



Module Code & Module Title

CC4001NI Programming

COURSEWORK-1

Assessment Weightage & Type

30% Individual Coursework

Year and Semester

2021 Spring

Student Name : Punam Thapa Magar

Group : C3

London Met ID:20048968

College ID : NPO1CP4S210273

Assignment Due Date : 23-05-2021

Assignment Submission Date : 20-05-2021

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

Table of Contents

Introduction.....	4
Class Diagram	5
Class Diagram of Course, Academic Course and Non Academic Course	6
Pseudocode	10
Course Class.....	10
Academic Course Class	12
Non-Academic Course	15
Method Description	18
Testing	22
Error Detection.....	32
Syntax Error	32
Semantic Error	33
Logical Error.....	34
Conclusion of the Report.....	37
Bibliography	38
Appendix.....	39

Table of figures

Figure 1 Class Diagram of the classes in BlueJ.....	5
Figure 2: Screenshot of assigning the data in Academic Course class.....	23
Figure 3: Screenshot for the inspection of the Academic Course class	23
Figure 4: Screenshot for assigning the data in void register academic course	24
Figure 5: Screenshot for re-inspection of Academic Course class.....	24
Figure 6: Screenshot of assigning the data in Non-Academic class.....	26
Figure 7: Screenshot for the inspection of Non-Academic Course class	26
Figure 8: Screenshot for assigning the data in void register Non-Academic Course class....	27
Figure 9: Screenshot for re-inspection of Non-Academic Course class.....	27

Figure 10: Screenshot of displaying course is removed	29
Figure 11: Screenshot for the inspection of Non-Academic Course class.....	29
Figure 12: Displaying the details of the Academic Course class.....	30
Figure 13: Displaying the details of the Non-Academic Course class	31
Figure 14: Error due to missing semi colon.....	32
Figure 15: Error solved.....	32
Figure 16: Error due to incompatible types	33
Figure 17: Error solve	34
Figure 18: Logical error of the program	35
Figure 19: Displaying Non-Academic Course detail due to error	35
Figure 20: Error fix	36
Figure 21: Displaying Non-Academic Course detail with no error	36

Table of tables

Table 1 Class Diagram of Course class	7
Table 2 Class Diagram of Academic Course class.....	8
Table 3 Class Diagram of Non Academic Course class	9
Table 4: To inspect Academic Course class, register academic course and re-inspect the Academic Course class.....	22
Table 5: To inspect Non-Academic Course class, register non-academic course and re-inspect the non-academic course class.....	25
Table 6: To inspect the Non-Academic Course class again and isremoved to true,re-inspect the Non-Academic course class	28
Table 7: To display the details of Academic Course and Non-Academic Course classes.....	30

Introduction

This is the first coursework which was appointed to us from the module "Introduction to Programming". The program is done by utilizing BlueJ. The primary nature of this coursework is to make a class to address a course which comprise of two sub classes which are academic and non-academic course class respectively. The program comprise of specific methods, constructor and accessor techniques for every one of the attributes of all classes which permits us to employ and fire the courses.

In all sincerity, The program consists of three classes which are Course, Academic course, and Non-academic course exceptionally. The Course class is the parent class and the academic and non-academic are the sub classes or child classes. The Course class set the information about the course namely, course ID, course name, course leader and duration respectively where course ID and course name are each represented as a string data type and duration as int data type. Moreover, the course class displays certain given attributes like the course ID, course name, duration and the course leader. In the same way, the academic course class set the information about the academic course namely, number of assessments, lecturer name, level, credit, starting date, and completion date and boolean isRegistered which is initialized to false and the registered status of the course is changed to true and course removed status is initialized to false. Likewise, after the registration of the academic course the given following attributes are also to be displayed. Similarly, the non-academic course also takes the information and displays the information about the non-academic course class. A method is used to display the details of the non-academic course where there is the same signature as the display method in the course class which call the method in course class.

The accessor method and the mutator method are used in order to return the values and assign the new values. The parent class is called in order to display the information of the academic and non-academic course class. The constructor of the classes are also assigned with the parameters which are said to be accepted. Similarly, the attributes are also assigned with different values. Each of the attributes of all classes have the accessor method or the 'get method'. And the mutator method or the 'set method' is also used in some attributes to assign or set the new values.

Class Diagram

A class diagram is a type of diagram and part of a unified modeling language (UML) that defines and provides the overview and structure of a system in terms of classes, attributes and methods, and the relationships between different classes.

It is a type of structure diagram and looks similar to a flow chart having three main parts illustrated in rectangular boxes. The first or top part specifies the class name, the second or middle specifies attributes of that class and the third or bottom section lists the methods or operations that specific class can perform. (Anon., n.d.)

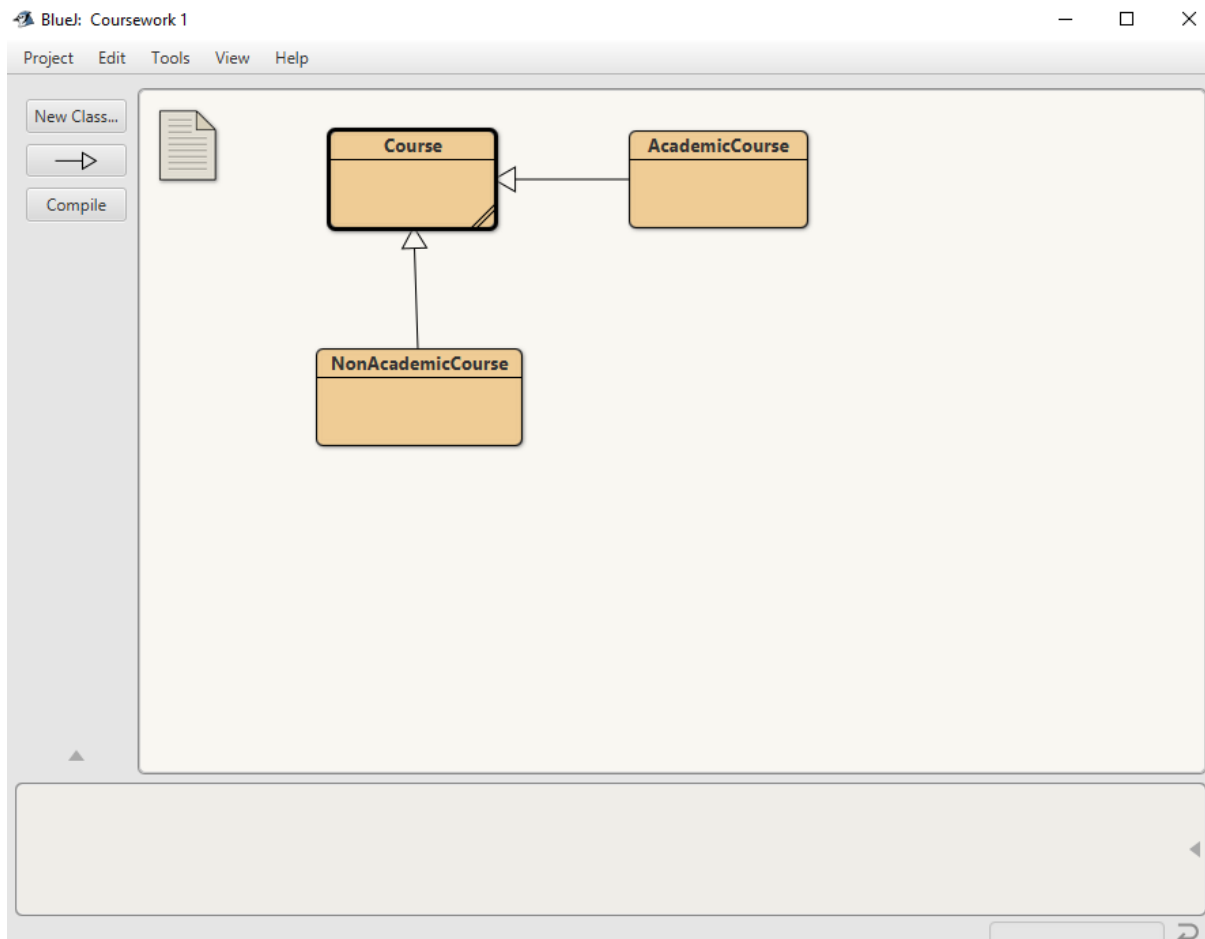
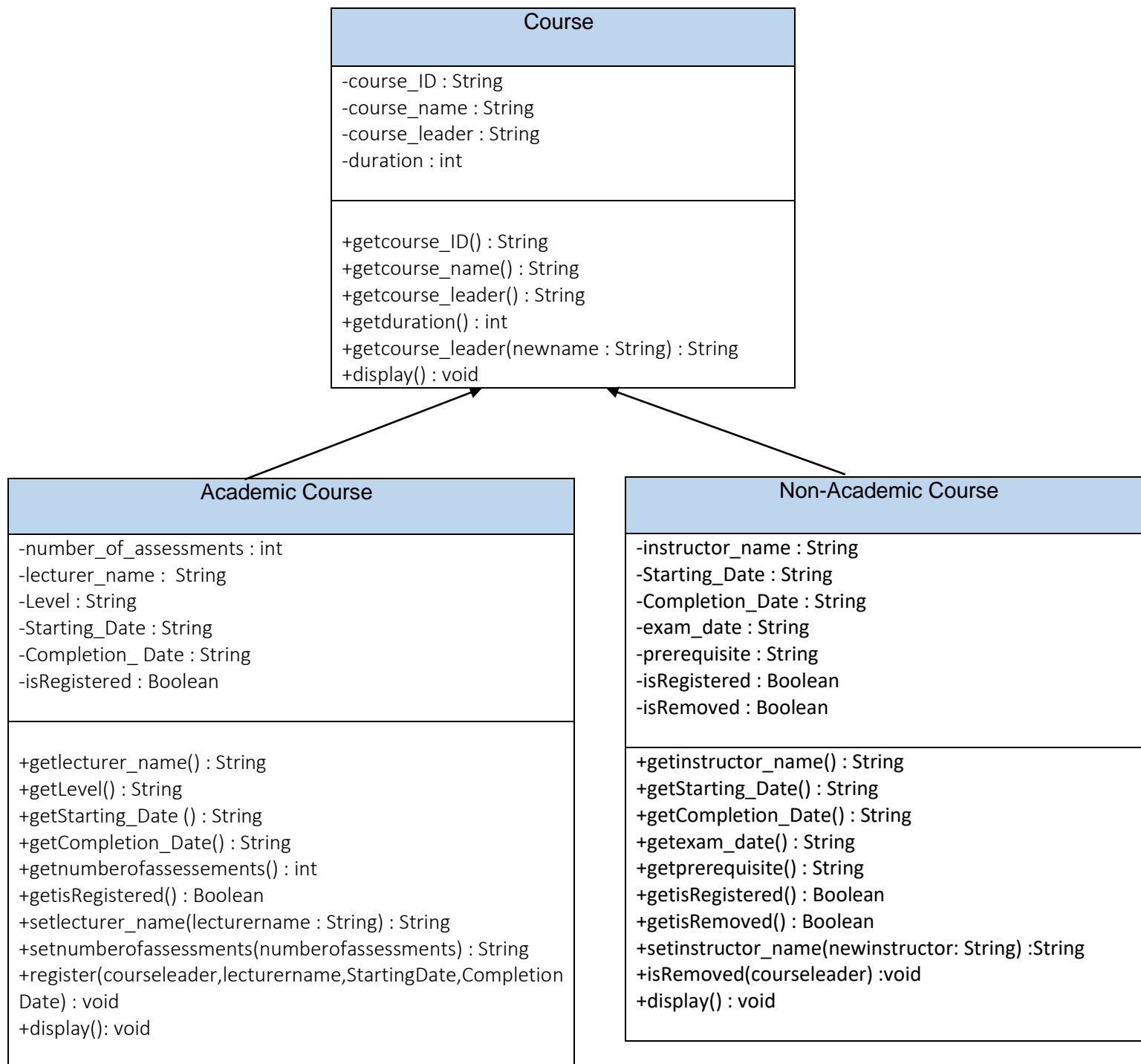


Figure 1 Class Diagram of the classes in BlueJ

Class Diagram of Course, Academic Course and Non Academic Course



The class diagram of three classes of the program of java are given below:

- Course Class

Course
-course_ID : String -course_name : String -course_leader : String -duration : int
+getcourse_ID() : String +getcourse_name() : String +getcourse_leader() : String +getduration() : int +setcourse_leader(newname : String) : String +display() : void

Table 1 Class Diagram of Course class

- Academic Course Class

Academic Course
-number_of_assessments : int -lecturer_name : String -Level : String -Starting_Date : String -Completion_Date : String -isRegistered : Boolean
+getlecturer_name() : String +getLevel() : String +getStarting_Date () : String +getCompletion_Date() : String +getnumber_of_assessments() : int +getisRegistered() : Boolean +setlecturer_name(lecturername : String) : String +setnumberofassessments(numberofassessments) : String +Register(courseleader,lecturername,StartingDate,CompletionDate) : void +display() : void

Table 2 Class Diagram of Academic Course class

- Non- Academic Course

Non Academic Course
-instructor_name : String -Starting_Date : String -Completion_Date : String -exam_date : String -prerequisite : String -isRegistered : Boolean -isRemoved : Boolean
+getinstructor_name() : String +getStarting_Date() : String +getCompletion_Date() : String +getexam_date() : String +getprerequisite() : String +getisRegistered() : Boolean +getisRemoved() : Boolean +setinstructorname(newinstructor : String) : String +isRemoved(courseleader) : void +display() : void

Table 3 Class Diagram of Non Academic Course class

Pseudocode

Pseudocode is an informal way of programming description that does not require any strict programming language syntax or underlying technology considerations.

Pseudocode is not an actual programming language. So it cannot be compiled into an executable program. It uses short terms or simple English language syntaxes to write code for programs before it is actually converted into a specific programming language. This is done to identify top level flow errors, and understand the programming data flows that the final program is going to use. This definitely helps save time during actual programming as conceptual errors have been already corrected. Firstly, program description and functionality is gathered and then pseudocode is used to create statements to achieve the required results for a program. Detailed pseudocode is inspected and verified by the designer's team or programmers to match design specifications. Catching errors or wrong program flow at the pseudocode stage is beneficial for development as it is less costly than catching them later. Once the pseudocode is accepted by the team, it is rewritten using the vocabulary and syntax of a programming language. The purpose of using pseudocode is an efficient key principle of an algorithm. It is used in planning an algorithm with sketching out the structure of the program before the actual coding takes place. (Anon., n.d.)

Course Class

CREATE class Course

DECLARE instance variable courseID with String Data type.

DECLARE instance variable coursename with String Data type.

DECLARE instance variable courseleader with String Data type.

DECLARE instance variable duration with int type.

CREATE constructor Course (PASS parameters courseID of String type, coursename of String type, courseleader of string type, duration of int type)

DO

ASSIGN parameter courseID to instance variable
courseID

ASSIGN parameter coursename to instance variable
coursename

ASSIGN parameter courseleader to empty string("")

ASSIGN parameter duration to instance variable duration

ENDDO

CREATE method getcourseID() with return type

DO

RETURN courseID

ENDDO

CREATE method getcoursename() with return type

DO

RETURN coursename

CREATE method getcourseleader() with return type

ENDDO

RETURN courseleader

CREATE method getduration() with return type

DO

RETURN duration

ENDDO

CREATE method setcourseleader (PASS parameter newname of String type)

DO

ASSIGN parameter courseleader to instance variable
newname

ENDDO

```
CREATE method display()
    DO
        PRINT courseID
        PRINT coursename
        PRINT duration
        If (courseleader! = "")
            PRINT courseleader
        ENDIF
    ENDDO
```

Academic Course Class

```
CREATE class Academic Course
    DECLARE instance variable number of assessments with int
    Data type
    DECLARE instance variable lecturername with String Data type
    DECLARE instance variable level with String Data type
    DECLARE instance variable credit with String Data type
    DECLARE instance variable StartingDate with String Data type
    DECLARE instance variable CompletionDate with String Data
    type
    DECLARE isRegistered with Boolean
    CREATE constructor AcademicCourse (PASS parameters level
    of String type, credit of String type, courseID of String type,
    CompletionDate of String type)
```

DO

ASSIGN parameter level to instance variable level

ASSIGN parameter credit to instance variable
credit

ASSIGN parameter number of assessment to
instance variable number of assessment

ASSIGN lecturername to empty String ("")

ASSIGN startingdate to empty String ("")

ASSIGN completiondate to empty String ("")

ASSIGN isRegistered to false

ENDDO

CREATE method getLecturername() with return type String

DO

RETURN lecturername

ENDDO

CREATE method getLevel() with return type String

DO

RETURN Level

ENDDO

CREATE method getCredit() with return type String

DO

RETURN credit

ENDDO

CREATE method getStartingDate() with return type String

```
DO
    RETURN StartingDate
ENDDO
```

CREATE method getCompletionDate() with return type String

```
DO
    RETURN CompletionDate
ENDDO
```

CREATE method getnumberofassessment() with return type int

```
DO
    RETURN numberofassessments
ENDDO
```

CREATE method getisRegistered with return type Boolean

```
DO
    RETURN boolean
ENDDO
```

CREATE method setlecturername(PASS parameter lecturername of String type)

```
DO
    ASSIGN parameter lecturername to instance
    variable lecturername
ENDDO
```

CREATE method setnumberofassessments(PASS parameter numberofassessments of int type)

```
DO
    ASSIGN parameter numberofassessments to
    instance variable numberofassessments
```

```
                ENDDO

CREATE method display()

    DO

        CALL display()from parent class

        IF (isRegistered==true)

            PRINT lecturername

            PRINT Level

            PRINT credit

            PRINT CompletionDate

            PRINT numberofassessments

        ENDIF

    ENDDO
```

Non-Academic Course

CREATE class Non-Academic Course

```
    DECLARE instance variable instructorname with String
    Data type

    DECLARE instance variable startingdate with String Data

    DECLARE instance variable completiondate with String
    Data type

    DECLARE instance variable examdate with String Data
    type
```

DECLARE instance variable prerequisite with String Data type

DECLARE isRegistered with Boolean

DECLARE isRemoved with Boolean

CREATE constructor Non-Academic Course (PASS parameters courseID of String type, coursename of String type, duration of int type, prerequisite of String type)

DO

ASSIGN parameter prerequisite to instance variable prerequisite

ASSIGN startingdate to empty String ("")

ASSIGN examdate to empty String("")

ASSIGN isRegistered is false

ASSIGN isRemoved is false

ENDDO

CREATE method getinstructorname() with return type String

DO

RETURN instructorname

ENDDO

CREATE method getstartingdate() with return type String

DO

RETURN startingdate

ENDDO

CREATE method getcompletiondate() with return type String

DO

RETURN completiondate

ENDDO

CREATE method getexamdate() with return type String

DO

RETURN examdate

ENDDO

CREATE method getprerequisite() with return type String

DO

RETURN prerequisite

ENDDO

CREATE method getisRegistered() with return type Boolean

DO

RETURN isRegistered

ENDDO

CREATE method getisRemoved() with return type Boolean

DO

RETURN isRemoved

ENDDO

CREATE method setinstructorname (PASS parameter newinstructor of String type)

DO

ASSIGN parameter instructorname to instance
variable newinstructor

ENDDO

CREATE method display()

DO

CALL display() from parent class

IF (isRegistered==true)

PRINT instructorname

PRINT startingdate

PRINT completiondate

PRINT examdate

ENDIF

ENDDO

Method Description

In this program different methods has been used. We have three classes which have used different methods. Different methods of the parent class have been used in the child class.

❖ Course Class

Different methods used in the Course class are given below:

- `getcourseID():`

This is the getter method that returns the value of the `courseID`.

- `getcoursename():`

This accessor method returns the value of the `course_name`.

- `getcourselider():`

This accessor method returns the value of the `course_leader`.

- `getduration()`:
This accessor method returns the value of the duration.
- `setcourseleader(String newname)`:
This accessor method is used to change the value of the `course_leader` through parameter.
- `Display()`:
This method displays the details of the Course.

❖ Academic Course

Some of the methods used in Academic Course class are mention below:

- `getlecturername()`:
This accessor method is used to return `lecturer_name`.
- `getLevel()`:
This accessor method is used to return Level.
- `getcredit()`:
This accessor method is used to return credit.
- `getStartingDate()`:
This accessor method is used to return `Starting_Date`.
- `getCompletionDate()`:
This accessor method is used to return `Completion_Date`.

- `getnumberofassessments()`:
This accessor method retruns `number_of_assessments`.
- `getisRegistered()`:
This accessor returns `isRegistered`.
- `setLecturername(String lecturername)`:
This method is used to change the value of the `lecturer_name` through parameter.
- `Setnumberofassessments(int numberofassessments)`:
This method is used to change the value of the `number_of_assessments` through parameter.
- `register()`:
If the academic course has been registered then the method will display `course_leader`, `Starting_Date`, `Lecturer_name`, `Completion_Date` else it will assign values to parameter.
- `Display()`:
It calls the display method from parent class and displays the details of academic course.

❖ Non-Academic Course

Some of the methods used in Non-Academic Course are given below:

- `getinstructorname()`:
This method returns the value of the `instructor_name`.
- `getstartingdate()`:
This method returns the value of `starting_date`.

- `Getcompletiondate()`:
This method returns `completion_date`.
- `getexamdate()`:
This method returns `exam_date`.
- `getprerequisite()`:
This method returns `prerequisite`.
- `getisRegistered()`:
This method returns `isRegistered`.
- `isRemoved()`:
This method returns `isRemoved`.
- `Setinstructorname(Stringnewinstructor)`:
This method is used to change the value of `new_instructor` through parameter but if the `instructor_name` `isRegistered` is set to `false` then the `instructor_name` cannot be changed.
- `register()`:
If non-academic course has been registered then the method will display `course_leader`, `course_name`, `duration` and `prerequisite` else it will assign values to parameter.
- `remove()`:
If non-academic course has been removed then the method will display `course_leader` else it will assign values to parameter.

Testing

Testing is the process of executing a program or application in-order to find its error and to perform correctly so that a software should be error-free. (Fatehi, 2021).

In simple word, testing is a strategy for discovering how well the program functions. In different terms, testing communicates what level of information or skill has been procured. All in all, testing is utilized at significant checkpoints in the general cycle to decide if objectives are being met.

Test 1: To inspect Academic Course class, register academic course and re-inspect the Academic Course class

Test No:	1
Objective:	To inspect Academic Course class, register academic course and re-inspect the Academic Course class.
Action:	<p>→The Academic Course class is called with the following arguments:</p> <p>Level= "3"</p> <p>credit= "30"</p> <p>course ID= "CS4001NI"</p> <p>Completion Date= "September 20,2021"</p> <p>course name= "Introduction to Programming"</p> <p>duration= 8</p> <p>→Inspection of the Academic Course class.</p> <p>→void register is called with the following arguments:</p> <p>Course leader= "Bishesh Shrestha"</p> <p>Lecturer name= "Roshan Sapkota"</p> <p>Starting Date= "February 4,2021"</p> <p>Completion Date= "September 20,2021"</p> <p>→Re-inspection of the Academic Course.</p>
Expected Result:	The academic course would be registered
Actual Result:	The academic Course was registered.
Conclusion:	The test is successful.

Table 4: To inspect Academic Course class, register academic course and re-inspect the Academic Course class

Output Result:

BlueJ: Create Object

AcademicCourse(String Level, String credit, String courseID, String CompletionDate, String coursename, int duration)

Name of Instance: academic2

new AcademicCourse("3" ,
"30" ,
"CS4001NI" ,
"September 20,2021" ,
"Introduction to Programming" ,
8)

OK Cancel

Figure 2: Screenshot of assigning the data in Academic Course class

academic2 : AcademicCourse

int numberOfassessments	0
String lectureName	""
String Level	"3"
String credit	"30"
String StartingDate	""
String CompletionDate	null
boolean isRegistered	false
String courseID	"CS4001NI"
String coursename	"Introduction to Programming"
String courseleader	""
int duration	8

Inspect Get Show static fields Close

Figure 3: Screenshot for the inspection of the Academic Course class

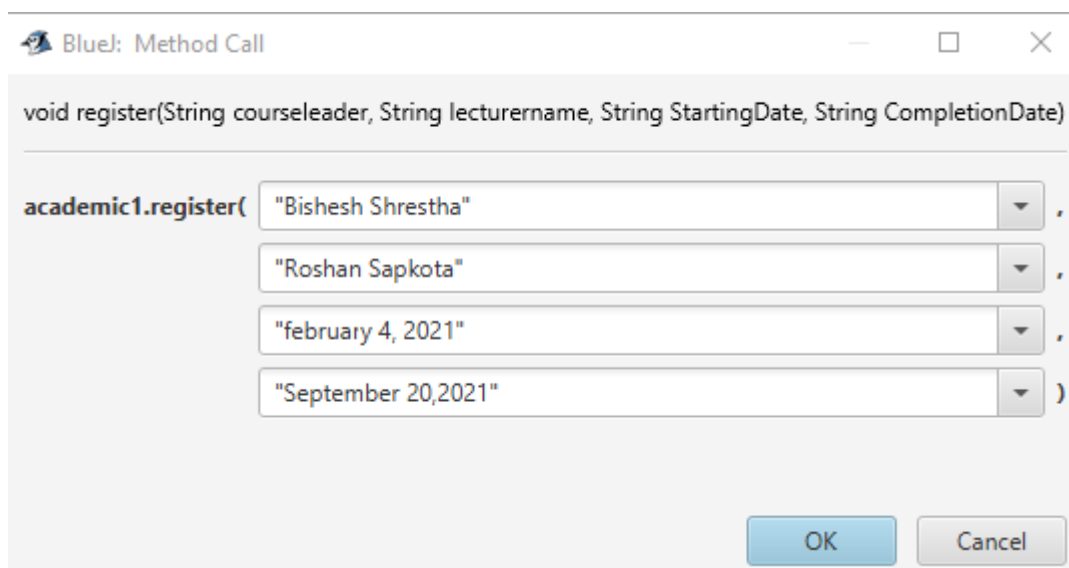


Figure 4: Screenshot for assigning the data in void register academic course

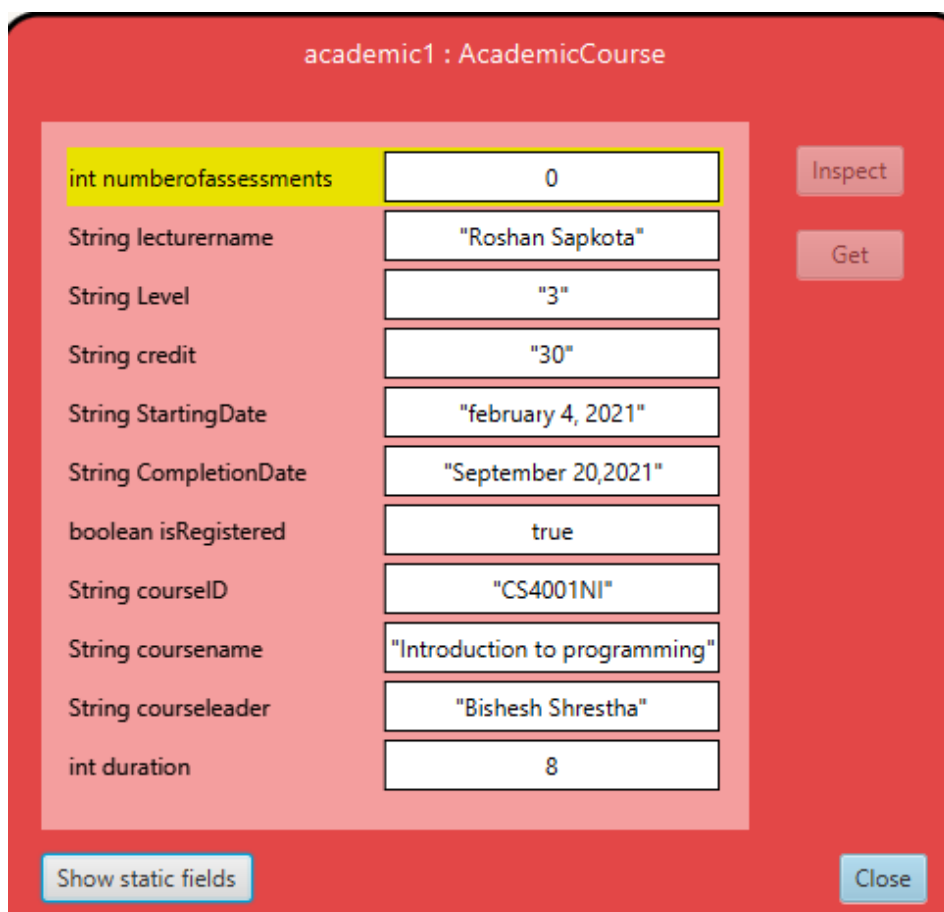


Figure 5: Screenshot for re-inspection of Academic Course class

Test 2: To inspect Non-Academic Course class, register Non-Academic Course and re-inspect the Non-Academic Course class

Test No:	2
Objective:	To inspect Non-Academic Course class, register Non-Academic Course and re-inspect the Non-Academic Course class.
Action:	<p>The Non-Academic Course is called with the following arguments:</p> <p>course ID= "CC4057NI"</p> <p>course name="Introduction to Information System"</p> <p>duration= 8</p> <p>prerequisite= "+2 passed"</p> <p>→Inspection of the Non-Academic Course class.</p> <p>→void register Non-Academic course is called with the following arguments:</p> <p>course leader= "Biken Gurung"</p> <p>instructor name= "Sukrit Sakya"</p> <p>starting date= "June 10, 2021"</p> <p>completion date= "March 7, 2022"</p> <p>exam date= February 2, 2022"</p> <p>→Re-inspection of the Non-Academic Course class.</p>
Expected Result:	The Non-Academic Course would be registered.
Actual Result:	The Non-Academic Course was registered.
Conclusion:	The test is successful.

Table 5: To inspect Non-Academic Course class, register non-academic course and re-inspect the non-academic course class

Output Result:

Blue: Create Object

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

Name of Instance:

new NonAcademicCourse(,
 ,
 ,
)

OK Cancel

Figure 6: Screenshot of assigning the data in Non-Academic class

nonAcade1 : NonAcademicCourse

String instructorname	null
String startingdate	""
String completiondate	null
String examdate	""
String prerequisite	" +2 passed"
boolean isRegistered	false
boolean isRemoved	false
String courseID	"CC4057NI"
String coursename	"Introduction to Information Syst..."
String courseleader	""
int duration	8

Inspect Get

Show static fields Close

Figure 7: Screenshot for the inspection of Non-Academic Course class

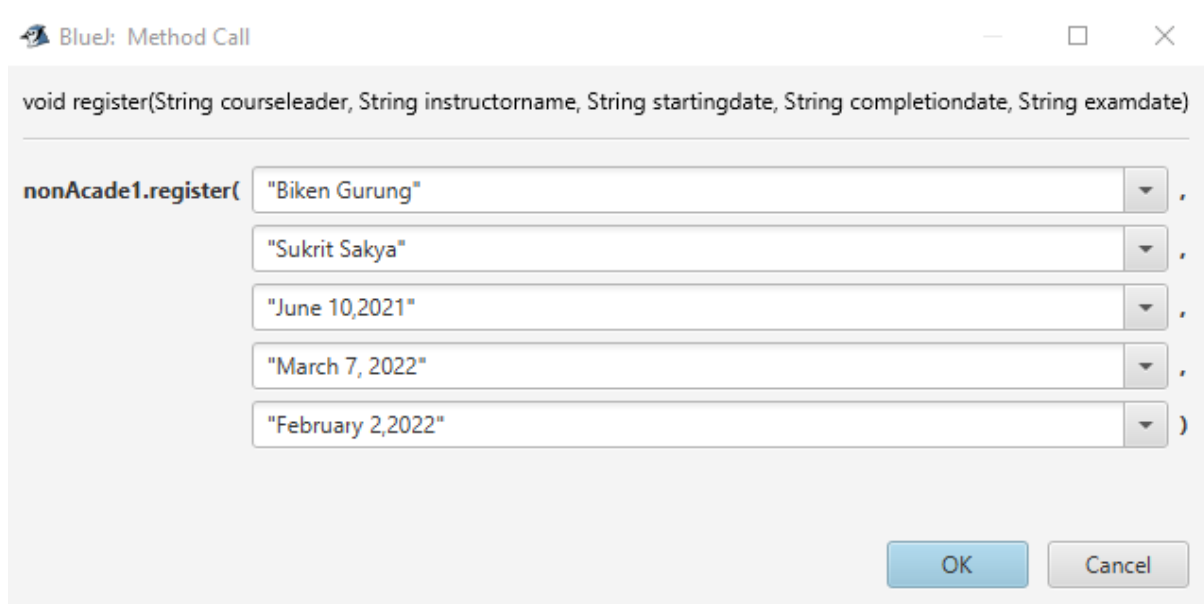


Figure 8: Screenshot for assigning the data in void register Non-Academic Course class

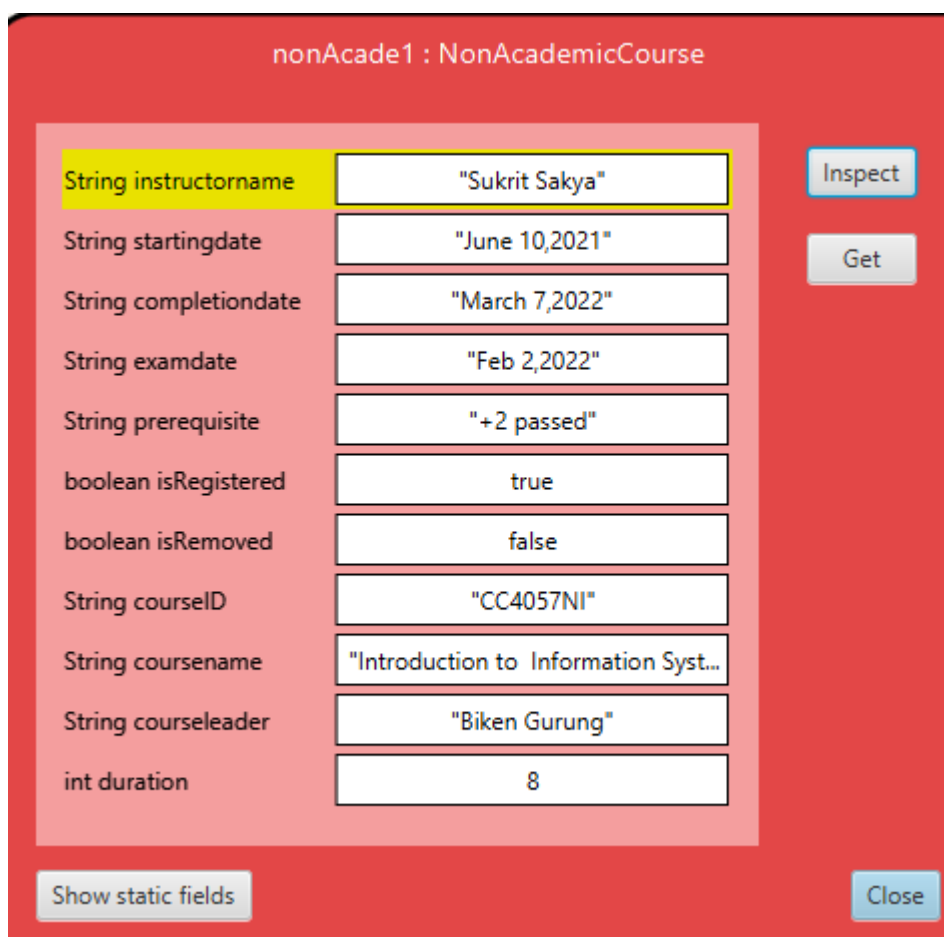


Figure 9: Screenshot for re-inspection of Non-Academic Course class

Test 3: To inspect Non-Academic Course class again, and isremoved to true to true and re-inspect the Non-Academic Course

Test No:	3
Objectives:	To inspect Non-Academic Course class again, and isremoved to true and re-inspect the Non-Academic Course class.
Action:	→The Non-Academic Course is called with the following arguments: course ID= "MA4001NI" course name= "Logic and Problem Solving" duration= 3 prerequisite= "+2 passed" →Re-inspection of the Non-Academic Course class →Calling void remove()
Expected Result:	The position of isRemoved would be changed.
Actual Result:	The position of isRemoved is changed.
Conclusion:	The test is successful.

Table 6: To inspect the Non-Academic Course class again and isremoved to true, re-inspect the Non-Academic course class

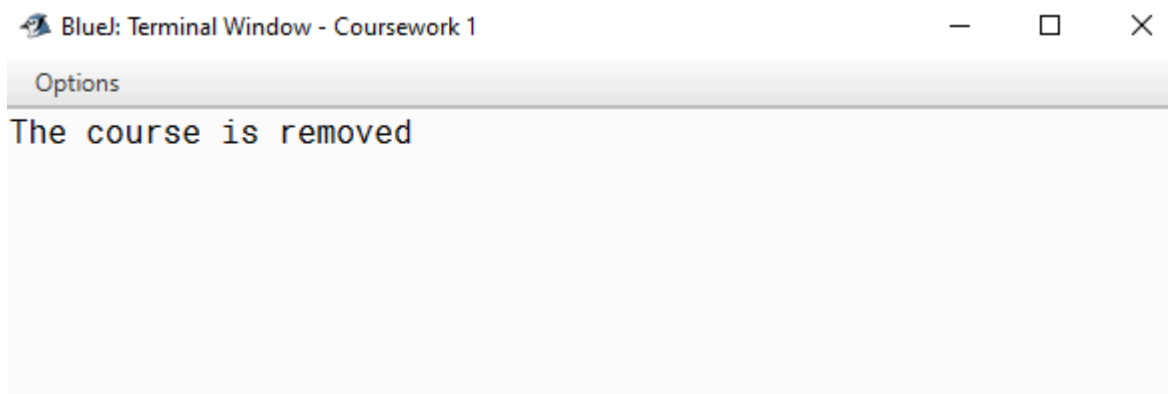
Output Result:

Figure 10: Screenshot of displaying course is removed

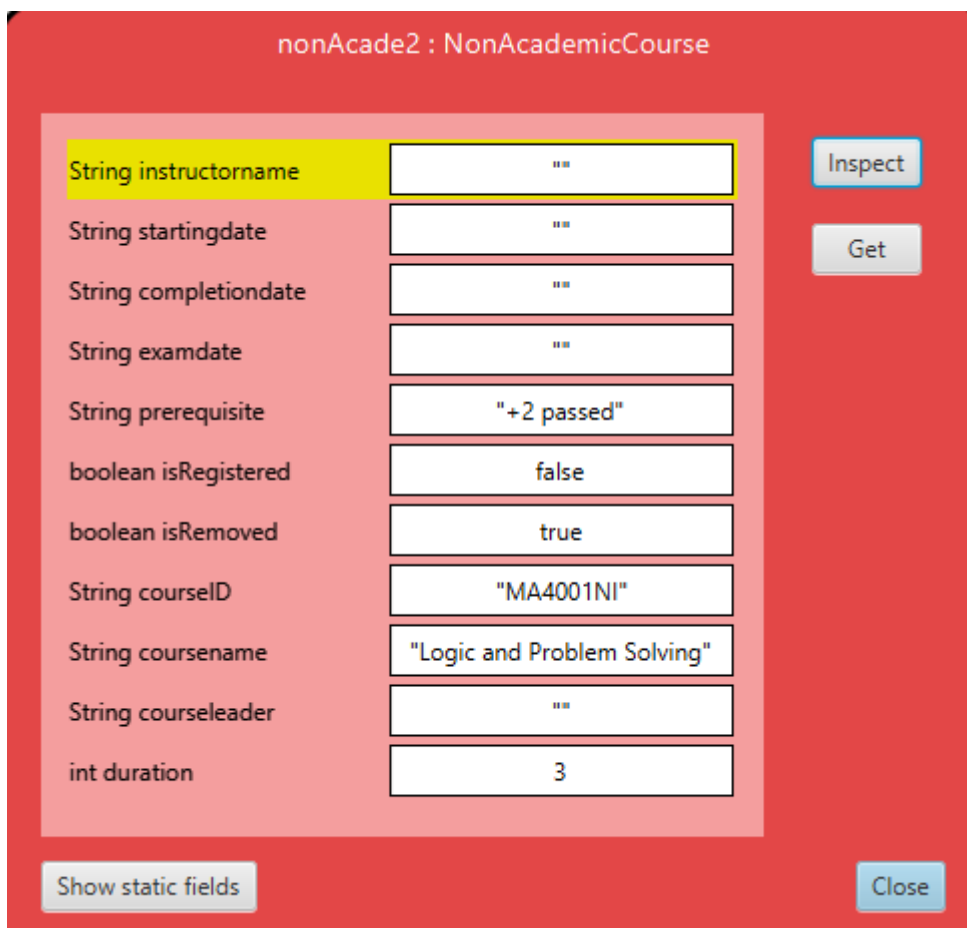


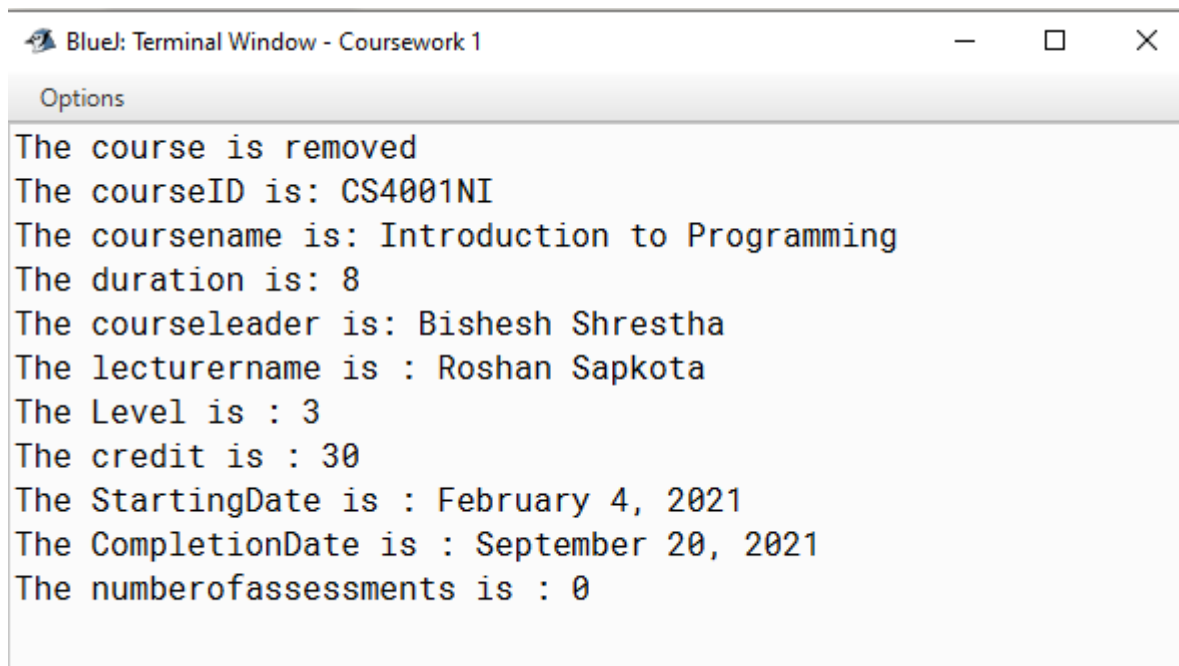
Figure 11: Screenshot for the inspection of Non-Academic Course class

Test 4: To display the detail of Academic Course and Non-Academic Course classes

Test No:	4
Objectives:	To display the detail of Academic Course and Non-Academic Course classes.
Action:	The details of the Academic Course class is displayed The details of Non-Academic Course class is displayed
Expected Result:	The details of Academic Course and Non-Academic Course class would be displayed
Actual Result:	The details of the Academic and Non-Academic Course class is displayed
Conclusion:	The test is successful.

Table 7: To display the details of Academic Course and Non-Academic Course classes.

Output Result:

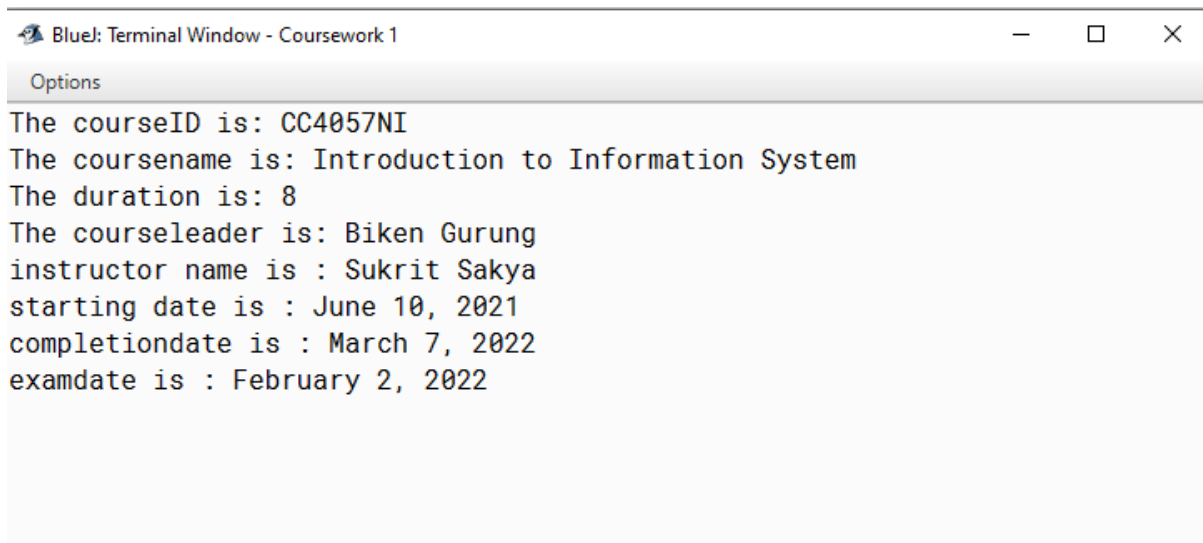


The screenshot shows a terminal window titled "BlueJ: Terminal Window - Coursework 1". The terminal output displays the following details for an Academic Course class:

```

The course is removed
The courseID is: CS4001NI
The coursename is: Introduction to Programming
The duration is: 8
The courseleader is: Bishesh Shrestha
The lecturername is : Roshan Sapkota
The Level is : 3
The credit is : 30
The StartingDate is : February 4, 2021
The CompletionDate is : September 20, 2021
The numberofassessments is : 0
  
```

Figure 12: Displaying the details of the Academic Course class

A screenshot of a terminal window titled "Blue: Terminal Window - Coursework 1". The window has a standard macOS-style title bar with minimize, maximize, and close buttons. Below the title bar is a tab labeled "Options". The terminal content displays the following text:

```
The courseID is: CC4057NI
The coursenam is: Introduction to Information System
The duration is: 8
The courseleader is: Biken Gurung
instructor name is : Sukrit Sakya
starting date is : June 10, 2021
completiondate is : March 7, 2022
examdate is : February 2, 2022
```

Figure 13: Displaying the details of the Non-Academic Course class

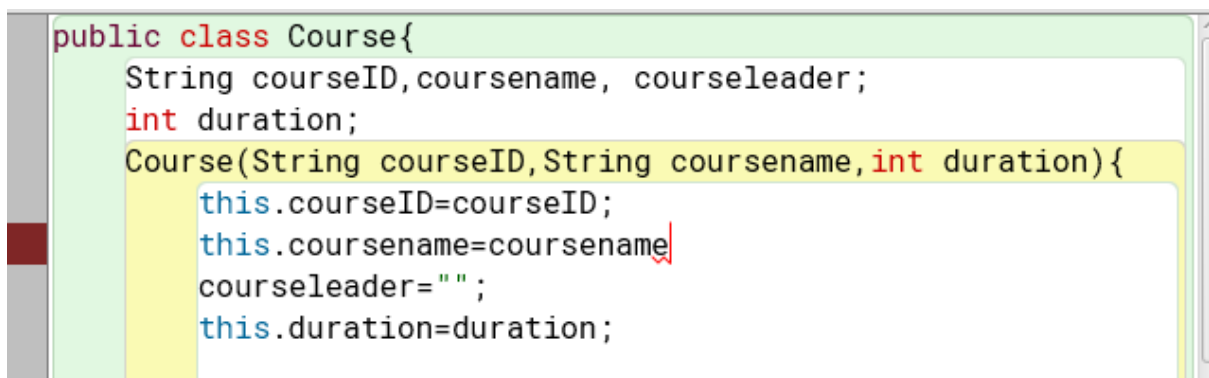
Error Detection

The errors were seen during the program. All those errors were solved by observing those errors closely.

Syntax Error

Everything in a computer is designed in a concrete syntax form. If our input does not match that set of syntax, there are high chances of us facing a syntax error. By definition, we can say, that syntax error is a mistake in the input by the user therefore, the computer cannot answer the input question. (Anon., n.d.)

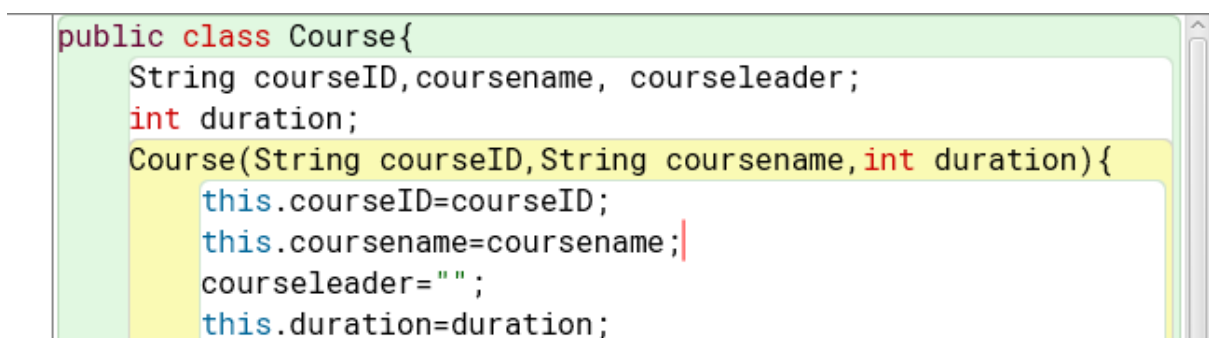
The errors found in the program are given below:



```
public class Course{
    String courseID,coursename, courseleader;
    int duration;
    Course(String courseID,String coursename,int duration){
        this.courseID=courseID;
        this.coursename=coursename
        courseleader="";
        this.duration=duration;
    }
}
```

Figure 14: Error due to missing semi colon

The error was solved by closing the following by semi colon



```
public class Course{
    String courseID,coursename, courseleader;
    int duration;
    Course(String courseID,String coursename,int duration){
        this.courseID=courseID;
        this.coursename=coursename;
        courseleader="";
        this.duration=duration;
    }
}
```

Figure 15: Error solved

Semantic Error

A semantic error is a violation of the rules of meaning of a natural language or a programming language. When there are semantic errors in a C++ program, the compiler does not translate the program into executable code. Most of the time, semantic errors do NOT generate compiler warnings. Most of the compile time errors are scope and declaration errors. For example: undeclared or multiple declared identifiers. Type mismatched is another compile time error. The semantic error can arise using the wrong variable or using the wrong operator or doing an operation in the wrong order. (Anon., n.d.)

The errors found in the program are given below:

BlueJ: Create Object

AcademicCourse(String Level, String credit, String courseID, String CompletionDate, String courseName, int duration)

Name of Instance:

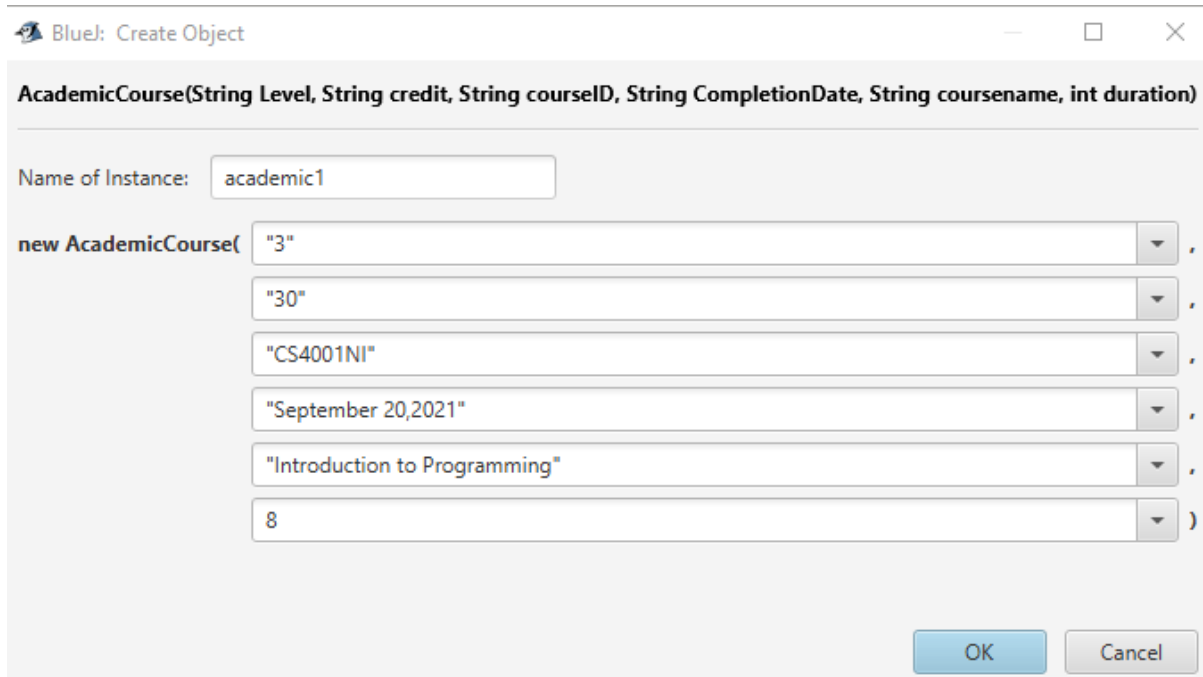
new AcademicCourse(,
 ,
 ,
 ,
 ,
)

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

OK Cancel

Figure 16: Error due to incompatible types

The following error was solved by writing the correct value in place of the integer instead of the string.



BlueJ: Create Object

AcademicCourse(String Level, String credit, String courseID, String CompletionDate, String coursename, int duration)

Name of Instance:

new AcademicCourse(,
 ,
 ,
 ,
 ,
)

OK Cancel

Figure 17: Error solve

Logical Error

The logical error is a program error made by the programmer while writing the program source code. In general, logical error executes unexpected results for logic value. Both interpreter and compiler and even scripting language has logical errors. Unlike syntax error and runtime error, the logical error cannot be detected by the compiler and interpreter. (Anon., n.d.)

The Logical error seen in the program is given below:

```
}  
public void register(String courseleader,String instructorname,String start  
    if(isRegistered!=false){  
        this.setinstructorname(instructorname);  
        this.isRegistered=true;  
        this.courseleader=courseleader;  
        this.startingdate=startingdate;  
        this.completiondate=completiondate;  
        this.examdate=examdate;  
    }else{
```

Figure 18: Logical error of the program

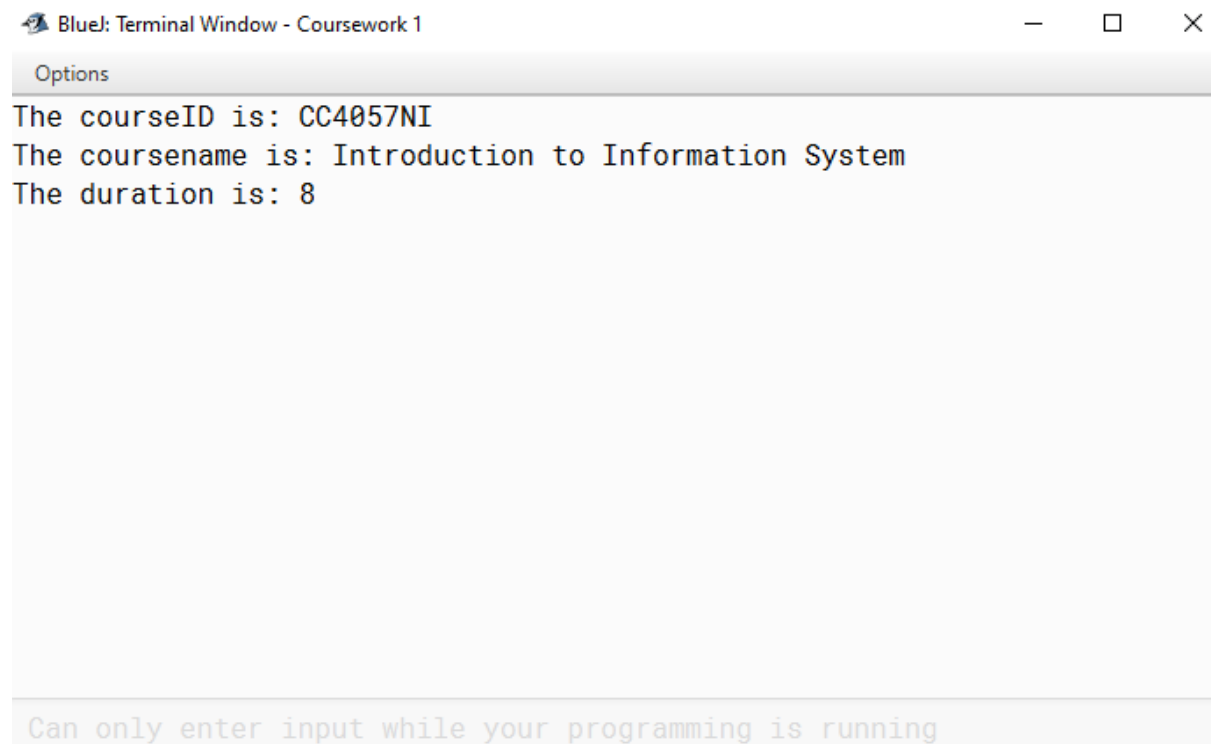
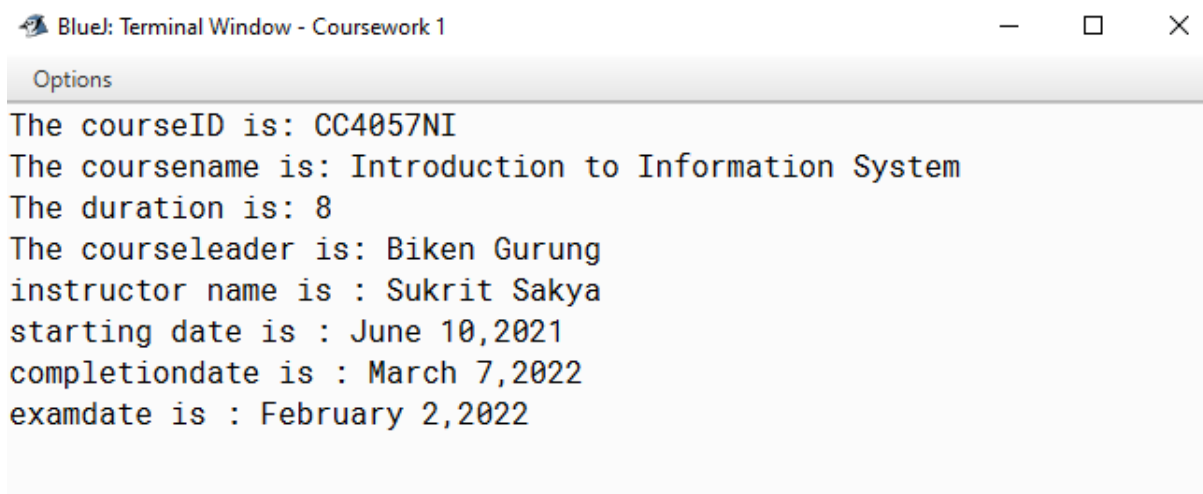


Figure 19: Displaying Non-Academic Course detail due to error

The error was seen during the displaying of the details. So, to remove the error we corrected the statement in the code which was suitable for the program.

```
public void register(String courseleader,String instructorname,String start  
if(isRegistered==false){  
    this.setinstructorname(instructorname);  
    this.isRegistered=true;  
    this.courseleader=courseleader;  
    this.startingdate=startingdate;  
    this.completiondate=completiondate;  
    this.examdate=examdate;  
}else{
```

Figure 20: Error fix



BlueJ: Terminal Window - Coursework 1

Options

The courseID is: CC4057NI
The coursenam is: Introduction to Information System
The duration is: 8
The courseleader is: Biken Gurung
instructor name is : Sukrit Sakya
starting date is : June 10,2021
completiondate is : March 7,2022
examdate is : February 2,2022

Figure 21: Displaying Non-Academic Course detail with no error

Conclusion of the Report

To whole up, this coursework was relegated to us for registering the Course, Academic Course, and Non-Academic Course. Three distinct classes were created in this coursework exceptionally. The course class is the parent class and the academic and non-academic classes are the sub classes. Each of the methods were allotted with different ascribes along with various data types (integer, string). So, this project helps me a lot to acquire information about the diverse data types and the attributes alongside various accessor methods such as setter and getter method.

While doing this coursework I needed to confront the challenges particularly in the coding segment as numerous mistakes occurred and as I was new to the theme too I was exceptionally confounded. I was likewise somewhat confused in regards to the pseudocode and the class diagram. Yet in order to conquer the confusion and troubles, a lot of explores were finished with respect to the applicable topics. Cent percent exertion was given to finish the undertaking allocated in this topic. Research was done in regards to java programming from the beginning and it was executed in this coursework which helped a lot to achieve the tasks. Regular interaction with instructors, consistent exertion and a lot of explores, going through the lecture slides and surfing in the web assisted with acquiring sound information about java and its purpose. I likewise, became more acquainted with about the various methods which are utilized in the program, I knew about various terms alongside their capabilities.

Despite the fact that it was challenging and troublesome from the outset, the coursework was finished on schedule and submitted on time as well. I had the chance to learn numerous new things and themes which I knew about. It was decent encounter to foster a program utilizing java and it was enjoyable to chip away at this project as well.

Bibliography

Anon., n.d. *appuals.com*. [Online]

Available at: <https://appuals.com/what-is-a-syntax-error/>

[Accessed 18 05 2021].

Anon., n.d. *economictimes.indiatimes.com*. [Online]

Available at: <https://economictimes.indiatimes.com/definition/pseudocode>

[Accessed 05 15 2021].

Anon., n.d. *findanyanswer.com*. [Online]

Available at: [https://findanyanswer.com/what-is-semantic-error-in-](https://findanyanswer.com/what-is-semantic-error-in-c#:~:text=A%20semantic%20error%20is%20a%20violation%20of%20the,time%2C%20semantic%20errors%20do%20NOT%20generate%20compiler%20warnings.)

[c#:~:text=A%20semantic%20error%20is%20a%20violation%20of%20the,time%2C%20semantic%20errors%20do%20NOT%20generate%20compiler%20warnings.](https://findanyanswer.com/what-is-semantic-error-in-c#:~:text=A%20semantic%20error%20is%20a%20violation%20of%20the,time%2C%20semantic%20errors%20do%20NOT%20generate%20compiler%20warnings.)

[Accessed 18 05 2021].

Anon., n.d. *www.progracoding.com*. [Online]

Available at: <https://www.progracoding.com/logical-error/>

[Accessed 18 05 2021].

Anon., n.d. *www.techopedia.com*. [Online]

Available at: <https://www.techopedia.com/definition/16466/class-diagram>

[Accessed 05 12 2021].

Fatehi, A., 2021. *codifyexperts.com*. [Online]

Available at: <https://codifyexperts.com/blog/what-is-testing-and-why-is-it-important-in-coding-process/>

[Accessed 17 05 2021].

Appendix

❖ Course class

```
public class Course{
    String courseID,coursename, courseleader;
    int duration;
    Course(String courseID,String coursename,int duration){
        this.courseID=courseID;
        this.coursename=coursename;
        courseleader="";
        this.duration=duration;
    }
    public String getcourseID(){
        return courseID;
    }
    public String getcoursename(){
        return coursename;
    }
    public String getcourseleader(){
        return courseleader;
    }
    public int getduration(){
        return duration;
    }
    public void setcourseleader(String newname){
        this.courseleader=newname;
    }
    public void display(){
        System.out.println("The courseID is: "+courseID);
        System.out.println("The coursename is: "+coursename);
    }
}
```

```
        System.out.println("The duration is: "+duration);
        if(courseleader!=""){
            System.out.println("The courseleader is: "+courseleader);
        }

    }

}
```

❖ Academic Course class

```
public class AcademicCourse extends Course{

    int numberofassessments;

    String lecturername, Level, credit, StartingDate, CompletionDate;

    boolean isRegistered;

    public AcademicCourse(String Level, String credit, String courseID, String
    CompletionDate ,String coursename,int duration){

        super(courseID,coursename,duration);

        this.Level=Level;

        this.credit=credit;

        this.numberofassessments=numberofassessments;

        lecturername="";

        StartingDate="";

        CompletionDate="";

        isRegistered=false;

    }

}
```



```
public String getlecturername(){
    return lecturername;
}

public String getLevel(){
    return Level;
}

public String getcredit(){
    return credit;
}

public String getStartingDate(){
    return StartingDate;
}

public String getCompletionDate(){
    return CompletionDate;
}

public int getnumberofassessments(){
    return numberofassessments;
}

public boolean getisRegistered(){
    return isRegistered;
}

public void setlecturername(String lecturername){
    this.lecturername=lecturername;
}
```

```
public void setnumberofassessments(int numberofassessments){  
    this.numberofassessments=numberofassessments;  
}  
  
public void register(String courseleader,String lecturername,String  
StartingDate,String CompletionDate){  
    if (isRegistered==true){  
        System.out.println("The course is already registered");  
        System.out.println("The courseleader is : "+courseleader);  
        System.out.println("The StartingDate is : "+StartingDate);  
        System.out.println("The CompletionDate is : "+CompletionDate);  
    }else{  
        super.setcourseleader(courseleader);  
this.lecturername=lecturername;  
        this.StartingDate=StartingDate;  
        this.CompletionDate=CompletionDate;  
        this.isRegistered=true;  
    }  
}  
  
public void display(){  
    super.display();  
    if (isRegistered==true){  
        System.out.println("The lecturername is : "+lecturername);  
        System.out.println("The Level is : "+Level);  
        System.out.println("The credit is : "+credit);  
        System.out.println("The StartingDate is : "+StartingDate);
```

```
        System.out.println("The CompletionDate is : "+CompletionDate);

        System.out.println("The numberofassessments is : 
"+numberofassessments);

    }

}

}
```

❖ Non-Academic Course class

```
public class NonAcademicCourse extends Course{

    String instructorname, startingdate,completiondate,examdate,prerequisite;

    boolean isRegistered,isRemoved;

    public NonAcademicCourse(String courseID, String coursename, int duration,
String prerequisite){

        super(courseID,coursename,duration);

        this.prerequisite=prerequisite;

        startingdate="";

        completiondate="";

        examdate="";

        isRegistered=false;

        isRemoved=false;

    }

    public String getinstructorname(){

        return instructorname;

    }

}
```

```
public String getstartingdate(){
    return startingdate;
}

public String getcompletiondate(){
    return completiondate;
}

public String getexamdate(){
    return examdate;
}

public String getprerequisite(){
    return prerequisite;
}

boolean getisRegistered(){
    return isRegistered;
}

boolean getisRemoved(){
    return isRemoved;
}

public void setinstructorname(String newinstructor){
    if(isRegistered==false){
        instructorname=newinstructor;
    }
    else{
        System.out.println("The instructorname cannot be changed");
    }
}
```

```
    }  
}  
  
public void register(String courseleader,String instructorname,String  
startingdate,String completiondate,String examdate){  
    if(isRegistered==false){  
        this.setinstructorname(instructorname);  
        this.isRegistered=true;  
        this.courseleader=courseleader;  
        this.startingdate=startingdate;  
        this.completiondate=completiondate;  
        this.examdate=examdate;  
    }else{  
        System.out.println("The course is already registered");  
    }  
}  
  
public void remove(){  
    if(isRemoved==true){  
        System.out.println("The course is removed");  
    }else{  
        super.setcourseleader("");  
        this.instructorname="";  
        this.startingdate="";  
        this.completiondate="";  
        this.examdate="";  
        this.isRegistered=false;  
    }  
}
```

```
        this.isRemoved=true;
    }
}

public void display(){
    super.display();
    if(isRegistered==true){
        System.out.println("instructor name is : " + instructorname);
        System.out.println("starting date is : " +startingdate);
        System.out.println("completiondate is : " +completiondate);
        System.out.println("examdate is : " +examdate);

    }
}
}
```