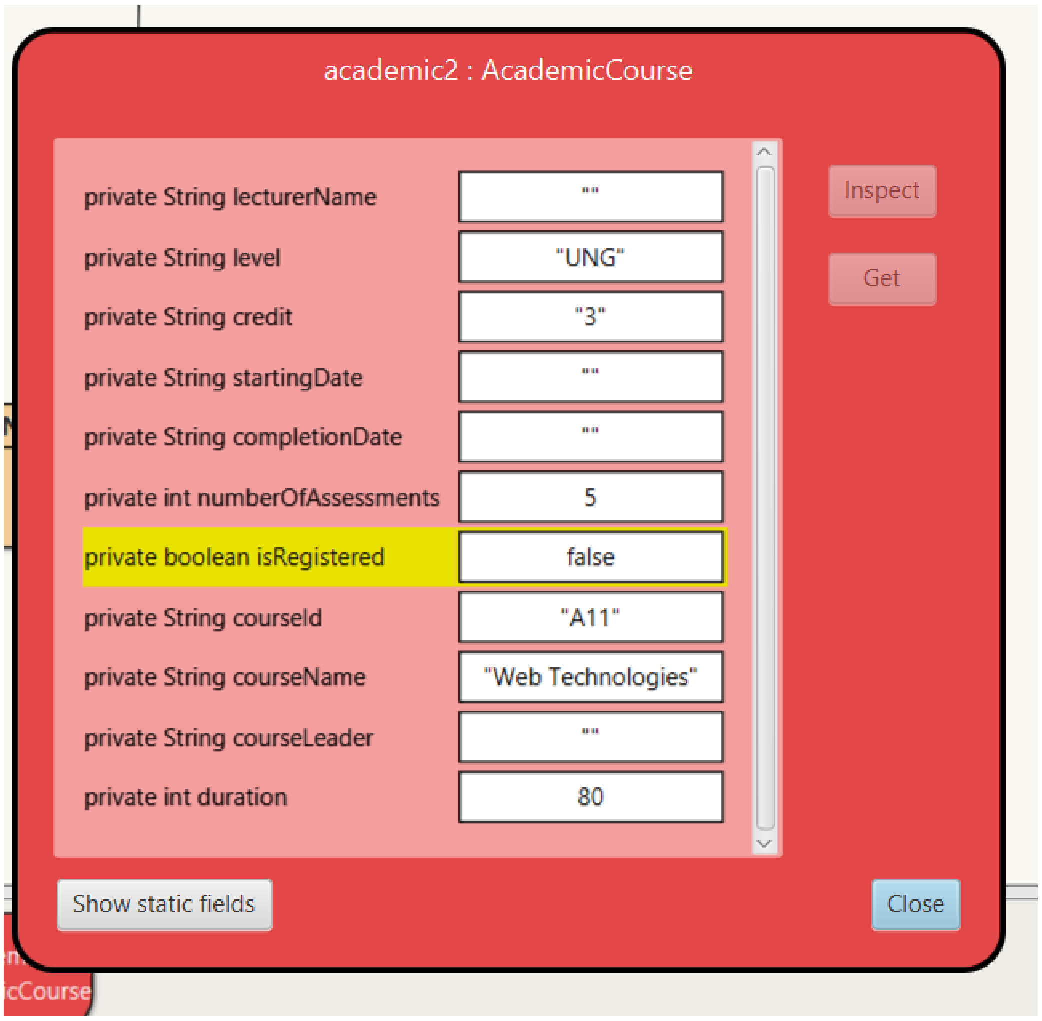
Action	Call Constructor of AcademicCourse class.
	courseId=”A11”
	courseName= “Web Technologies”
	duration=80
	level=”UNG”
	credit=”3”
	numberOfAssessments=5
	
	Calling method to register AcademicCourse
	courseLeaderName=”Sailesh_Dahal”
	lecturerName=”Prabesh_KC”
	startingDate=”2021-06-15”
	completionDate=”2021-09-15”

Expected Result	AcademicCourse should be registered.
Actual Result	AcademicCourse is registered.
Conclusion	Test is successful.

Table 1: Testing table of AcademicCourse

- The inspected AcademicCourse with registered academic course and re-inspected AcademicCourse presented below referencing screenshot:

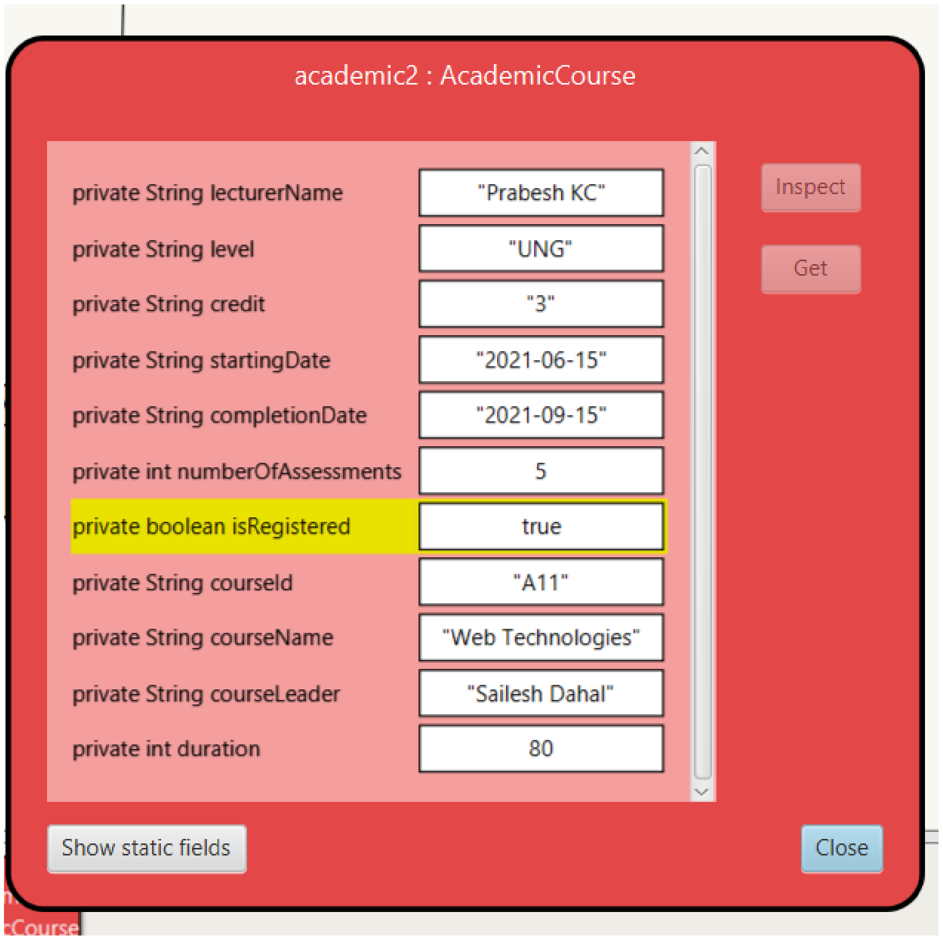
5.1.1. Inspected AcademicCourse before Invoke register method


academic2 : AcademicCourse

private String lecturerName	""
private String level	"UNG"
private String credit	"3"
private String startingDate	""
private String completionDate	""
private int numberOfAssessments	5
private boolean isRegistered	false
private String courseId	"A11"
private String courseName	"Web Technologies"
private String courseLeader	""
private int duration	80

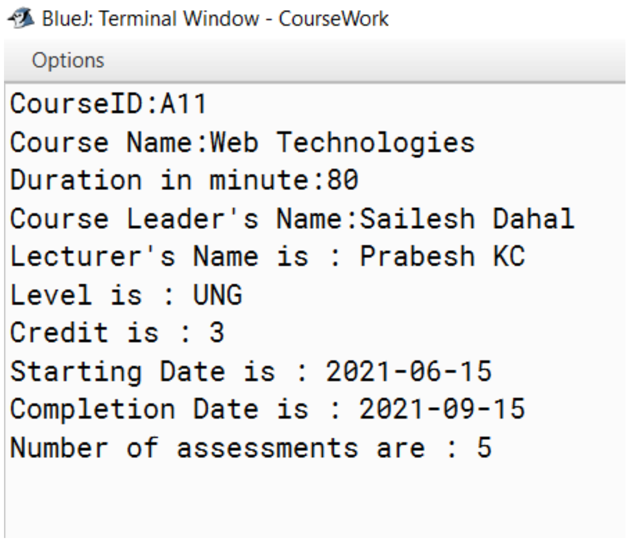
Figure 2: Inspected AcademicCourse class before invoking register method

5.1.2. Inspected AcademicCourse after invoking register method



private String lecturerName	"Prabesh KC"
private String level	"UNG"
private String credit	"3"
private String startingDate	"2021-06-15"
private String completionDate	"2021-09-15"
private int numberOfAssessments	5
private boolean isRegistered	true
private String courseId	"A11"
private String courseName	"Web Technologies"
private String courseLeader	"Sailesh Dahal"
private int duration	80

Figure 3:Inspected AcademicCourse class after invoking register method



```
CourseID:A11
Course Name:Web Technologies
Duration in minute:80
Course Leader's Name:Sailesh Dahal
Lecturer's Name is : Prabesh KC
Level is : UNG
Credit is : 3
Starting Date is : 2021-06-15
Completion Date is : 2021-09-15
Number of assessments are : 5
```

Figure 4: Display Output AcademicCourse class

5.2. Test 2: Inspect NonAcademicCourse class, register a non-academic course and re-inspect the NonAcademicCourse Class

- **Testing Table 2:**

Objective	Inspect NonAcademicCourse class, register a non-academic course and re-inspect NonAcademicCourse

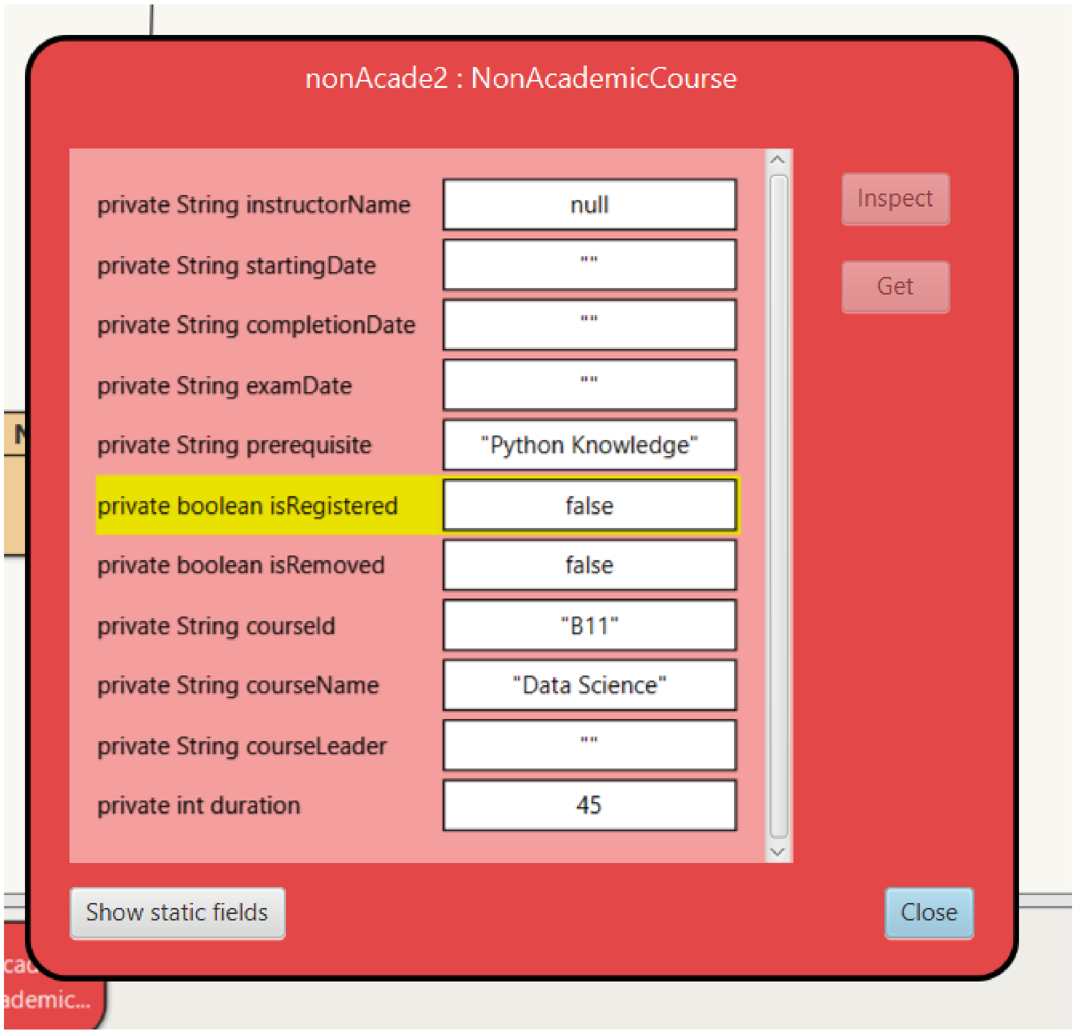
Action	Call Constructor of NonAcademicCourse class.
	courseId=”B11”
	courseName= “Data Science”
	duration=45
	prerequisite: “Python Knowledge”
	
	Calling method to register NonAcademicCourse
	courseLeaderName=”Kiran_Basnet”
	instructorName=”Bhabin_Khadka”
	startingDate=”2021-07-10”
	completionDate=”2021-10-10”
	examDate=”2021-12-05”

Expected Result	NonAcademicCourse should be registered.
Actual Result	NonAcademicCourse is registered.
Conclusion	Test is successful.

Table 2: Testing Table of NonAcademicCourse

- The inspected NonAcademicCourse with registered non-academic course and re-inspected NonAcademicCourse is presented below referencing screenshot:

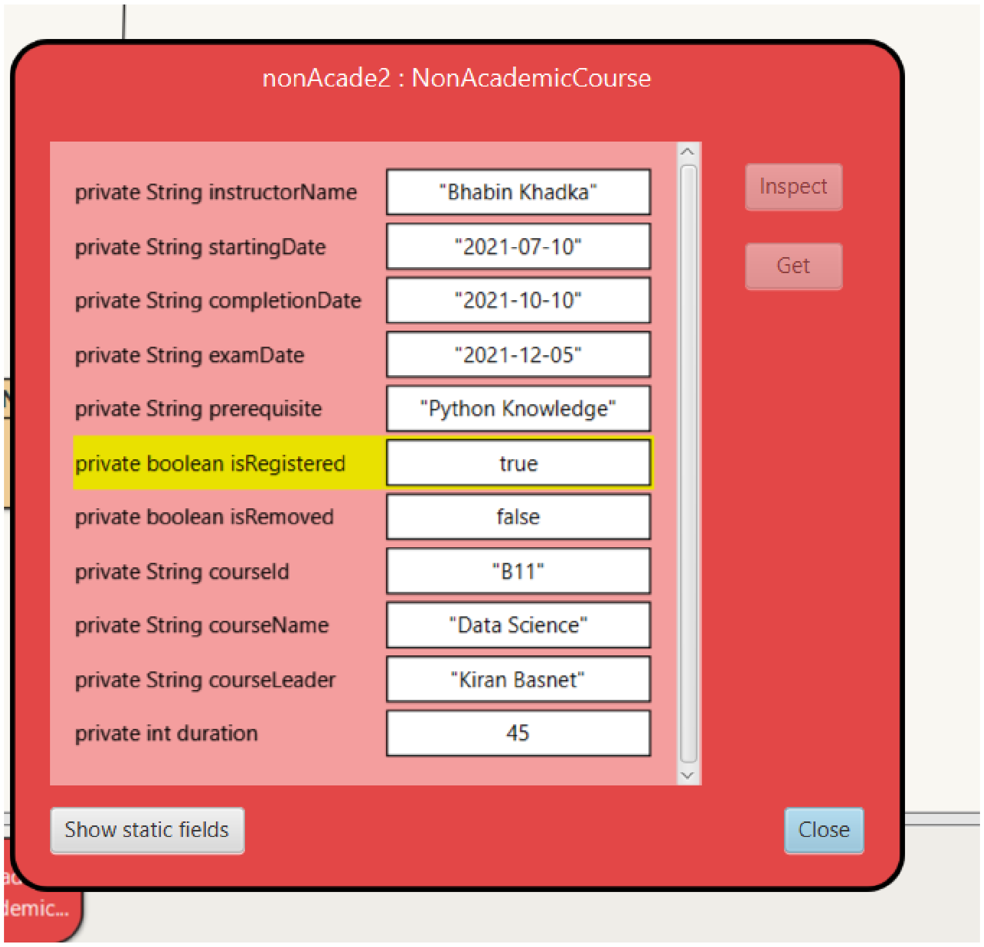
5.2.1. Inspected NonAcademicCourse before invoking register method:



private String instructorName	null
private String startingDate	""
private String completionDate	""
private String examDate	""
private String prerequisite	"Python Knowledge"
private boolean isRegistered	false
private boolean isRemoved	false
private String courseId	"B11"
private String courseName	"Data Science"
private String courseLeader	""
private int duration	45

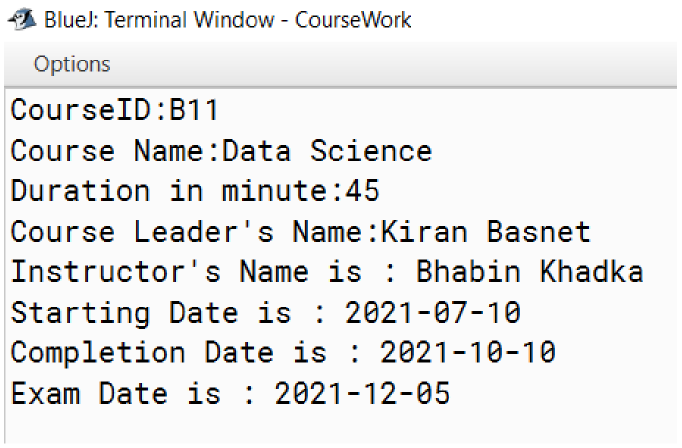
Figure 5: Inspected NonAcademicCourse class before invoking register method

5.2.2. Inspected NonAcademicCourse after invoking register method



private String instructorName	"Bhabin Khadka"
private String startingDate	"2021-07-10"
private String completionDate	"2021-10-10"
private String examDate	"2021-12-05"
private String prerequisite	"Python Knowledge"
private boolean isRegistered	true
private boolean isRemoved	false
private String courseId	"B11"
private String courseName	"Data Science"
private String courseLeader	"Kiran Basnet"
private int duration	45

Figure 6: Inspected NonAcademicCourse class after invoking register method



```
CourseID:B11
Course Name:Data Science
Duration in minute:45
Course Leader's Name:Kiran Basnet
Instructor's Name is : Bhabin Khadka
Starting Date is : 2021-07-10
Completion Date is : 2021-10-10
Exam Date is : 2021-12-05
```

Figure 7: Display Output of NonAcademicCourse class after invoking register method

5.3. Test 3: Inspect NonAcademicCourse class again, change the status of isRemoved to true and re-inspect the NonAcademicCourse class

- **Testing Table 3:**

Objective	Inspect NonAcademicCourse class, change the status of isRemoved to true and re-inspect NonAcademicCourse

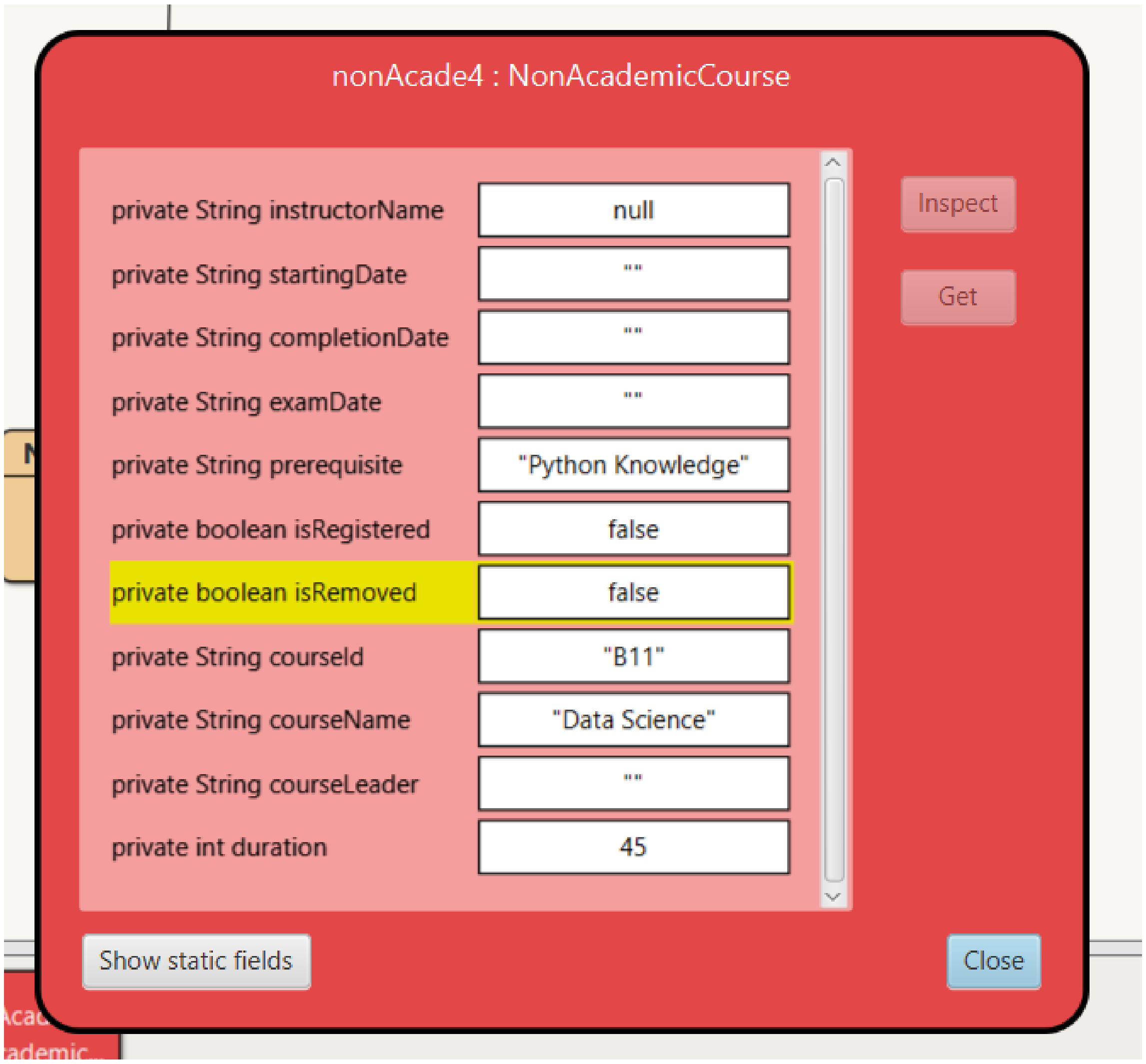
Action	Call Constructor of NonAcademicCourse class.
	courseId=”B11”
	courseName= “Data Science”
	duration=45
	prerequisite: “Python Knowledge”
	
	Calling method to remove NonAcademicCourse
	isRemoved= true

Expected Result	NonAcademicCourse should be removed.
Actual Result	The Non-academic course has been already removed.
Conclusion	Test is successful.

Table 3: Testing table of re-inspect NonAcademicCourse

- The inspected NonAcademicCourse class again, changed the status of isRemoved to true and re-inspected NonAcademicCourse is presented below referencing screenshot:

5.3.1. Inspected NonAcademicCourse before invoking remove method

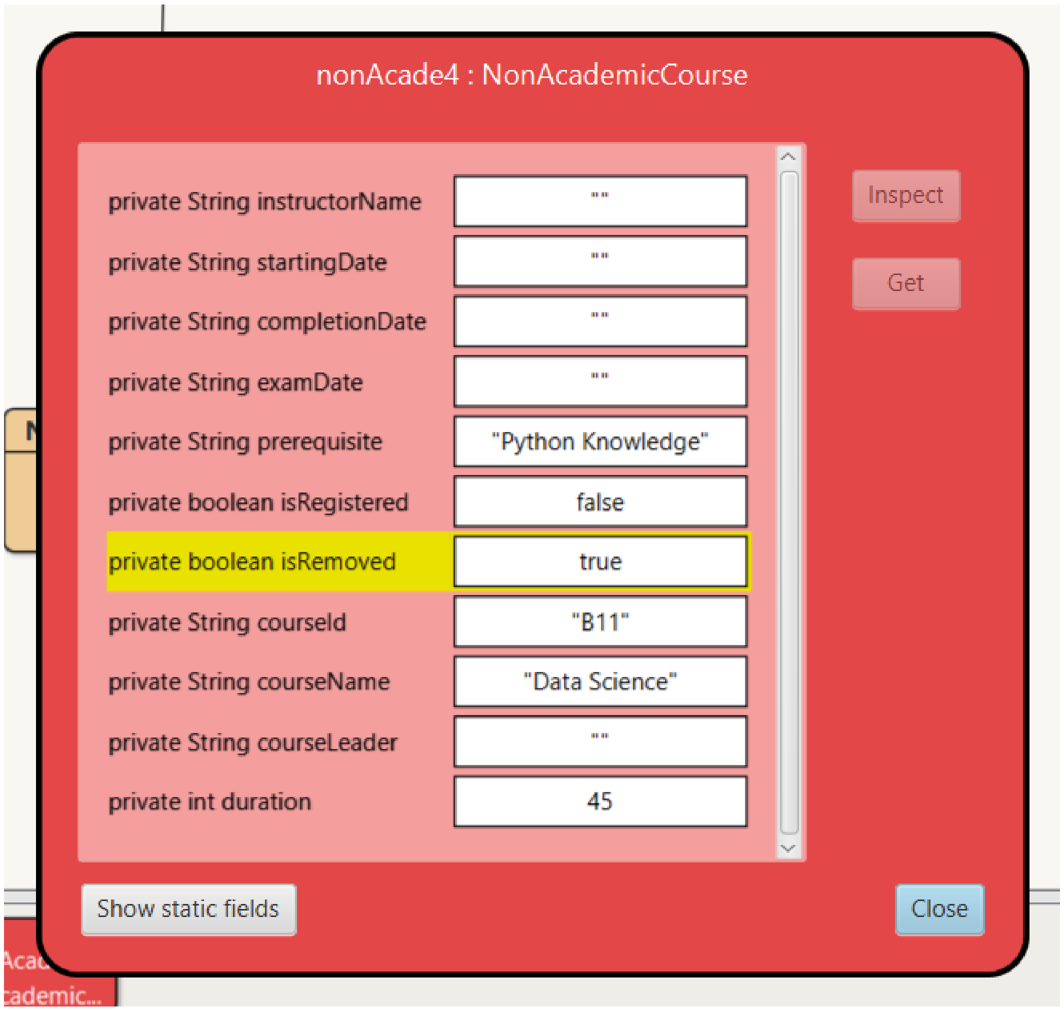


nonAcade4 : NonAcademicCourse

private String instructorName	null
private String startingDate	""
private String completionDate	""
private String examDate	""
private String prerequisite	"Python Knowledge"
private boolean isRegistered	false
private boolean isRemoved	false
private String courseId	"B11"
private String courseName	"Data Science"
private String courseLeader	""
private int duration	45

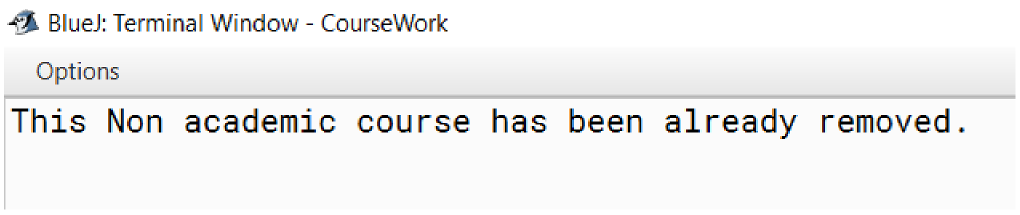
Figure 8: Inspected NonAcademicCourse before invoking remove method

5.3.2. Inspected NonAcademicCourse after invoking remove method



```
private String instructorName      ""
private String startingDate        ""
private String completionDate      ""
private String examDate            ""
private String prerequisite        "Python Knowledge"
private boolean isRegistered       false
private boolean isRemoved          true
private String courseId            "B11"
private String courseName          "Data Science"
private String courseLeader        ""
private int duration               45
```

Figure 9: Inspected NonAcademicCourse after invoking remove method



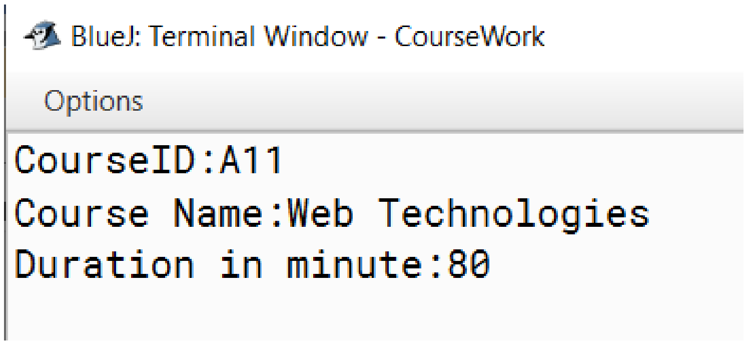
```
This Non academic course has been already removed.
```

Figure 10: Display output after invoking remove method

5.4. Test 4: Display the detail of AcademicCourse and NonAcademicCourse classes.

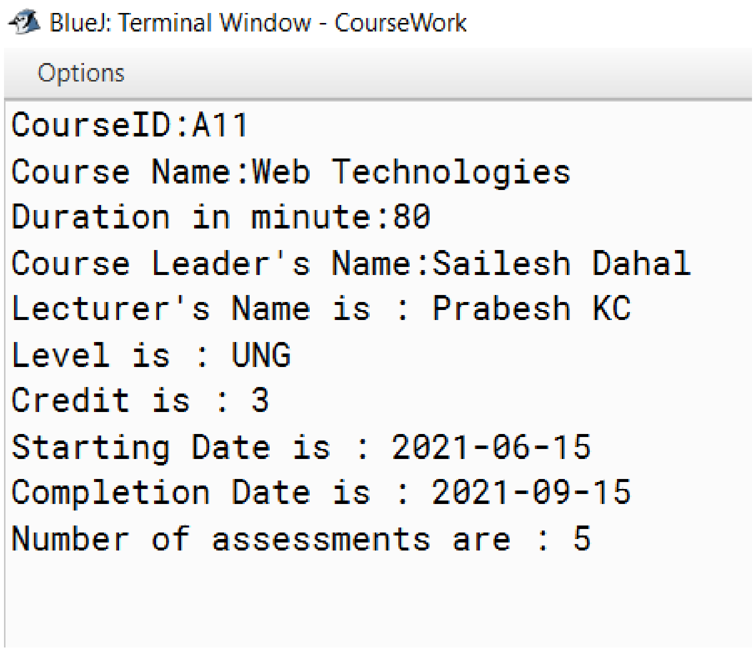
The output from each method of AcademicCourse and NonAcademicCourse are referenced by screenshots below:

5.4.1. AcademicCourse Class Screenshots of each method:



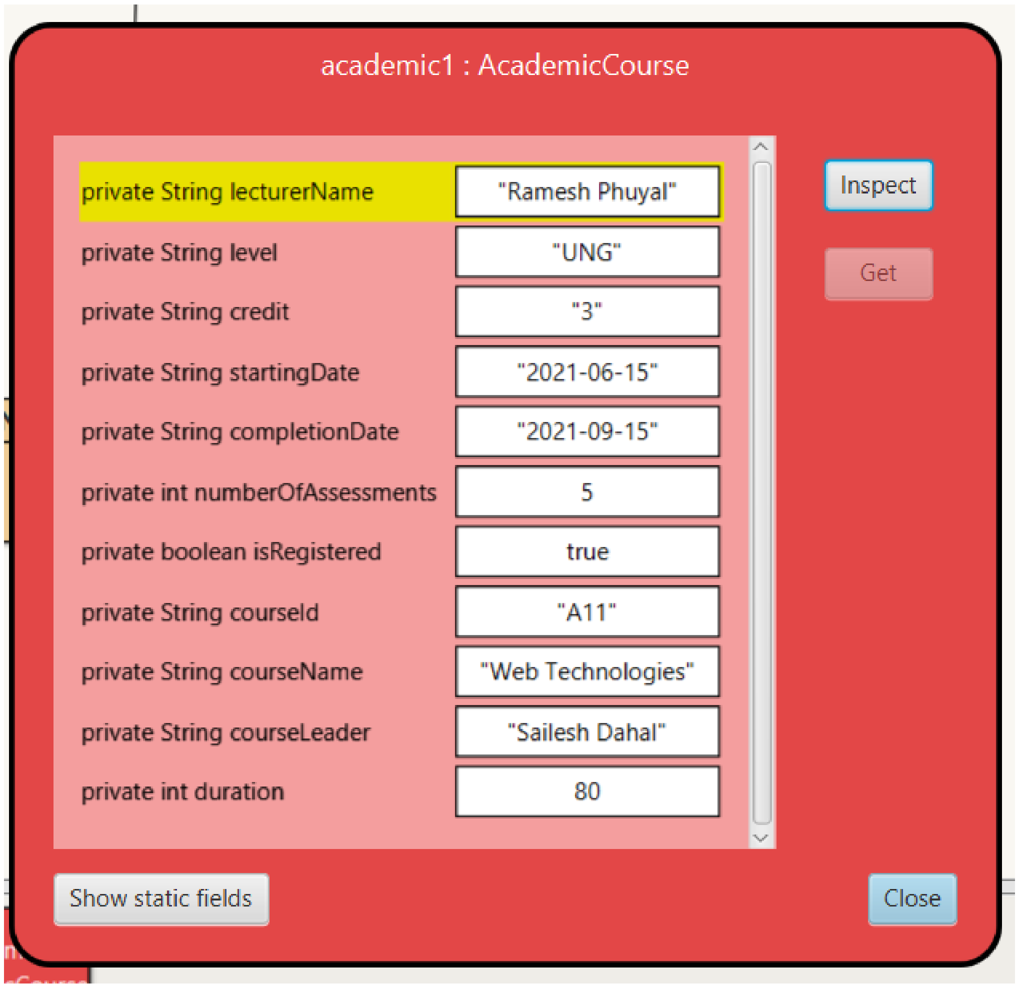
```
CourseID:A11
Course Name:Web Technologies
Duration in minute:80
```

Figure 11:Output AcademicCourse display method



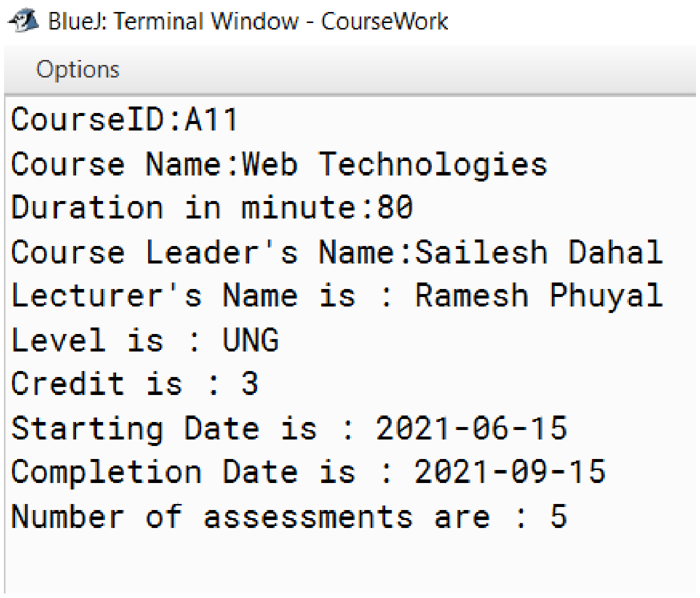
```
CourseID:A11
Course Name:Web Technologies
Duration in minute:80
Course Leader's Name:Sailesh Dahal
Lecturer's Name is : Prabesh KC
Level is : UNG
Credit is : 3
Starting Date is : 2021-06-15
Completion Date is : 2021-09-15
Number of assessments are : 5
```

Figure 12:Output AcademicCourse class register method



private String lecturerName	"Ramesh Phuyal"
private String level	"UNG"
private String credit	"3"
private String startingDate	"2021-06-15"
private String completionDate	"2021-09-15"
private int numberOfAssessments	5
private boolean isRegistered	true
private String courseId	"A11"
private String courseName	"Web Technologies"
private String courseLeader	"Sailesh Dahal"
private int duration	80

Figure 13: Inspected by setting new name of Lecturer

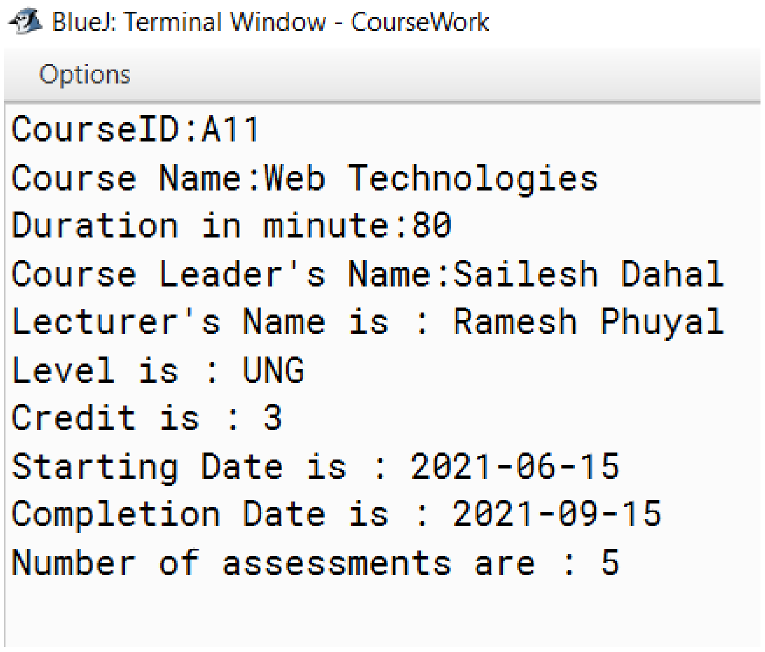


BlueJ: Terminal Window - CourseWork

Options

```
CourseID:A11
Course Name:Web Technologies
Duration in minute:80
Course Leader's Name:Sailesh Dahal
Lecturer's Name is : Ramesh Phuyal
Level is : UNG
Credit is : 3
Starting Date is : 2021-06-15
Completion Date is : 2021-09-15
Number of assessments are : 5
```

Figure 14:Output from setLecturerName method

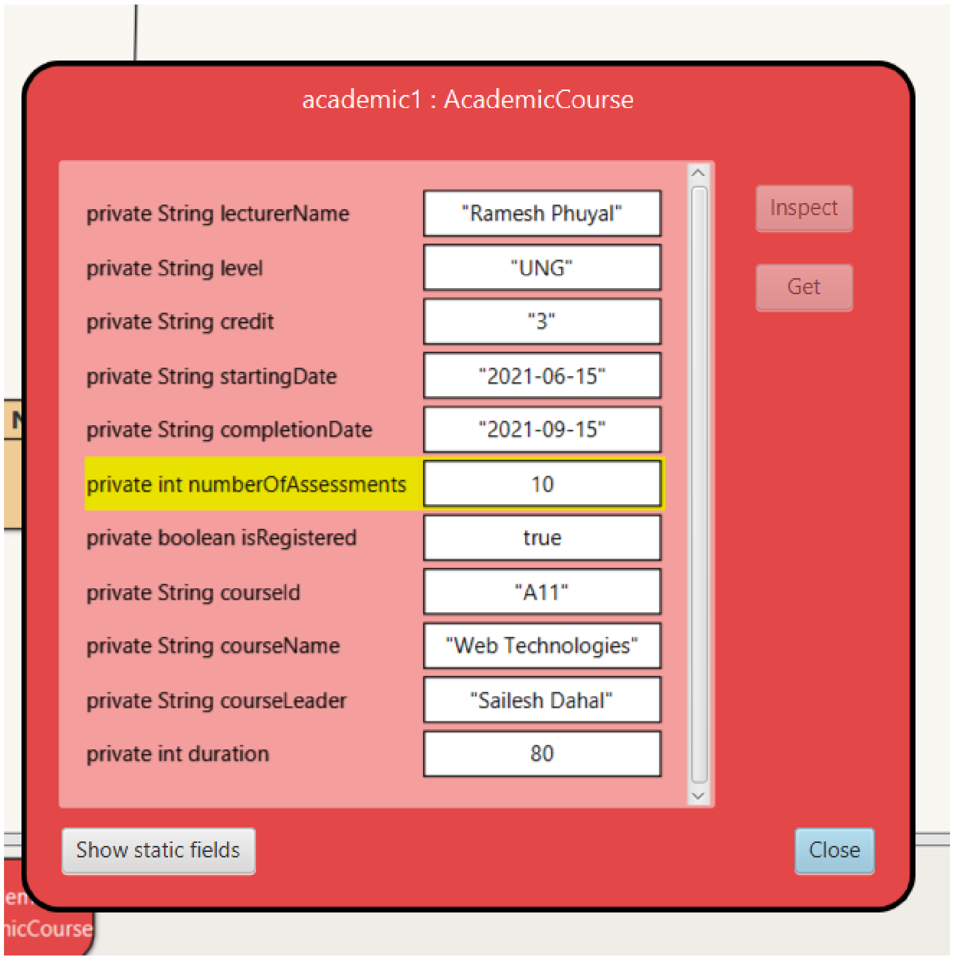


BlueJ: Terminal Window - CourseWork

Options

```
CourseID:A11
Course Name:Web Technologies
Duration in minute:80
Course Leader's Name:Sailesh Dahal
Lecturer's Name is : Ramesh Phuyal
Level is : UNG
Credit is : 3
Starting Date is : 2021-06-15
Completion Date is : 2021-09-15
Number of assessments are : 5
```

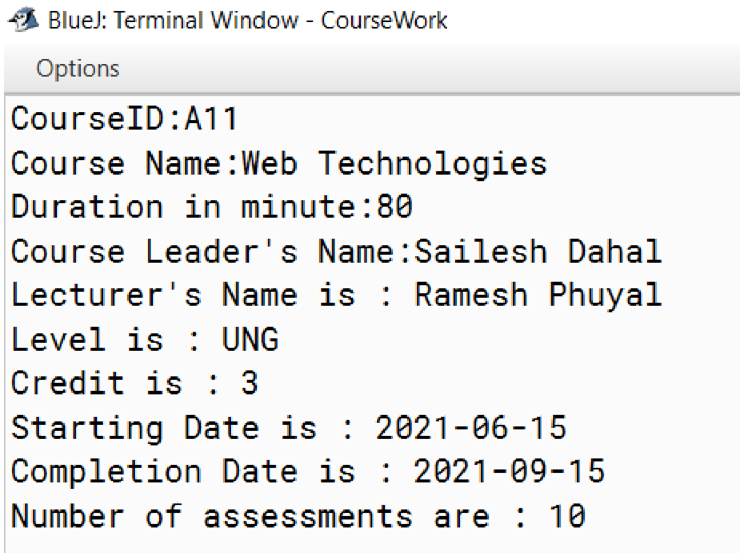
Figure 15:Output before calling setNumberOfAssessments method



academic1 : AcademicCourse

private String lecturerName	"Ramesh Phuyal"
private String level	"UNG"
private String credit	"3"
private String startingDate	"2021-06-15"
private String completionDate	"2021-09-15"
private int numberOfAssessments	10
private boolean isRegistered	true
private String courseId	"A11"
private String courseName	"Web Technologies"
private String courseLeader	"Sailesh Dahal"
private int duration	80

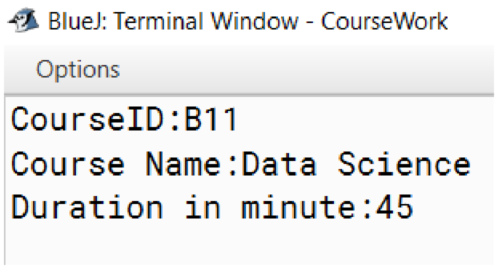
Figure 16: Inspected after calling setNumberOfAssessments



```
CourseID:A11
Course Name:Web Technologies
Duration in minute:80
Course Leader's Name:Sailesh Dahal
Lecturer's Name is : Ramesh Phuyal
Level is : UNG
Credit is : 3
Starting Date is : 2021-06-15
Completion Date is : 2021-09-15
Number of assessments are : 10
```

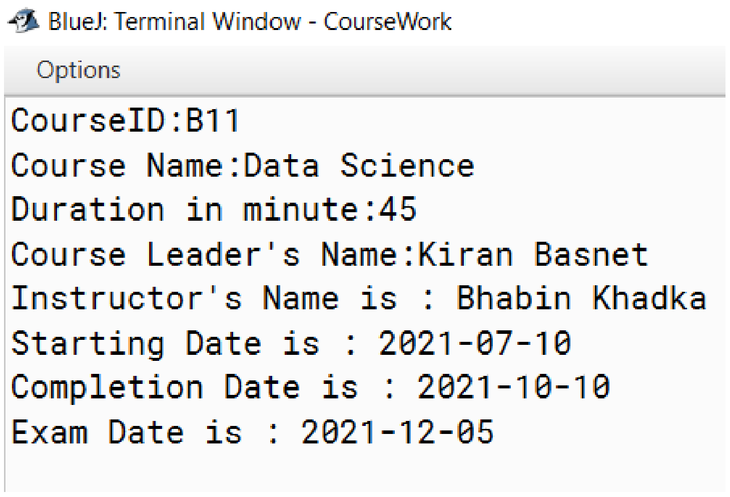
Figure 17: Output from setNumberOfAssessments

5.4.2. NonAcademicCourse class screenshots of each method:



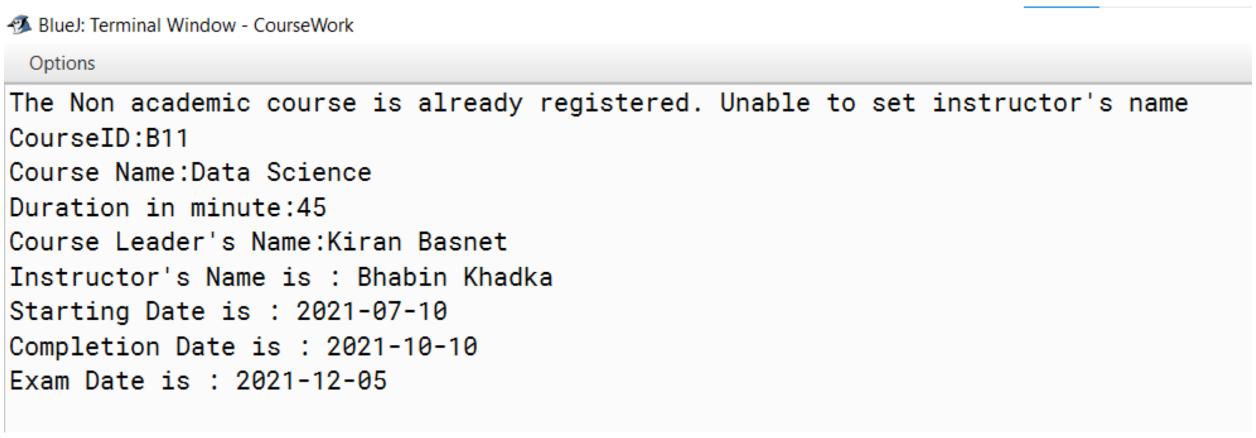
```
CourseID:B11
Course Name:Data Science
Duration in minute:45
```

Figure 18: Output from display method in NonAcademicCourse



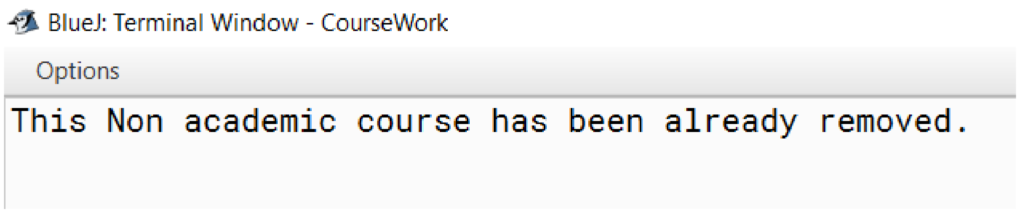
```
CourseID:B11
Course Name:Data Science
Duration in minute:45
Course Leader's Name:Kiran Basnet
Instructor's Name is : Bhabin Khadka
Starting Date is : 2021-07-10
Completion Date is : 2021-10-10
Exam Date is : 2021-12-05
```

Figure 19: Output from register method in NonAcademicCourse



```
The Non academic course is already registered. Unable to set instructor's name
CourseID:B11
Course Name:Data Science
Duration in minute:45
Course Leader's Name:Kiran Basnet
Instructor's Name is : Bhabin Khadka
Starting Date is : 2021-07-10
Completion Date is : 2021-10-10
Exam Date is : 2021-12-05
```

Figure 20: Output from setInstructorName method


BlueJ: Terminal Window - CourseWork

Options

```
This Non academic course has been already removed.
```

Figure 21: Output from remove method when it is true.

6. Error detection and correction:

The error detection and correction are done while coding. Screenshots are taken when encountered them. For reference screenshots is available:

6.1. Syntax Error :

I noticed there was missing “void” return type in method. So, after writing it, error was resolved.

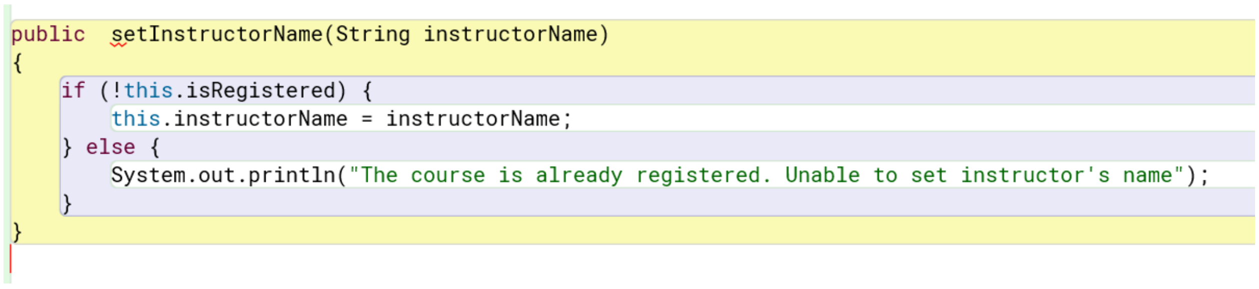

```java
public  setInstructorName(String instructorName)
{
    if (!this.isRegistered) {
        this.instructorName = instructorName;
    } else {
        System.out.println("The course is already registered. Unable to set instructor's name");
    }
}
```

Figure 22: Setter method syntax error

6.2. Correction:

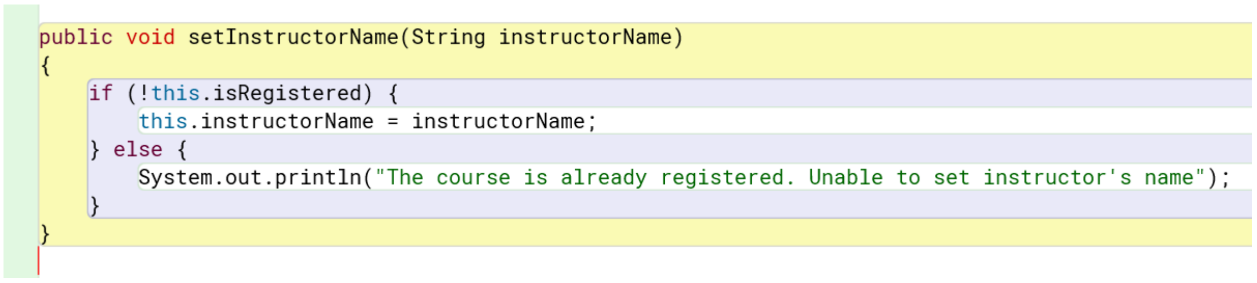
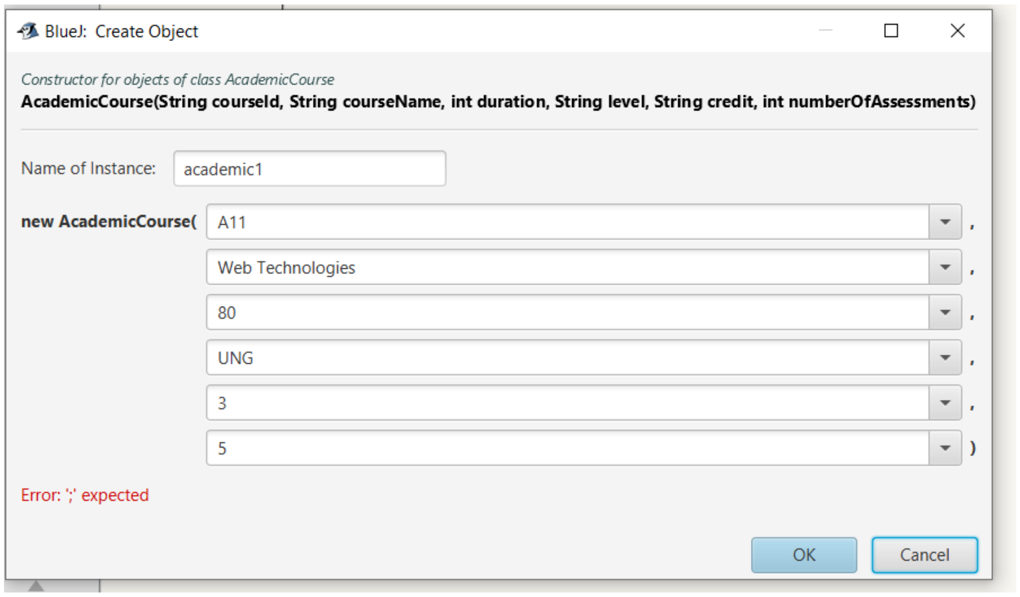

```java
public void setInstructorName(String instructorName)
{
    if (!this.isRegistered) {
        this.instructorName = instructorName;
    } else {
        System.out.println("The course is already registered. Unable to set instructor's name");
    }
}
```

Figure 23: Correction of setter method

There is always void return type in setter method. Later, I noticed and I to write void return type in the method. I have provided correction with void return type in setter method.

6.3. Semantics Error detection :

I detected error while creating object in BlueJ and providing value.


BlueJ: Create Object

Constructor for objects of class AcademicCourse

AcademicCourse(String courseId, String courseName, int duration, String level, String credit, int numberOfAssessments)

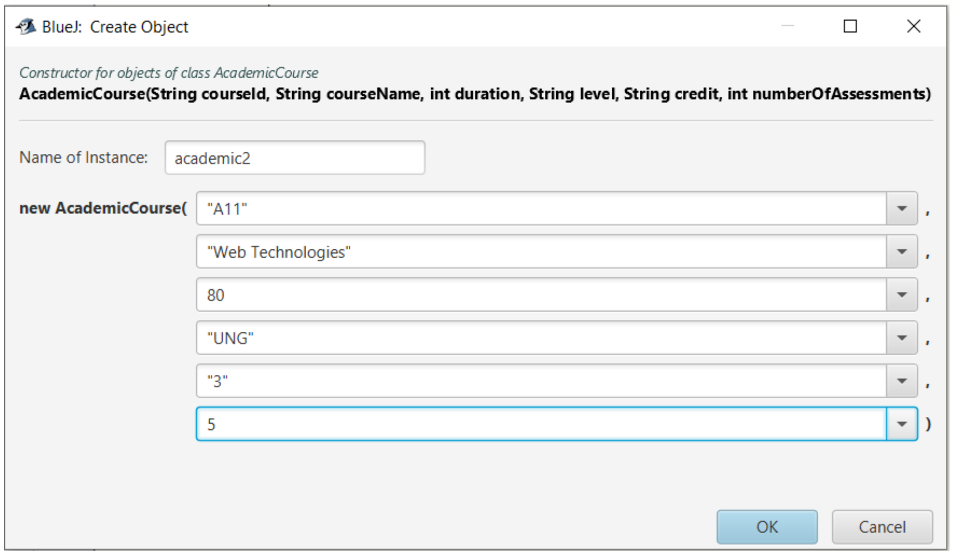
Name of Instance: academic1

new AcademicCourse(A11 , Web Technologies , 80 , UNG , 3 , 5)

Error: ';' expected

Figure 24: Semantics error

6.4. Correction of Semantics Error:



Constructor for objects of class AcademicCourse
AcademicCourse(String courseId, String courseName, int duration, String level, String credit, int numberOfAssessments)

Name of Instance: academic2

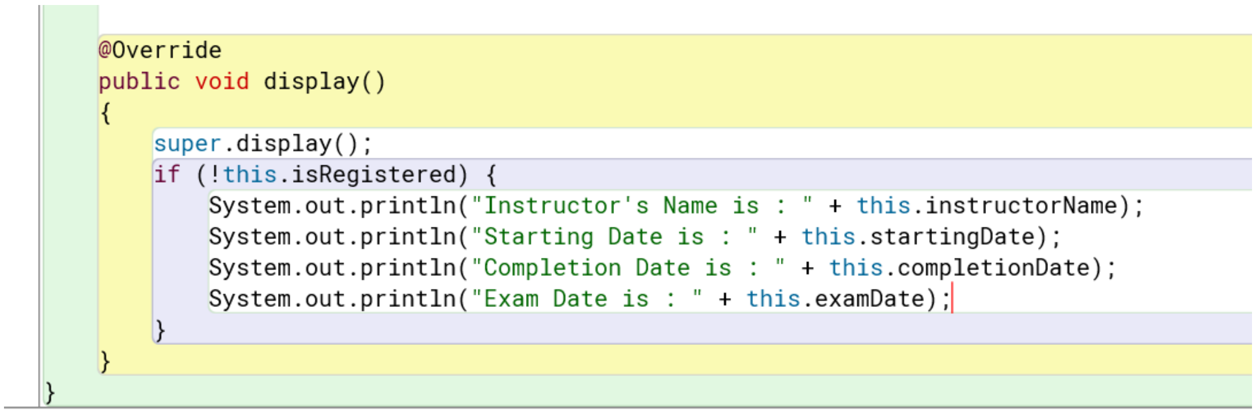
new AcademicCourse("A11" , "Web Technologies" , 80 , "UNG" , "3" , 5)

Figure 25: Solution Of Error

There was error I forgot to provide value inside double quotation for String data type. So, I provide string data type inside double quotation and error solved.

6.5. Logical error in if condition:

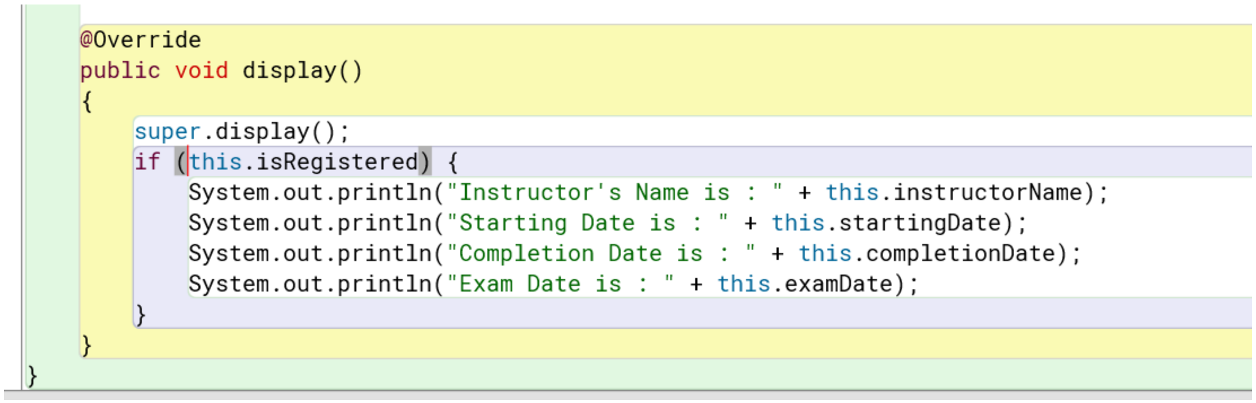
There was not compilation error but there was runtime error and program were not responding as I wanted the error is presented below:



```
    @Override
    public void display()
    {
        super.display();
        if (!this.isRegistered) {
            System.out.println("Instructor's Name is : " + this.instructorName);
            System.out.println("Starting Date is : " + this.startingDate);
            System.out.println("Completion Date is : " + this.completionDate);
            System.out.println("Exam Date is : " + this.examDate);
        }
    }
}
```

Figure 26: Logical error

6.6. Correction:



```
@Override
public void display()
{
    super.display();
    if (this.isRegistered) {
        System.out.println("Instructor's Name is : " + this.instructorName);
        System.out.println("Starting Date is : " + this.startingDate);
        System.out.println("Completion Date is : " + this.completionDate);
        System.out.println("Exam Date is : " + this.examDate);
    }
}
}
```

Figure 27: Correction in display method if condition logical error

I didn't get compilation error but the code wasn't giving output as I wanted to. I checked the code and found there's a mistake in if condition there was negation of the condition which impacted on the result and I checked result after removing negation and output was good as I wanted.

7. Conclusion:

7.1. Evaluation of your work:

After and working on the project there were many important findings. There were many things that need to be done. Without the help of the respective module leaders, it was quite difficult to work on it. The module leaders helped a lot they made clear on concepts and I can finish the project on time. For this project researching on internet is necessary, so as I did.

7.2. Reflection on what you learnt from the assignment:

This project mainly focuses on the concept of inheritance, polymorphism, data encapsulation, parametrized constructor etc. This project helps one to learn different programming concept like creating objects, method invoking, concept of ‘this’ and ‘super’ keyword, access modifiers etc. We learn getter and setter method and why it is important to implement accessor methods. We also learn to find errors understand them and solve it. We also learnt to test our project from BlueJ and knew how to inspect it.

7.3. What difficulties you encountered and:

The project was quite tough for Java programming beginner. The concept of Data Encapsulation (getter and setter method) and the concept of calling parent class method from child class with super keyword, class diagrams, pseudocode, and inspecting classes through BlueJ were the problems at the beginning.

7.4. How you overcame the difficulties:

Data encapsulation(setter and getter method) and the concept of super keyword concepts were the difficulties and they were solved by consulting with our respective module leaders. They cleared the concept and gave idea how to draw class diagram, and how to write pseudocode. They also guided us how to create object, inspect classes from BlueJ. The internet was also the great source for researching the problems. Tutorials in YouTube, information in different well-known websites also helped us.