Module Code & Module Title

CS4001NI Programming

COURSEWORK-1

Assessment Weightage & Type

30% Individual Coursework

Semester and Year

2021 Spring

Student Name: Sujen Shrestha

Group: N4

London Met ID: 20049250

College ID: NP01NT4S210105

Assignment Due Date: May 23, 2021

Assignment Submission Date: May 20, 2021

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Introduction to project content

Java is a class-based, object-oriented programming language that can be developed and executed for any device platforms. It is a fast, secure and reliable platform for program development. Therefore, it is commonly used for creating programs and developing applications in desktop computers, gaming consoles, mobile phones, etc. (Guru99, 2021)

BlueJ is an Integrated Development Environment (IDE) for Java programming language, created mainly for educational purpose, but also useful for small-scale program development. It was developed in Monash University particularly to help in learning the object-oriented programming. (N K, 2021)

This is the first coursework of programming module. The coursework was completed using various tools like Bluej, Draw.io, etc. The purpose of this coursework is to create a “Course” class which consists of two sub classes: “AcademicCourse” and “NonAcademicCourse”. The program consists of various structures of constructors, accessors, mutators and display methods and conditional statements, for performing actions on specific attributes. The “Course” is a parent class and “AcademicCourse” and “NonAcademicCourse” are the child classes. The accessor and mutator methods are used to assign and return the values to the class. The super or parent class is called in order to display the information of the AcademicCourse and NoonAcademicCourse classes.
The constructors of the classes are assigned with the parameters which are to be accepted. The attributes are also assigned with different values. Each of the attributes of all classes have the accessor method or the getter method and the mutator method or the setter method is also used in some attributes to assign or set the new value.

2. Class Diagram

A class diagram is a static diagram which represents the static view of an application. Class diagrams are used for visualising, describing, documenting and constructing executable codes for a software application. It describes the attributes and operations of a class and also the constraints applied on the system. The class diagrams are commonly used in the designing of object-oriented systems since they are the only UML diagrams, that can be mapped directly with object-oriented languages. A class diagram displays a collection of classes, associations collaborations, interfaces and constraints. It is also known as a structural diagram. (Tutorialspoint, 2021)

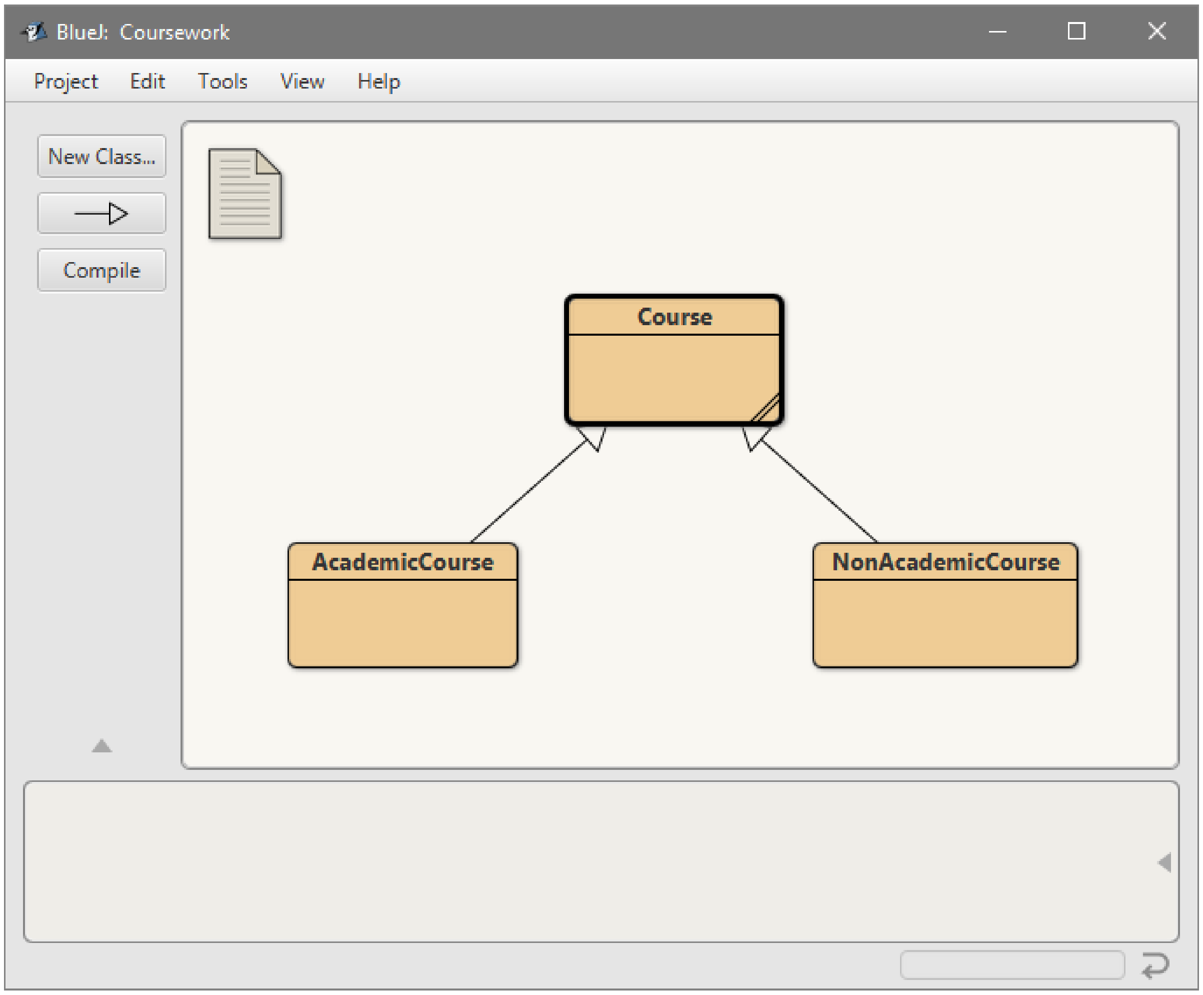


Figure 1: Class diagram of the classes in Bluej

2.1 Course

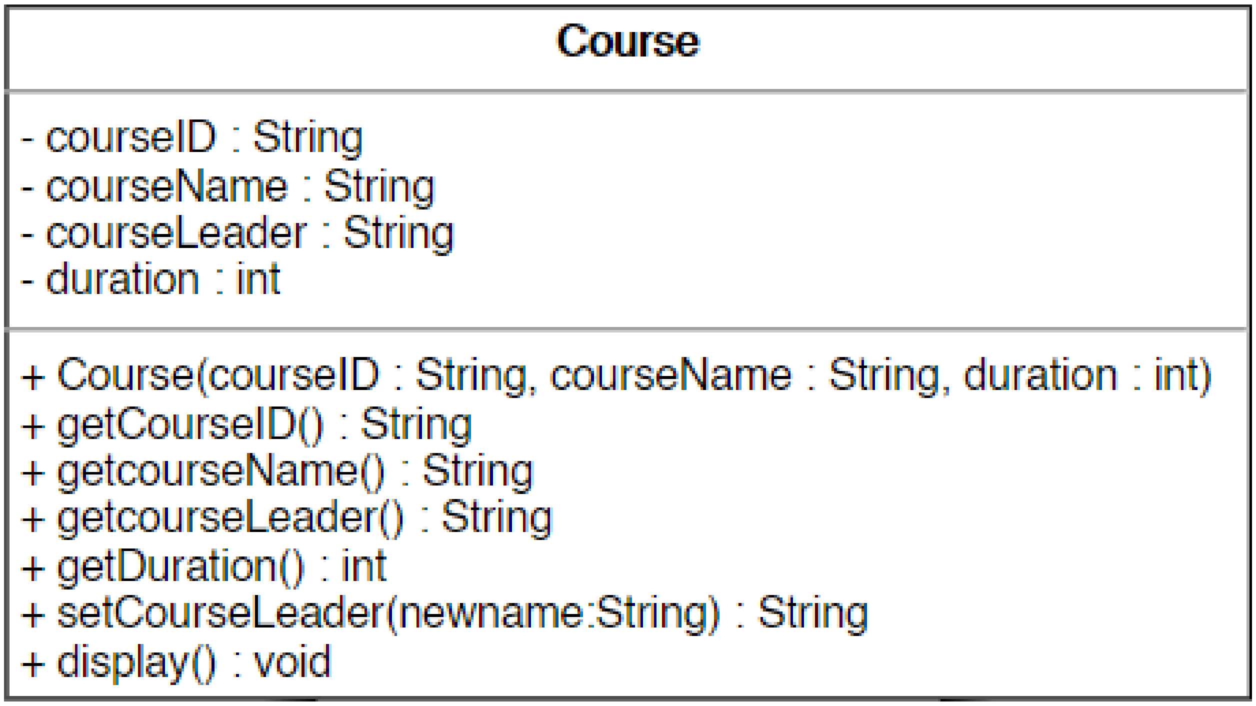

Course
- courseID : String
- courseName : String
- courseLeader : String
- duration : int
+ Course(courseID : String, courseName : String, duration : int)
+ getCourseID() : String
+ getcourseName() : String
+ getcourseLeader() : String
+ getDuration() : int
+ setCourseLeader(newname:String) : String
+ display() : void

Figure 2: Course Class Diagram

2.2 AcademicCourse

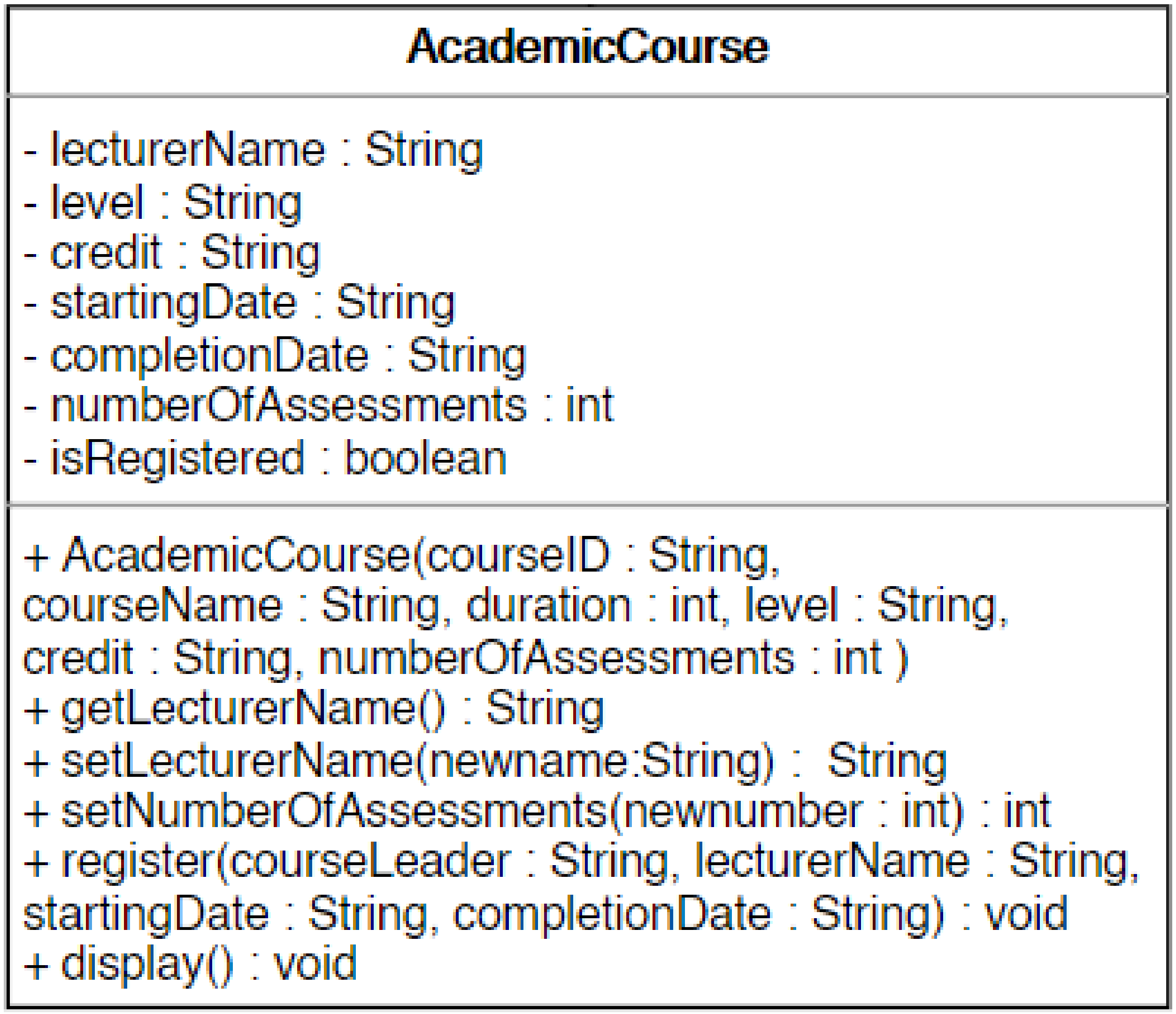
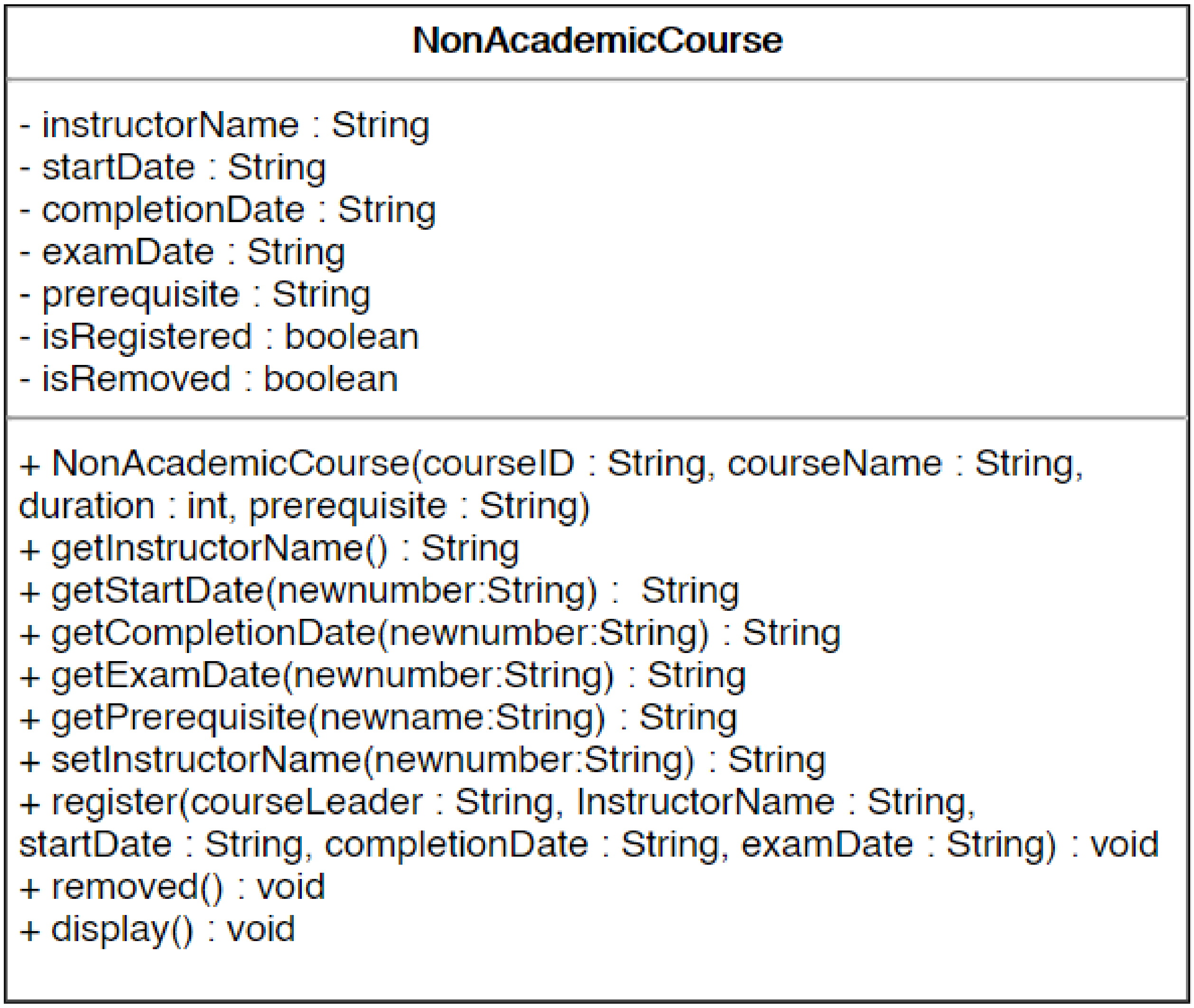

AcademicCourse
- lecturerName : String
- level : String
- credit : String
- startingDate : String
- completionDate : String
- numberOfAssessments : int
- isRegistered : boolean
+ AcademicCourse(courseID : String, courseName : String, duration : int, level : String, credit : String, numberOfAssessments : int)
+ getLecturerName() : String
+ setLecturerName(newname:String) : String
+ setNumberOfAssessments(newnumber : int) : int
+ register(courseLeader : String, lecturerName : String, startingDate : String, completionDate : String) : void
+ display() : void

Figure 3: AcademicCourse Class Diagram

2.3 NonAcademicCourse



NonAcademicCourse

- instructorName : String
- startDate : String
- completionDate : String
- examDate : String
- prerequisite : String
- isRegistered : boolean
- isRemoved : boolean

+ NonAcademicCourse(courseID : String, courseName : String, duration : int, prerequisite : String)
+ getInstructorName() : String
+ getStartDate(newnumber:String) : String
+ getCompletionDate(newnumber:String) : String
+ getExamDate(newnumber:String) : String
+ getPrerequisite(newname:String) : String
+ setInstructorName(newnumber:String) : String
+ register(courseLeader : String, InstructorName : String, startDate : String, completionDate : String, examDate : String) : void
+ removed() : void
+ display() : void

Figure 4: NonAcademicCourse Class Diagram

2.4 Combined Class Diagram

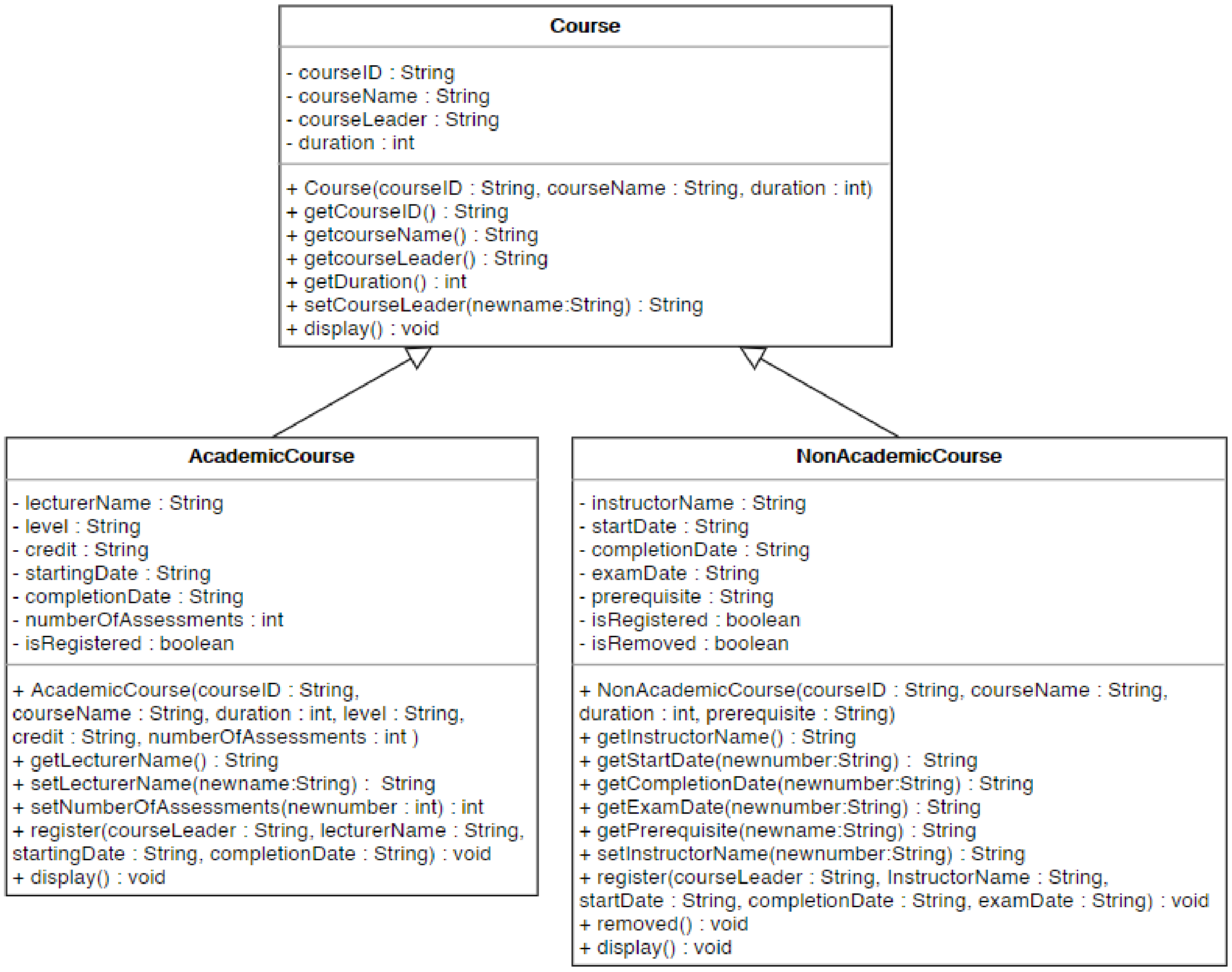

Course
- courseID : String
- courseName : String
- courseLeader : String
- duration : int
+ Course(courseID : String, courseName : String, duration : int)
+ getCourseID() : String
+ getcourseName() : String
+ getcourseLeader() : String
+ getDuration() : int
+ setCourseLeader(newname:String) : String
+ 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
+ setLecturerName(newname:String) : String
+ setNumberOfAssessments(newnumber : int) : int
+ register(courseLeader : String, lecturerName : String, startingDate : String, completionDate : String) : void
+ display() : void
NonAcademicCourse
- instructorName : String
- startDate : String
- completionDate : String
- examDate : String
- prerequisite : String
- isRegistered : boolean
- isRemoved : boolean
+ NonAcademicCourse(courseID : String, courseName : String, duration : int, prerequisite : String)
+ getInstructorName() : String
+ getStartDate(newnumber:String) : String
+ getCompletionDate(newnumber:String) : String
+ getExamDate(newnumber:String) : String
+ getPrerequisite(newname:String) : String
+ setInstructorName(newnumber:String) : String
+ register(courseLeader : String, InstructorName : String, startDate : String, completionDate : String, examDate : String) : void
+ removed() : void
+ display() : void

Figure 5: Combined Class Diagram

3. Pseudocode

A pseudocode is an unofficial way of coding description which does not need any programming language syntax or semantics. It is used for developing an outline of a program to understand the methods used in it. It is not an actual programming language so it cannot be compiled. It only summarizes the programs methods. (The Economic Times, 2021)

3.1 Course class pseudocode

```
CREATE class Course
    DEFINE four instance variables with private access modifiers as String courseID,
    String courseName, String courseLeader, int duration
    CREATE a constructor Course and initialize the variables
        INITIALIZE courseID as courseID
        INITIALIZE courseName as courseName
        INITIALIZE courseLeader as empty String
        INITIALIZE duration as duration
    DEFINE method getCourseID() as String
        RETURN courseID
     DEFINE method getCourseName() as String
        RETURN coursename
    DEFINE method getcourseLeader() as String
        RETURN courseLeader
    DEFINE method setCourseLeader() as String
        SET the value of courseLeader
    DEFINE method display()
        SET output as courseID + " " + courseName " " + duration
        PRINT message as string and output
        IF this.courseLeader is empty String
            PRINT suitable message showing courseID, courseName, duration
            and courseLeader
        END IF
        ELSE
            PRINT suitable message with courseID, courseName and duration
        END ELSE
```

3.2 AcademicCourse class pseudocode

```
CREATE class AcademicCourse
    DEFINE seven instance variables with private access modifiers as String
    lecturerName, String level, String credit, String startingDate, String
    completionDate, int numberOfAssessments, boolean isRegistered
    CREATE a constructor AcademicCourse and initialize the variables
        INITIALIZE super constructor with required parameters
        INITIALIZE lecturerName as empty String
        INITIALIZE level as level
        INITIALIZE credit as credit
        INITIALIZE startingDate as empty String
        INITIALIZE completionDate as empty String
        INITIALIZE numberOfAssessments as numberOfAssessments
        INITIALIZE isRegistered as false
    DEFINE method getLecturerName() as String
        RETURN lecturerName
    DEFINE method getNumberOfAssessments() as String
        RETURN numberOfAssessments
    DEFINE method setLecturerName() as String
        SET the value of lecturer
    DEFINE method setNumberOfAssessments() as String
        SET the value of assessments
    DEFINE method register()
        IF isRegistered is true
```

```
            PRINT suitable message showing lecturerName, startingDate and
            completionDate
        END IF
        ELSE
            CALL the super class variable courseLeader with courseLeader
            name as parameter
            SET the value of lecturerName
            SET the value of startingDate
            SET the value of completionDate
            SET the value of isRegistered to true
            PRINT suitable message showing courseLeader, lecturerName,
            startingDate and completionDate
        END ELSE
    DEFINE method display()
        CALL super class display method
            IF isRegistered is true
                PRINT suitable message showing lecturerName, l      evel,
                credit, startingDate, completionDate and
                numberOfAssessments
                END IF
```

3.3 NonAcedemicCourse class pseudocode

```
CREATE class NonAcademicCourse
    DEFINE seven instance variable with private access modifiers as String
    instructorName, Sring startDate, String completionDate, String examDate, String
    prerequisite, boolean isRegistered and boolean isRemoved
    CREATE a constructor NonAcademicConstructor and initialize the variables
        INITIALIZE super constructor with required paraeters
        INITIALIZE instructorName as instructorName
        INITIALIZE startDate as startDate
        INITIALIZE completionDate as completionDate
        INITIALIZE examDate as examDate
        INITIALIZE prerequisite as prerequisite
        INITIALIZE isRegistered as isRegistered
        INITIALIZE isRemoved as isRemoved
    DEFINE method getInstructorName() as String
        RETURN InstructorName
    DEFINE method getStartDate() as String
        RETURN startDate
    DEFINE method getCompletionDate() as String
        RETURN completionDate
    DEFINE method getExamDate () as String
        RETURN examDate
    DEFINE method getPrerequisite () as String
        RETURN prerequisite
    DEFINE method setInstructorName() as String
        IF isRegistered is false
            SET the value of insturctorName
        END IF
        ELSE
            PRINT suitable message
        END IF
    DEFINE method register()
        IF isRegistered is false
            CALL super class variable courseLeader with courseLeader name
            as parameter
            SET instructorName with new instructorName as parameter
            SET the value of startDate
            SET the calue of completionDate
            SET the value of examDate
```

```
                SET the value of isRegistered to true
                PRINT suitable message showing the details of mutators used
        END IF
        ELSE
                PRINT suitable message
        END ELSE
DEFINE method remove()
        SET courseLeader with empty String as parameter
        SET instructorName as empty String
        SET startDate as empty String
        SET completeDate as empty String
        SET examDate as empty String
        SET isRegistered as false
        SET isRemoved as true
        IF isRemoved is true
                PRINT suitable message
        END IF
DEFINE method display()
                CALL super display method
                IF isRegistered is true
                        PRINT suitable message showing instructorName, startDate,
                        completionDate and examDate
                END IF
```

4. Method Description

In this program various methods have been used. We have three class which have used the following methods:

4.1 Course Class

The methods used in Course class are given below:

- getCourseID()
  This is an accessor method that will return courseID as String type
- getCourseName()
  This is an accessor method that will return courseName as String type
- getCourseLeader()
  This is an accessor method that will return courseLeader as String type
- getDuration()
  This is an accessor method that will return duration as int type
- setCourseLeader()
  This is a mutator method that accepts the courseLeader variable of String type and set the value to its instance variable
- display()
  This method displays a suitable message with courseID, courseName, duration and courseLeader if the courseLeader is “” else it diplays a suitable message without the courseLeader

4.2 AcademicCourse

The methods used in the AcademicCourse class are given below:

- getLecturerName
  This is an accessor method that will return lecturerName as String type
- getNumberOfAssessments
  This is an accessor method that will return numberOfAssessments as String type

- getLecturerName

  This is an accessor method that will return lecturerName as String type
- setLecturerName

  This is a mutator method that accepts the lecturerName variable of String type and set the value to its instance variable
- setNumberOfAssessments

  This is a mutator method that accepts the numberOfAssessments variable as String type and set the value to its instance variable
- register(String courseLeader, String lecturerName, String startingDate, String completionDate)

  This method gives a suitable message showing lecturerName, startingDate and completionDate if isRegistered is true, else it sets the values for variables super courseLeader with courseLeader name as parameter, lecturerName, startingDate, completionDate, isRegistered to true and gives a suitable message showing the variables used.
- display()

  This method displays the output from super class and if the course is registered then the lecturerName, level, credit, startingDate, completionDate and numberOfAssessments are also shown with a suitable message

4.3 NonAcademicCourse

The methods used in the NonAcademicCourse are given below:

- getInstructorName

  This is an accessor method that will return InstructorName as String type
- getStartDate

  This is an accessor method that will return startDate as String type
- getCompletionDate

  This is an accessor method that will return completionDate as String type
- getExamDate

This is an accessor method that will return examDate as String type

- getPrerequisite

  This is an accessor method that will return prerequisite as String type
- setInstructorName

  This is a mutator method that exetues if the course is not registered then it accepts the instructorName and set the value to its instance variable else it shows a suitable message
- register(String courseLeader, String instructorName, String startDate, String completionDate, String examDate)

  This method sets values to variables super class variable courseLeader with courseLeader name as parameter, instructorName with new instructorName as parameter, startDate, completionDate, examDate, isRegistered to true and gives a suitable message showing variables used if this.isRegistered is false, else it gives a suitable message indicating the course has already registered.
- remove()

  This method sets the courseLeader name as “” in the parent class along with instructorName, startingDate, completionDate and examDate as “” in this class and isRemoved is set to true and if isRemoved is true it gives a suitable message.
- display()

  This method displays the output from super class and if the course is already registered then instructorName, startDate, completionDate and examDate is also shown with a suitable message

5. Testing Part

Test 1

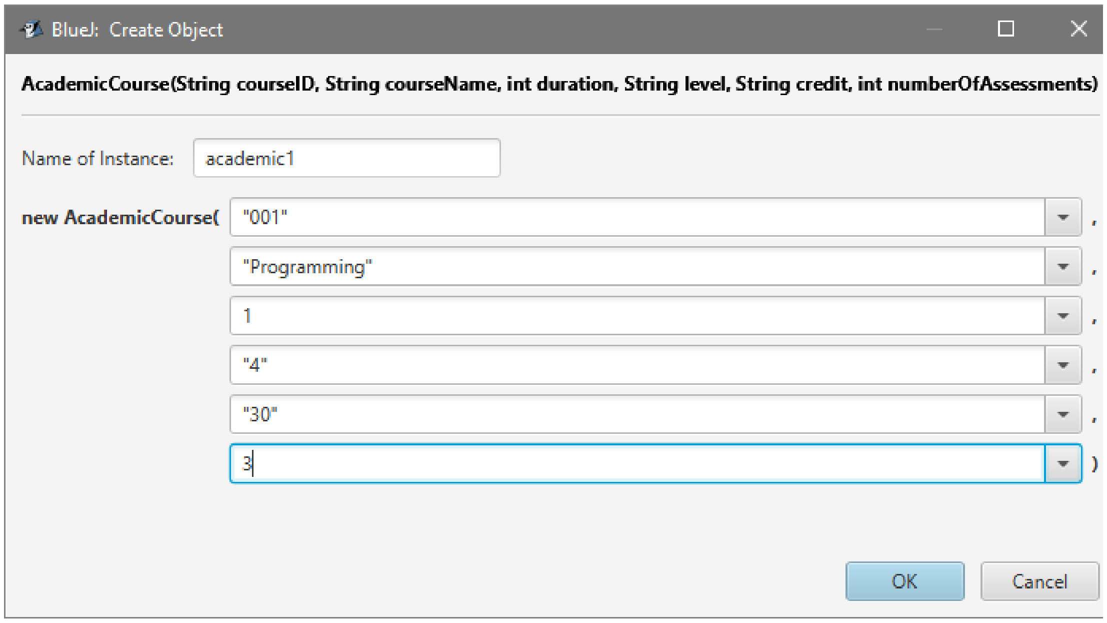
Test No:	1
Objective:	To inspect AcademicCourse class, register AcademicCourse and re-inspect the AcademicCourse class.

Action:	>> The AcademicCourse is called with the following arguments:
	courseID = “001”
	courseName = “Programming”
	duration = 1
	level = “4”
	credit = “30”
	numberOfAssessments = 3
	>> Inspection of the AcademicCourse class.
	>> void register is called with the following arguments:
	courseLeader = “Dhurba Sen”
	lecturerName = “Prabodh Tuladhar”
	startingDate = “8 March 2021”
	completionDate = “1 March 2022”
	>> Re-inspection of the lecturer class.

Expected Result:	The AcademicCourse would be registered.
Actual Result:	The AcademicCourse was registered.
Conclusion:	The test is successful.

Table 1: To inspect AcademicCourse class, register AcademicCourse and re-inspect the AcademicCourse class

Output results:

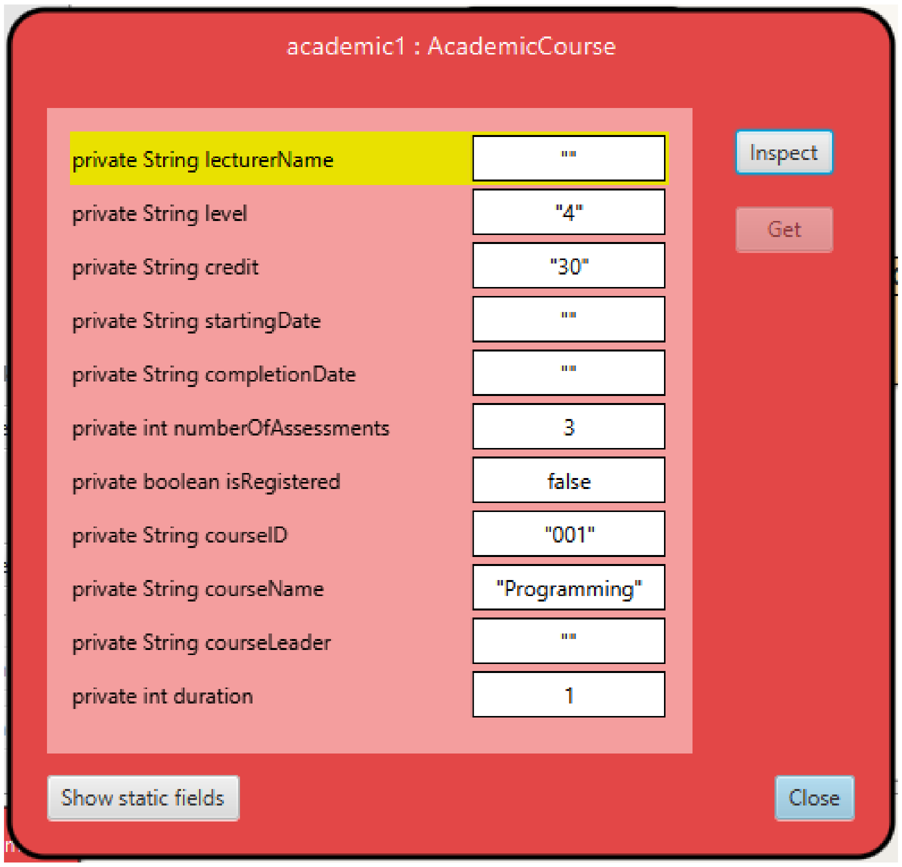

BlueJ: Create Object

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

Name of Instance: academic1

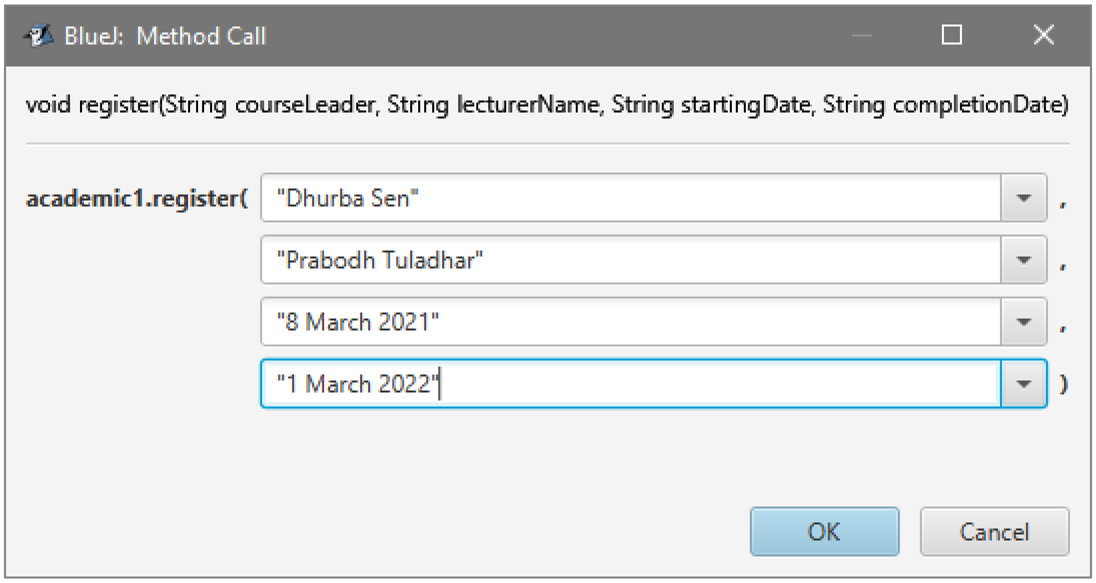
new AcademicCourse("001" ,
"Programming" ,
1 ,
"4" ,
"30" ,
3)

Figure 6: Screenshot of assigning data in AcademicCourse class


academic1 : AcademicCourse

private String lecturerName	""
private String level	"4"
private String credit	"30"
private String startingDate	""
private String completionDate	""
private int numberOfAssessments	3
private boolean isRegistered	false
private String courseID	"001"
private String courseName	"Programming"
private String courseLeader	""
private int duration	1

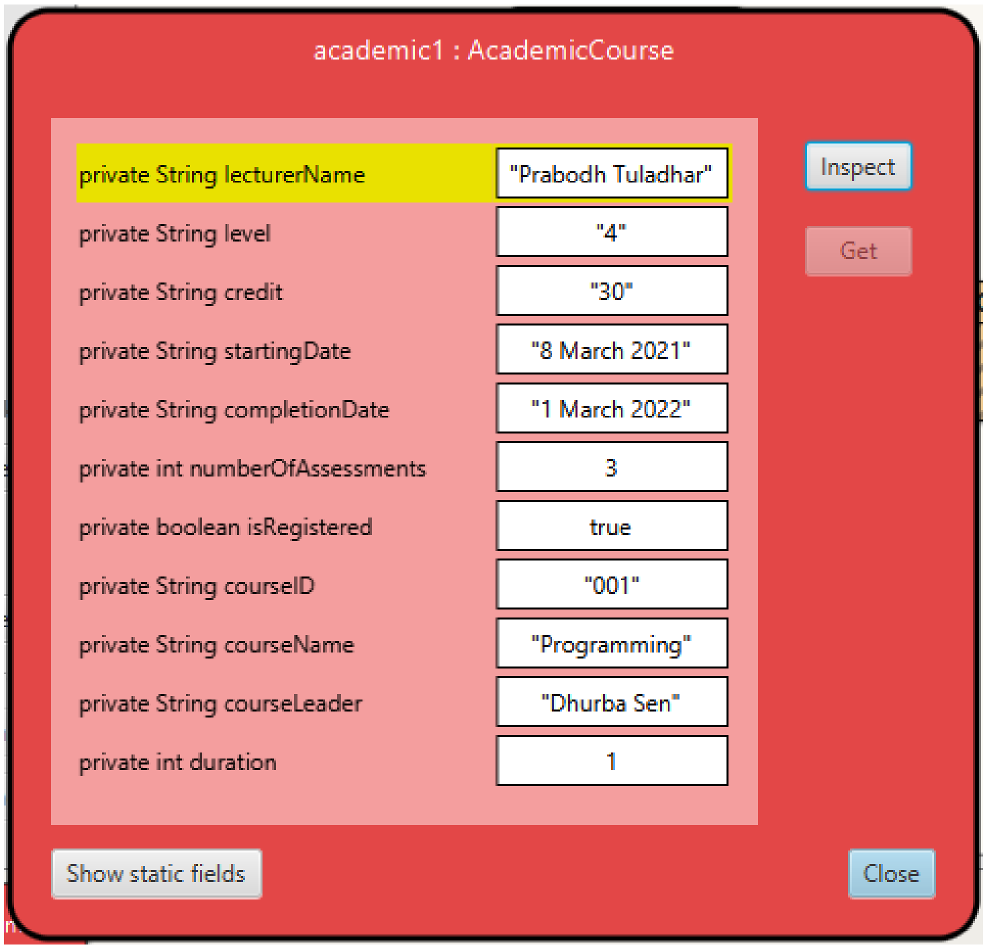
Figure 7: Screenshot for inspection of AcademicCourse class



void register(String courseLeader, String lecturerName, String startingDate, String completionDate)

academic1.register("Dhurba Sen" ,
"Prabodh Tuladhar" ,
"8 March 2021" ,
"1 March 2022")

Figure 8:Screenshot of assigning data in void register of AcademicCourse class



private String lecturerName	"Prabodh Tuladhar"
private String level	"4"
private String credit	"30"
private String startingDate	"8 March 2021"
private String completionDate	"1 March 2022"
private int numberOfAssessments	3
private boolean isRegistered	true
private String courseID	"001"
private String courseName	"Programming"
private String courseLeader	"Dhurba Sen"
private int duration	1

Figure 9: Screenshot for the re-inspection of AcademicCourse class

Test 2

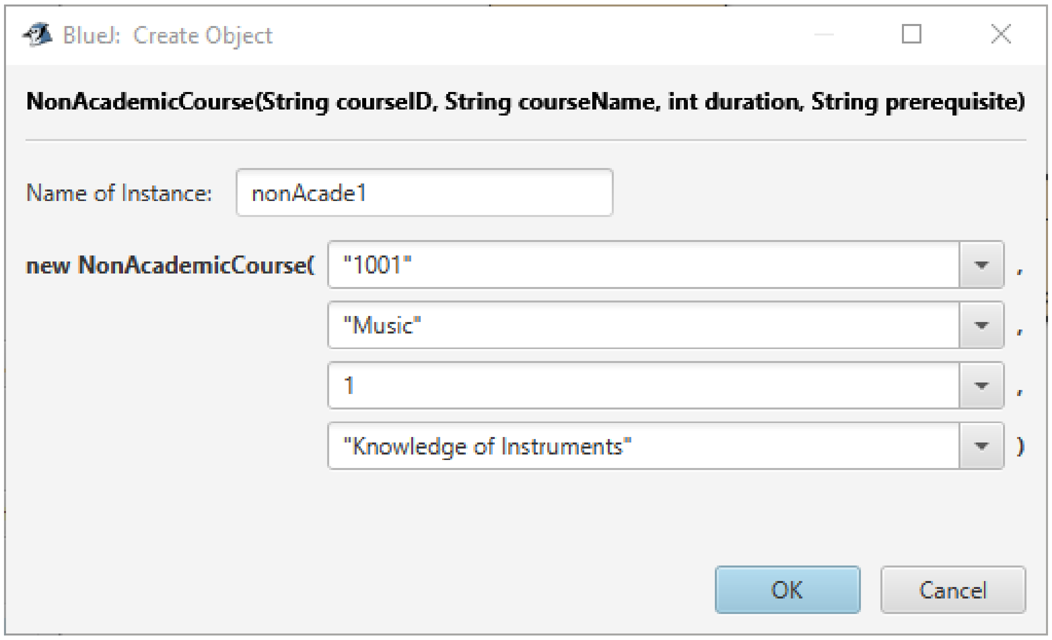
Test No:	2
Objective:	To inspect NonAcademicCourse class, register NonAcademicCourse and re-inspect the NonAcademicCourse class.

Action:	>> The NonAcademicCourse is called with the following arguments:
	courseID = “1001”
	courseName = “Music”
	duration = 1
	prerequisite = “Knowledge of Instruments”
	>> Inspection of the NonAcademicCourse class.
	>> void register is called with the following arguments:
	courseLeader = “Pramod Kharel”
	instructorName = “Raju Lama”
	startingDate = “15 May 2021”
	completionDate = “12 April 2022”
	examDate = “20 April 2022”
	>> Re-inspection of the lecturer class.

Expected Result:	The NonAcademicCourse would be registered.
Actual Result:	The NonAcademicCourse was registered.
Conclusion:	The test is successful.

Table 2: To inspect NonAcademicCourse class, register NonAcademicCourse and re-inspect the NonAcademicCourse class

Output results:

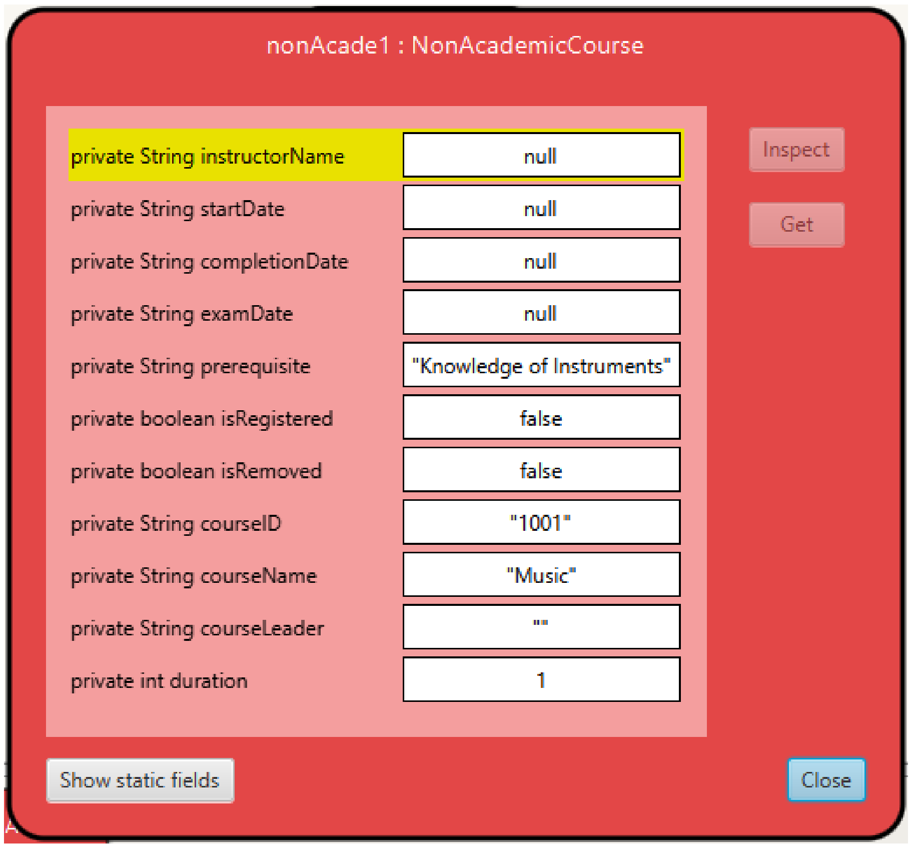


NonAcademicCourse(String courseID, String courseName, int duration, String prerequisite)

Name of Instance: nonAcade1

new NonAcademicCourse("1001" ,
"Music" ,
1 ,
"Knowledge of Instruments")

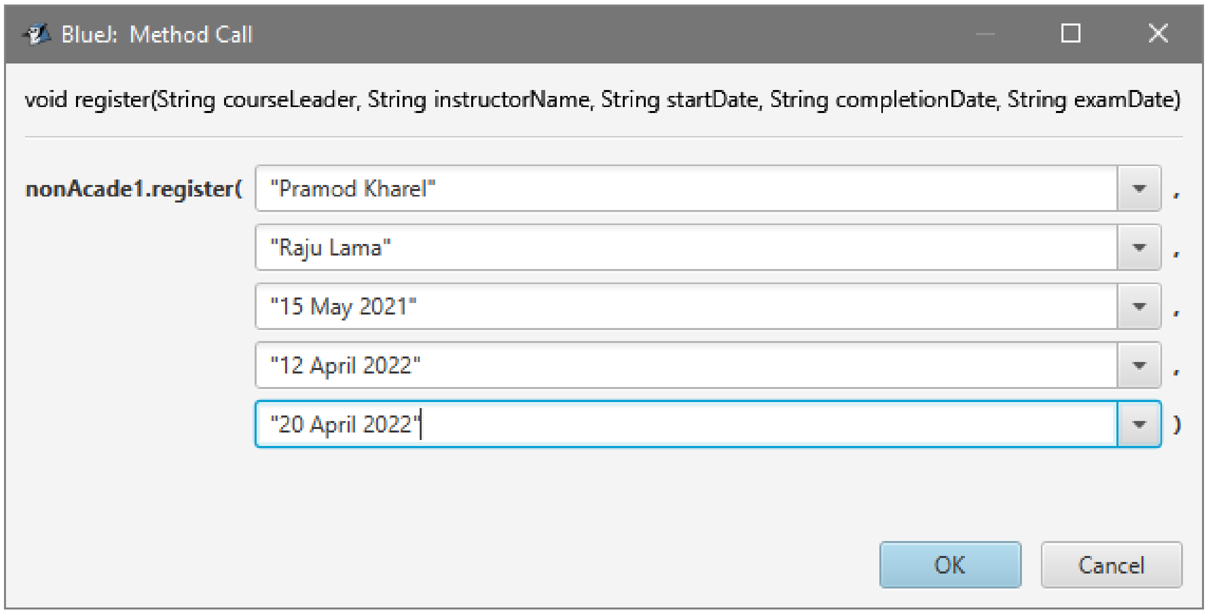
Figure 10: Screenshot of assigning data in NonAcademicCourse class:



nonAcade1 : NonAcademicCourse

private String instructorName	null
private String startDate	null
private String completionDate	null
private String examDate	null
private String prerequisite	"Knowledge of Instruments"
private boolean isRegistered	false
private boolean isRemoved	false
private String courseID	"1001"
private String courseName	"Music"
private String courseLeader	""
private int duration	1

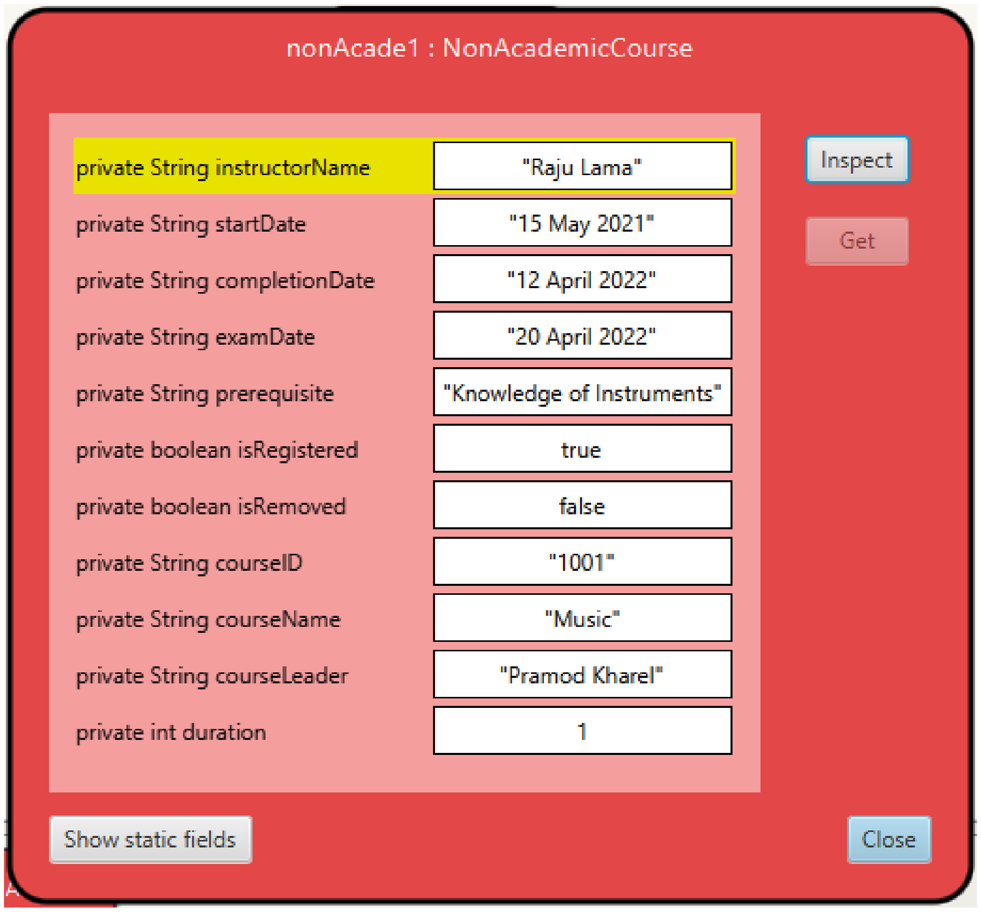
Figure 11: Screenshot for inspection of NonAcademicCourse class



void register(String courseLeader, String instructorName, String startDate, String completionDate, String examDate)

nonAcade1.register("Pramod Kharel" ,
"Raju Lama" ,
"15 May 2021" ,
"12 April 2022" ,
"20 April 2022")

Figure 12: Screenshot of assigning data in void register of NonAcademicCourse class



private String instructorName	"Raju Lama"
private String startDate	"15 May 2021"
private String completionDate	"12 April 2022"
private String examDate	"20 April 2022"
private String prerequisite	"Knowledge of Instruments"
private boolean isRegistered	true
private boolean isRemoved	false
private String courseID	"1001"
private String courseName	"Music"
private String courseLeader	"Pramod Kharel"
private int duration	1

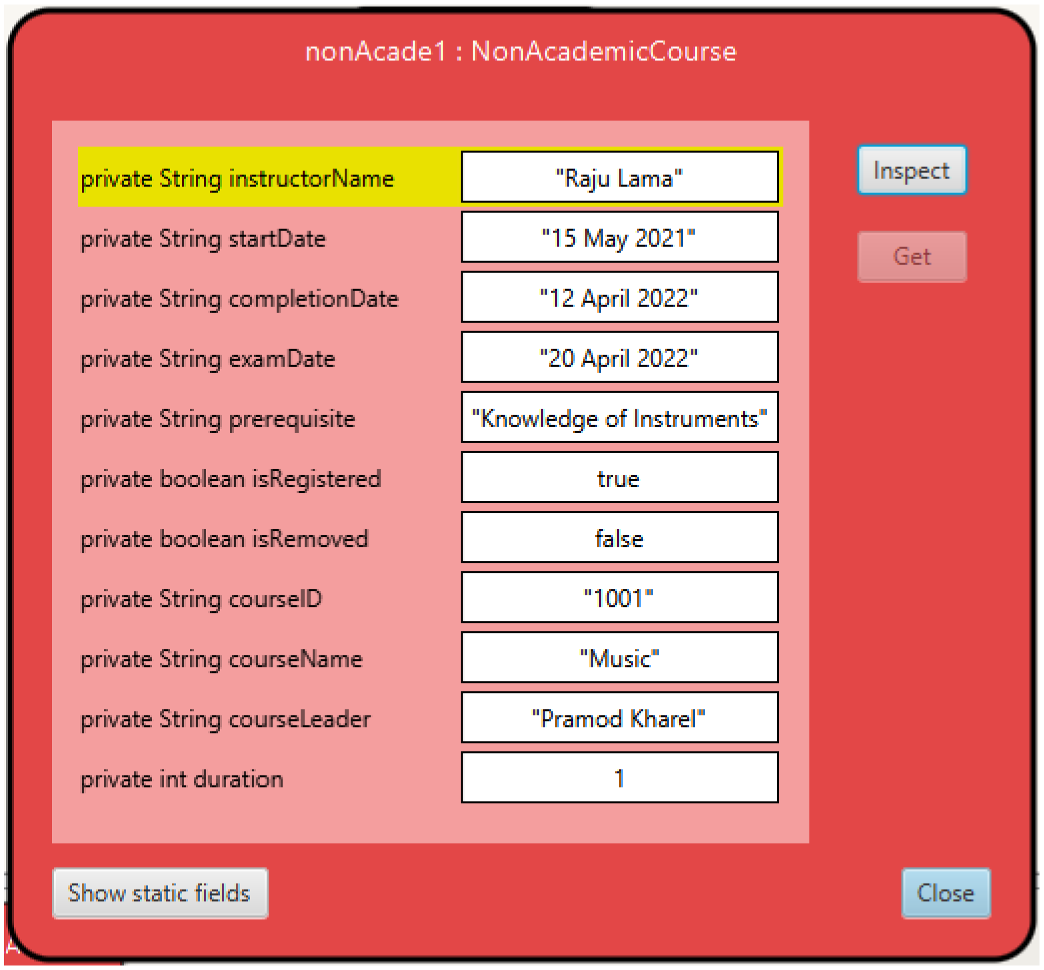
Figure 13: Screenshot for the re-inspection of NonAcademicCourse class

Test 3

Test No:	3
Objective:	To inspect NonAcademicCourse class, change the status of isRemoved to true and re-inspect the NonAcademicCourse class.
Action:	>> The NonAcademicCourse is called with the previous arguments. >> Inspection of the NonAcademicCourse class. >> void remove is called >> Re-inspection of the lecturer class.
Expected Result:	The NonAcademicCourse would be removed.
Actual Result:	The NonAcademicCourse was removed.
Conclusion:	The test is successful.

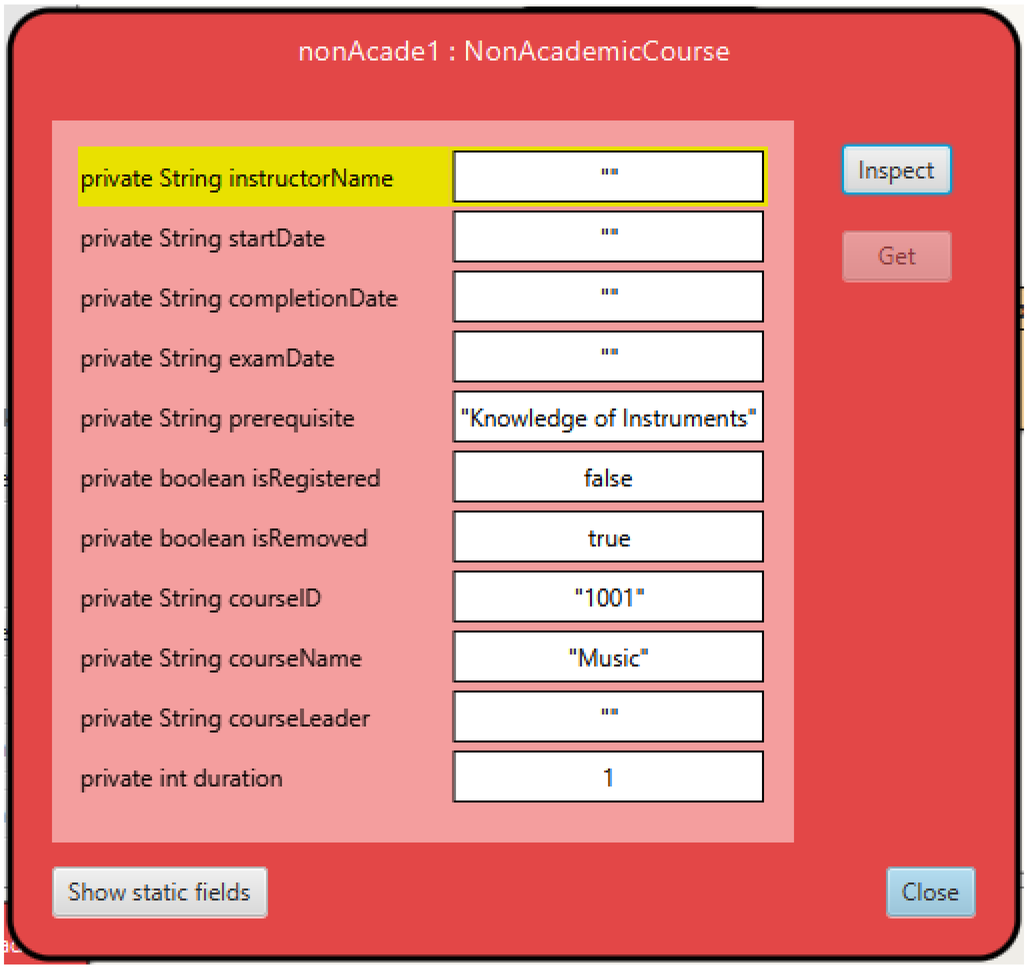
Table 3: To inspect NonAcademicCourse class, change the status of isRemoved to true and re-inspect the NonAcademicCourse class

Output results:



private String instructorName	"Raju Lama"
private String startDate	"15 May 2021"
private String completionDate	"12 April 2022"
private String examDate	"20 April 2022"
private String prerequisite	"Knowledge of Instruments"
private boolean isRegistered	true
private boolean isRemoved	false
private String courseID	"1001"
private String courseName	"Music"
private String courseLeader	"Pramod Kharel"
private int duration	1

Figure 14: Screenshot for inspection of NonAcademicCourse class


nonAcade1 : NonAcademicCourse

private String instructorName	""
private String startDate	""
private String completionDate	""
private String examDate	""
private String prerequisite	"Knowledge of Instruments"
private boolean isRegistered	false
private boolean isRemoved	true
private String courseID	"1001"
private String courseName	"Music"
private String courseLeader	""
private int duration	1

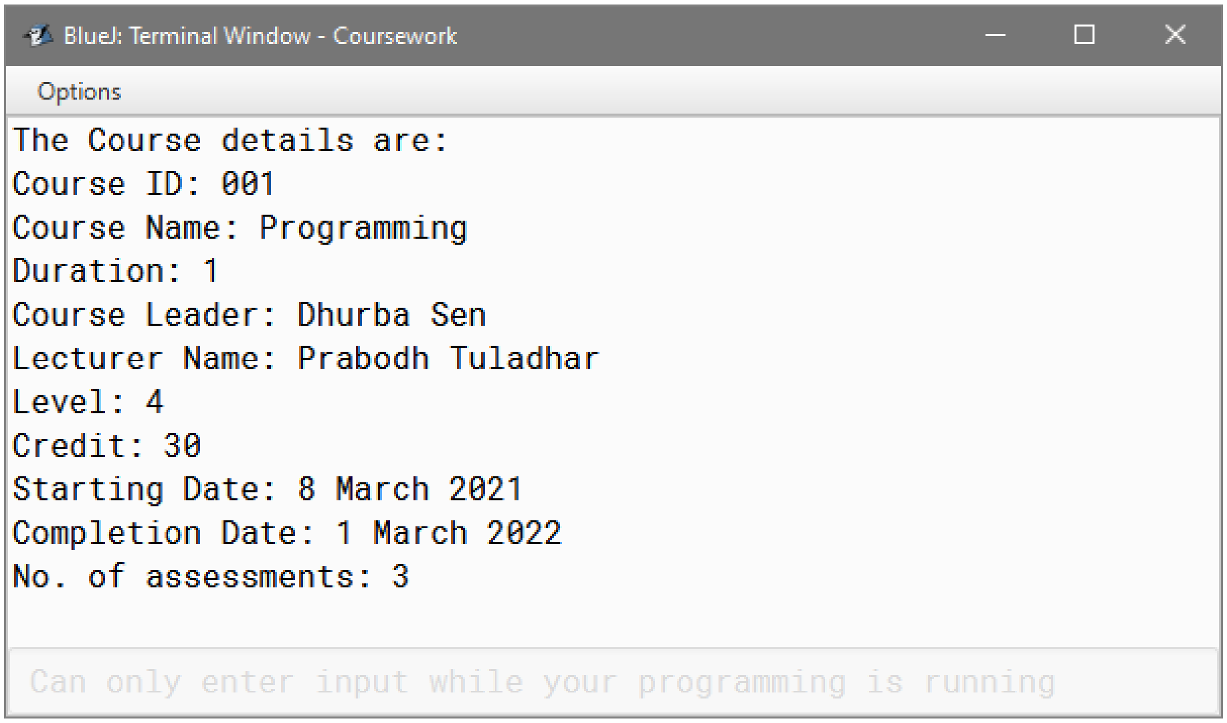
Figure 15: Screenshot for the re-inspection of NonAcademicCourse class

Test 4

Test No:	4
Objective:	To display the details of AcademicCourse and NonAcademicCourse classes.
Action:	>> Display the details of both AcademicCourse and NonAcademicCourse
Expected Result:	The details of AcademicCourse and NonAcademicCourse would be displayed.
Actual Result:	The details of both AcademicCourse and NonAcademicCourse was displayed.
Conclusion:	The test is successful.

Table 4: To display the details of AcademicCourse and NonAcademicCourse classes

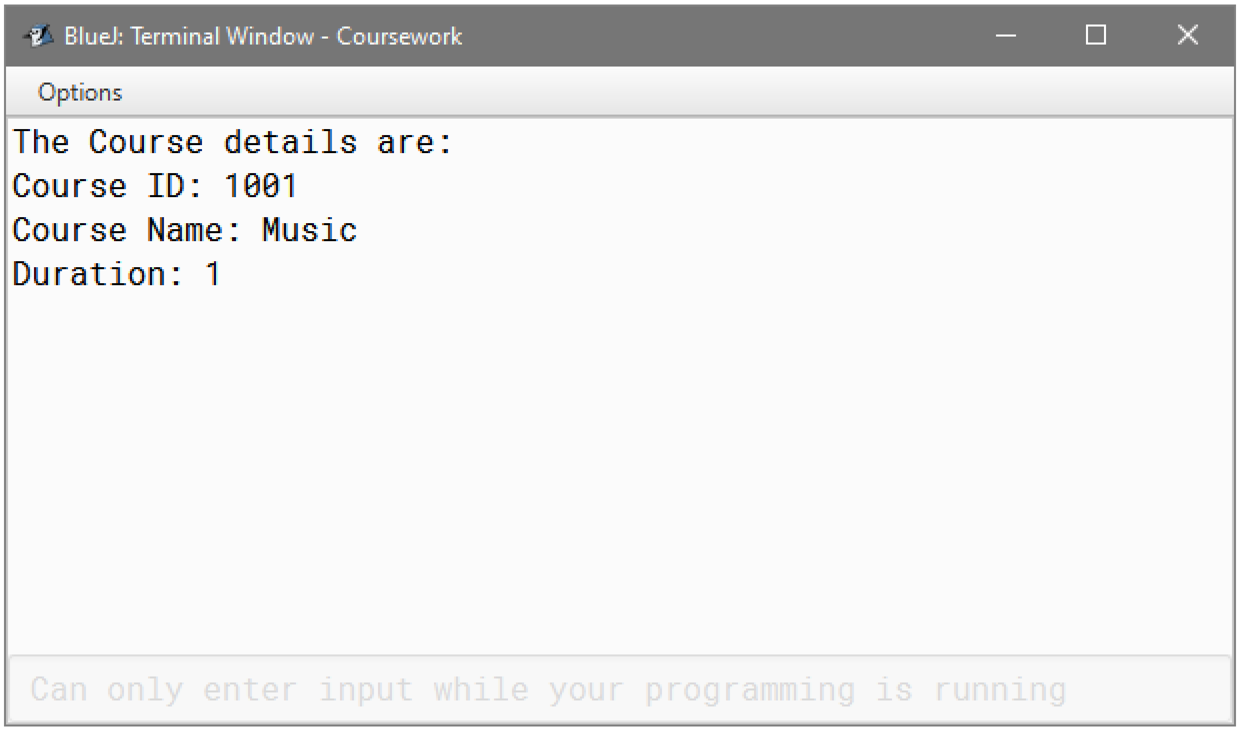
Output results:


BlueJ: Terminal Window - Coursework
Options

```
The Course details are:
Course ID: 001
Course Name: Programming
Duration: 1
Course Leader: Dhurba Sen
Lecturer Name: Prabodh Tuladhar
Level: 4
Credit: 30
Starting Date: 8 March 2021
Completion Date: 1 March 2022
No. of assessments: 3
```

Can only enter input while your programming is running

Figure 16: Display AcademicCourse


BlueJ: Terminal Window - Coursework
Options

```
The Course details are:
Course ID: 1001
Course Name: Music
Duration: 1
```

Can only enter input while your programming is running

Figure 17: Display NonAcademicCourse

6. Error Detection

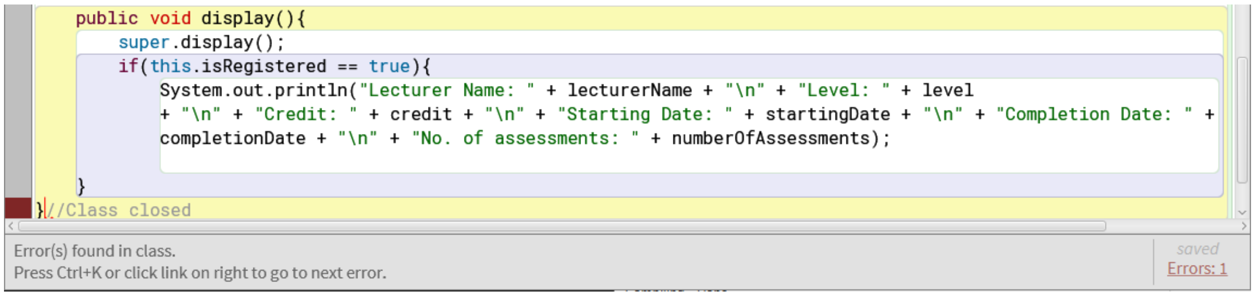
There were various errors that were detected during coursework while coding in bluej. These problems were solved by observing the nature of the problem in detail. The various types of errors that arose while coding are as follows:

6.1 Syntax Error

A syntax error is an error that occurs when the arrangement of structure in the source code of a program is incorrect. The computer programs must follow the correct structure or syntax in order to compile successfully. Any portion of the code which does not conform to the syntax of a programming language produces a syntax error. (Christensson, 2012)

One of the syntax errors detected while compiling the program is given below:

A curly bracket was missing at the end of the method.



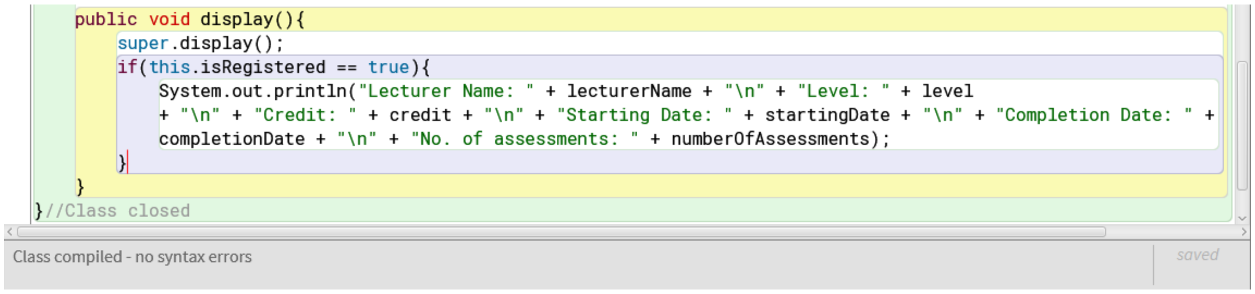
```
public void display(){
    super.display();
    if(this.isRegistered == true){
        System.out.println("Lecturer Name: " + lecturerName + "\n" + "Level: " + level
        + "\n" + "Credit: " + credit + "\n" + "Starting Date: " + startingDate + "\n" + "Completion Date: " +
        completionDate + "\n" + "No. of assessments: " + numberOfAssessments);

    }
}//Class closed
```

```
Error(s) found in class.
Press Ctrl+K or click link on right to go to next error.
saved
Errors: 1
```

Figure 18: Syntax error in the program

The error was solved by closing the curly bracket in the required position.



```
public void display(){
    super.display();
    if(this.isRegistered == true){
        System.out.println("Lecturer Name: " + lecturerName + "\n" + "Level: " + level
        + "\n" + "Credit: " + credit + "\n" + "Starting Date: " + startingDate + "\n" + "Completion Date: " +
        completionDate + "\n" + "No. of assessments: " + numberOfAssessments);
    }
}
}//Class closed
```

```
Class compiled - no syntax errors
saved
```

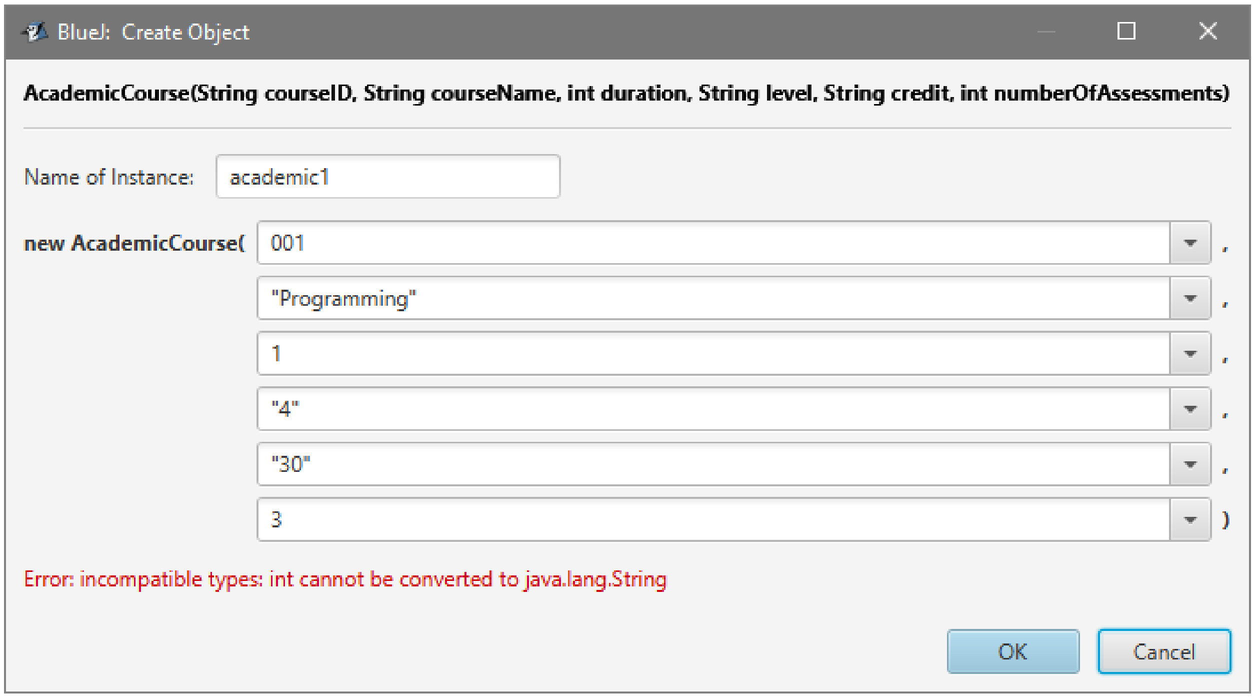
Figure 19: Correction of syntax error

6.2 Semantic Error

A semantic error is the error that occurs when wrong variable or operator is used, or when an operation is done in a wrong order. These types of errors are detected at the time of compilation. These types of errors are grammatically or syntactically correct so it is a little more difficult to find out than the syntax error. (Javatpoint, 2021)

One of the semantic errors detected during the compilation of the program is given below:

Integer values were assigned to a String type variable.



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

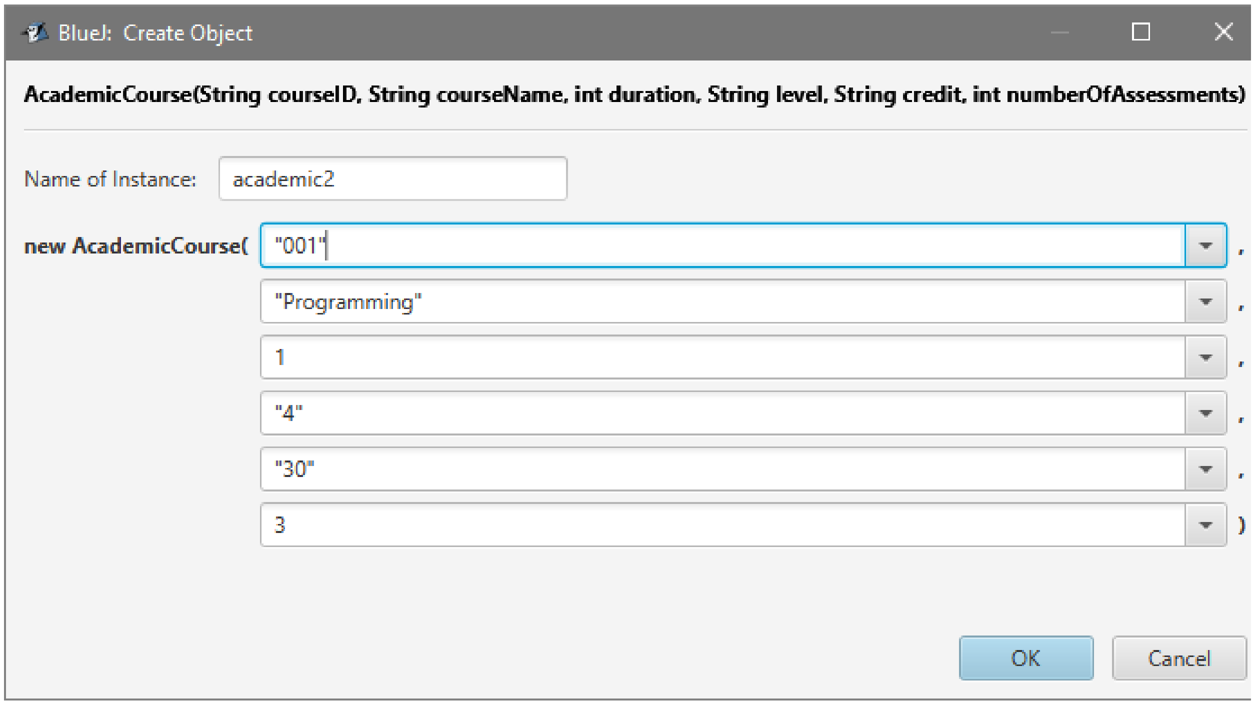
Name of Instance: academic1

```
new AcademicCourse( 001            ,
                    "Programming"  ,
                    1              ,
                    "4"            ,
                    "30"           ,
                    3              )
```

Error: incompatible types: int cannot be converted to java.lang.String

Figure 20: Semantic error in the program

The error was solved by placing a quotation mark in the required String position.


BlueJ: Create Object

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

Name of Instance: academic2

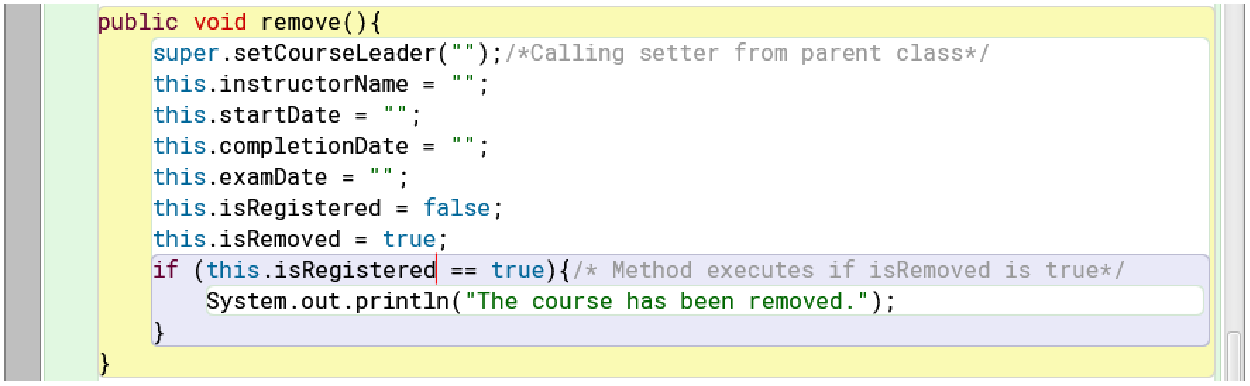
new AcademicCourse("001", "Programming", 1, "4", "30", 3)

Figure 21: Correction of semantic error

6.3 Logic Error

A logic or logical error is the error that occurs when a mistake in the programs source code causes incorrect or unexpected result. This type of error is the most difficult to find out than other types of errors as all the syntax and semantics are correct. This type of error generally occurs when a different operator is used, a typo is made or when a programmer misunderstands the required output to be something else. (Christensson, 2012)

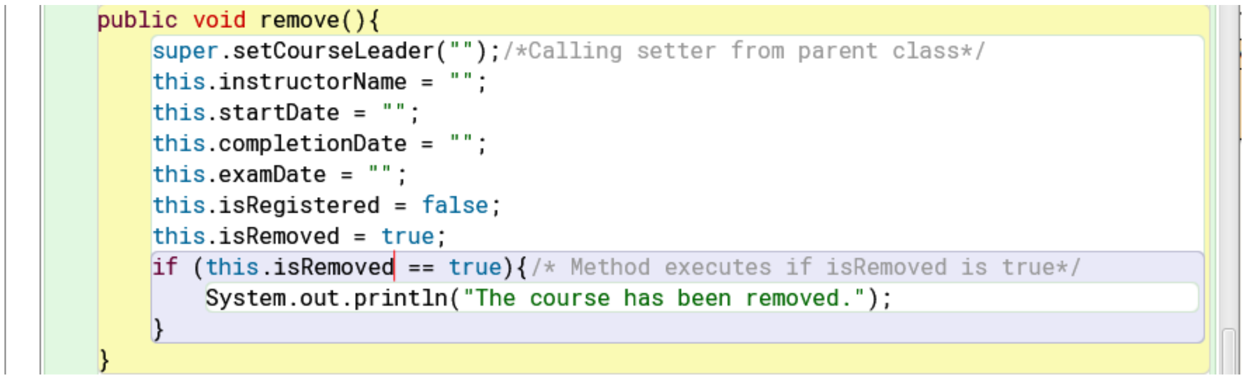
One of the logical errors detected after the compilation of the program is given below: Incorrect function is used in the given method.



```
public void remove(){
    super.setCourseLeader("");/*Calling setter from parent class*/
    this.instructorName = "";
    this.startDate = "";
    this.completionDate = "";
    this.examDate = "";
    this.isRegistered = false;
    this.isRemoved = true;
    if (this.isRegistered == true){/* Method executes if isRemoved is true*/
        System.out.println("The course has been removed.");
    }
}
```

Figure 22: Logic error in the program

The error was solved by using the correct function in the required conditional statement.



```
public void remove(){
    super.setCourseLeader("");/*Calling setter from parent class*/
    this.instructorName = "";
    this.startDate = "";
    this.completionDate = "";
    this.examDate = "";
    this.isRegistered = false;
    this.isRemoved = true;
    if (this.isRemoved == true){/* Method executes if isRemoved is true*/
        System.out.println("The course has been removed.");
    }
}
```

Figure 23: Correction of logic error

7. Conclusion

This is my first coursework for programming module. I was not familiar with a lot of the terms involved in this module. I learnt about most of them from various websites, books and the lecture class. This helped me gain some context on the vast subject of programming. Many times, I could not grasp a particular idea or concept. So, I asked my lecturers about those topics. They were very supportive and consistently guided me when I was confused in various aspects of the course. Even while developing this coursework, I made various errors in the coding methods and the program was not executing as required. A lot of it I learnt through trial and error and online research. There were specific activities to implement in this project which I could not find online and got stuck in them for a long time. In those times I contacted my lecturers through email messages, they articulated the nature of the problem and guided me in a way that was very easy for me to understand. In this way, I completed this project by taking references from various online sites, inquiring about particular issues with my lecturers and putting an effort to understand and implement various techniques required in the coursework.

References

Christensson, P., 2012. *Logic Error Definition.* [Online]
Available at: https://techterms.com/definition/logic_error
[Accessed 20 May 2021].

Christensson, P., 2012. *Syntax Error Definition.* [Online]
Available at: https://techterms.com/definition/syntax_error
[Accessed 20 May 2021].

Guru99, 2021. *Guru99.* [Online]
Available at: https://www.guru99.com/java-platform.html
[Accessed 17 May 2021].

Javatpoint, 2021. *Semantic Error.* [Online]
Available at: https://www.javatpoint.com/semantic-error
[Accessed 20 May 2021].

N K, R., 2021. *Java Tutorial by N K Raju.* [Online]
Available at: https://sites.google.com/site/javatutorialbynkraju/1-introduction/5-what-is-bluej
[Accessed 17 May 2021].

The Economic Times, 2021. *Software-Development.* [Online]
Available at: https://economictimes.indiatimes.com/definition/pseudocode
[Accessed 17 May 2021].

Tutorialspoint, 2021. *UML - Class Diagram.* [Online]
Available at: https://www.tutorialspoint.com/uml/uml_class_diagram.htm
[Accessed 17 May 2021].

Appendix

Course Class

```
/**Course class is a parent of the AcademicCourse and NonAcademicCourse class
 * It's various methods can be called from the child classes.

 */public class Course
{
    //Instance Variable Declaration
    private String courseID;
    private String courseName;
    private String courseLeader;
    private int duration;

    //Creating a parameterized constructor
    public Course(String courseID, String courseName, int duration){
        this.courseID = courseID;
        this.courseName = courseName;
        this.courseLeader = "";
        this.duration = duration;
        }

    //Assigning accessor methods to return and initialize the values of variables
    public String getCourseID()
    {
        return this.courseID;
    }

    public String getCourseName()
    {
        return this.courseName;
    }

    public String getCourseLeader()
    {
        return this.courseLeader;
    }

    public int getDuration()
    {
        return this.duration;
    }

    //Using mutator method to set the values of variables
    public void setCourseLeader(String courseLeader){
        this.courseLeader = courseLeader;
    }

    public void display(){
        if(this.courseLeader.equals("")){ /*method executes if the courseLeader is empty
String*/
            System.out.println("The Course details are:");
            System.out.println("Course ID: " + courseID + "\n" + "Course Name: " +
courseName + "\n" + "Duration: "
            + duration);
        }
        else{/*method executes when courseLeader is not empty*/
            System.out.println("The Course details are:");
            System.out.println("Course ID: " + courseID + "\n" + "Course Name: " +
courseName + "\n" + "Duration: "
            + duration + "\n" + "Course Leader: " + courseLeader);
        }
    }
}
```

AcademicCourse Class

```
/** The AcademicCourse class is a subclass of the Course class
 * It inherits various methods from the parent class

 */
public class AcademicCourse extends Course
{
    //Instance Variable Declaration
    private String lecturerName;
    private String level;
    private String credit;
    private String startingDate;
    private String completionDate;
    private int numberOfAssessments;
    private boolean isRegistered;

    //Creating a parameterzed constructor
    AcademicCourse(String courseID, String courseName, int duration, String level,
String credit,int numberOfAssessments) {
        super(courseID, courseName, duration);/*Calling Super Class Constructor*/
        this.lecturerName = "";
        this.level = level;
        this.credit = credit;
        this.startingDate = "";
        this.completionDate = "";
        this.numberOfAssessments = numberOfAssessments;
        this.isRegistered = false;
    }

    //Assigning accessor methods to return and initialize the values of variables
    public String getLecturerName()
    {
        return this.lecturerName;
    }

    public int getNumberOfAssessments()
    {
        return this.numberOfAssessments;
    }

    //Using mutator method to set the values of variables
    public void setLecturerName(String lecturer){
        this.lecturerName = lecturer;
    }

    public void setNumberOfAssessments(int assessments){
        this.numberOfAssessments = assessments;
    }

    //Method to register a course
    public void register(String courseLeader, String lecturerName, String startingDate,
String completionDate) {
        if(this.isRegistered == true){ /*method executes if the condtion is true*/
            System.out.println("The academic course has already been registered.");
            System.out.println("Lecturer name: " + lecturerName + "\n" + "Starting Date:  " +
startingDate + "\n"
```

```
            + "Completion Date: " + completionDate);
        }
        else{/*else this method is executed*/
            super.setCourseLeader(courseLeader);
            this.lecturerName = lecturerName;
            this.startingDate = startingDate;
            this.completionDate = completionDate;
            this.isRegistered = true;
            System.out.println("The academic course has been registered successfully.");
            System.out.println("Course Leader: " + courseLeader + "\n"+ "Lecturer Name: "
+ lecturerName + "\n" +
            "Starting Date: " + startingDate + "\n" + "Completion Date: " + completionDate);
        }
    }

    public void display(){
        super.display();/*Calling display method from parent class*/
        if(this.isRegistered == true){/*Method executes if the condition is true*/
            System.out.println("Lecturer Name: " + lecturerName + "\n" + "Level: " + level
            + "\n" + "Credit: " + credit + "\n" + "Starting Date: " + startingDate + "\n" +
"Completion Date: " +
            completionDate + "\n" + "No. of assessments: " + numberOfAssessments);
        }
    }
}
```

NonAcademicCourse Class

```
/** The NonAcademicCourse class is a subclass of the Course class
 * It inherits various methods from the parent class
 */
public class NonAcademicCourse extends Course
{
    //Instance Variable Declaration
    private String instructorName;
    private String startDate;
    private String completionDate;
    private String examDate;
    private String prerequisite;
    private boolean isRegistered;
    private boolean isRemoved;

    NonAcademicCourse(String courseID, String courseName, int duration, String
prerequisite) {
        super(courseID, courseName, duration);//super class constructor
        this.instructorName = instructorName;
        this.startDate = startDate;
        this.completionDate = completionDate;
        this.examDate = examDate;
        this.prerequisite = prerequisite;
        this.isRegistered = false;
        this.isRemoved = false;
    }

    public String getInstructorName()
    {
        return this.instructorName;
    }

    public String getStartDate()
    {
        return this.startDate;
    }

    public String getCompletionDate()
    {
        return this.completionDate;
    }

    public String getExamDate()
    {
        return this.examDate;
    }

    public String getPrerequisite()
    {
        return this.prerequisite;
    }

    //Using mutator method to set the values of variables
    public void setInstructorName(String instructorName){
        if(this.isRegistered == false) {/*Method executes if the condition is false*/
            this.instructorName = instructorName;
        }
        else {/*Else this method gets executed*/
            System.out.println("The Course has already been registered. Instructor cannot
change");
        }
    }

    //Method to register a course
    public void register(String courseLeader, String instructorName, String startDate,
String completionDate, String examDate) {
        if(this.isRegistered == false) {/*Method eecutes if isRegistered is true*/
            super.setCourseLeader(courseLeader);
            setInstructorName(instructorName);
            this.startDate = startDate;
            this.completionDate = completionDate;
            this.examDate = examDate;
            isRegistered = true;
            System.out.println("The course has been registered.");
            System.out.println("Course leader: " + courseLeader + "\n" + "Instructor name: "
+ instructorName +"\n" +
            "Start Date: " + startDate  + "\n" + "Completion Date: " + completionDate + "\n" +
"Exam Date: " + examDate);
        }
        else{/*Else this method gets executed*/
            System.out.println("The course has already been registered.");
        }
    }

    //Method to remove a course
    public void remove(){
        super.setCourseLeader("");/*Calling setter from parent class*/
        this.instructorName = "";
        this.startDate = "";
        this.completionDate = "";
        this.examDate = "";
        this.isRegistered = false;
        this.isRemoved = true;
        if (this.isRemoved == true){/* Method executes if isRemoved is true*/
            System.out.println("The course has been removed.");
        }
    }

    public void display(){
        super.display();/*Calling display method from parent class*/
        if(this.isRegistered == true){/*Method executes if the condition is true*/
            System.out.println("Instructor Name: " + instructorName + "\n" + "Start Date: " +
startDate +
            "\n" + "Completion Date: " + completionDate + "\n" + "Exam Date: " +
examDate);
        }
    }
}
```