LONDON METROPOLITAN UNIVERSITY

Islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CC4001NI Programming

COURSEWORK-2

Assessment Weightage & Type

30% Individual Coursework

Year and Semester

Spring 2021

Student Name: Niwahang Angbuhang

Group: C9

London Met ID: 20048942

College ID: NP01CP4S210237

Assignment Due Date: 20th August 2021

Assignment Submission Date: 19th August 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

List of Figures

List of Tables

1. Introduction

This is the second coursework of the programming module. The second coursework is related to the first coursework we were given. We are required to make a Graphical User Interface (GUI) for the first coursework. The program is created using Java as a programming language. The codes which were used in the previous coursework are also made use of in this coursework.

The main purpose of this coursework is to make a GUI for the academic course and non-academic course. Through the help of GUI, add and registering the courses are easier and the program is more user-friendly. The GUI will make it easier to add, register, remove and display the courses. The methods used in the previous coursework like register, remove, and display is also used in this coursework. The methods are added in the form of a button using an action event. Some new methods are also introduced in the coursework.

The tools used in the coursework were BlueJ, Draw.io, and Microsoft Word. BlueJ was used in writing the program, compiling it, and running the program for errors and testing. Draw.io was used to create the class diagrams for the INGCollege class. Finally, Microsoft Word was used to write the report of the coursework.

2. Class Diagram

2.1 Class Diagram of Course Class

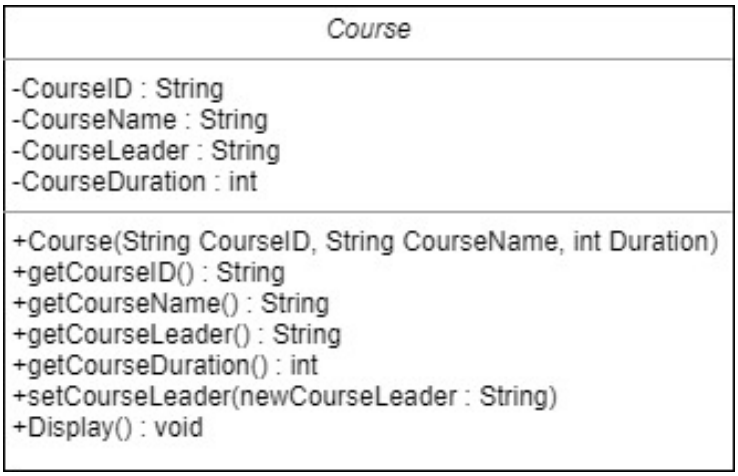

Figure 1 Class Diagram of Course Class

2.2 Class Diagram of Academic Course Class

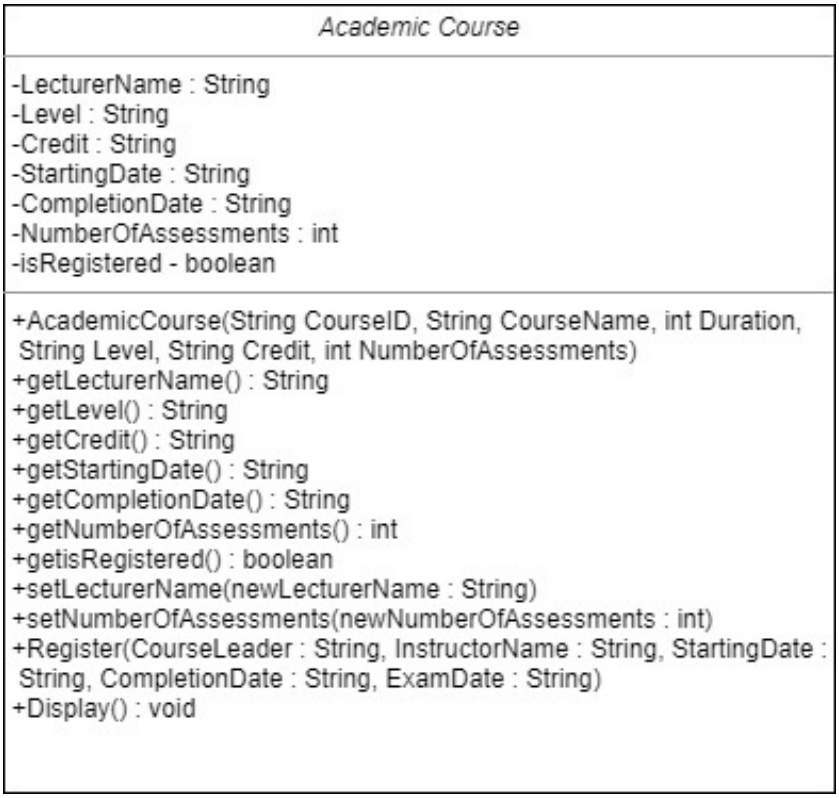

Figure 2 Class Diagram of Academic Course Class

2.3 Class Diagram of Non-Academic Course Class

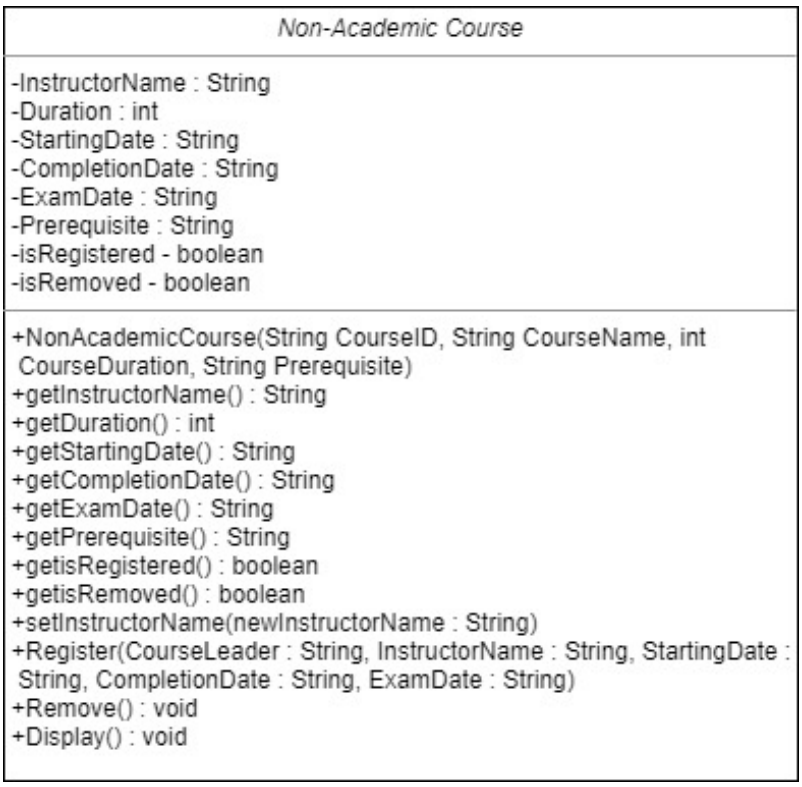

Figure 3 Class Diagram of Non-Academic Course Class

2.4 Class Diagram of INGCollege Class

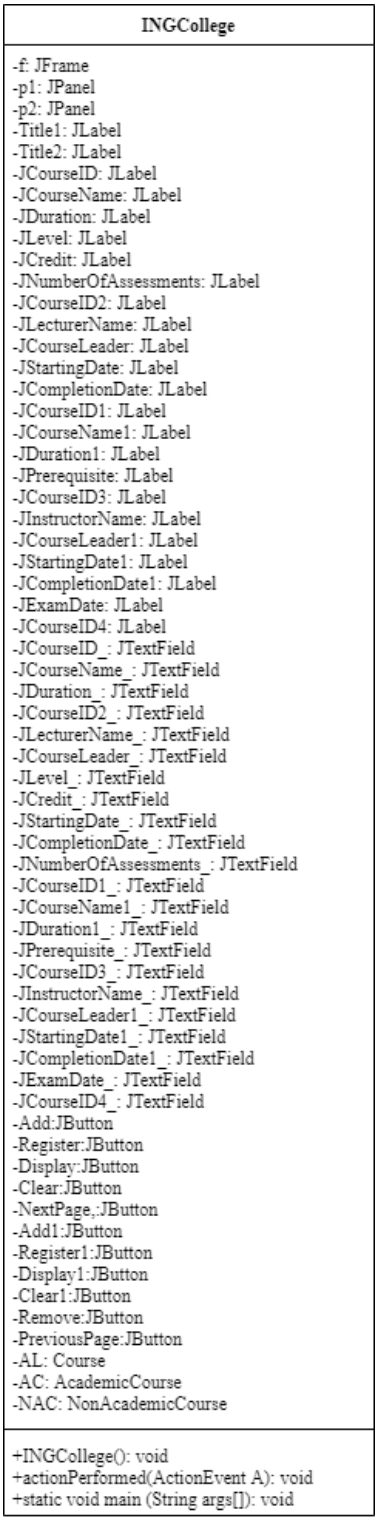

Figure 4 Class Diagram of INGCollege class

2.5 Final Class Diagram

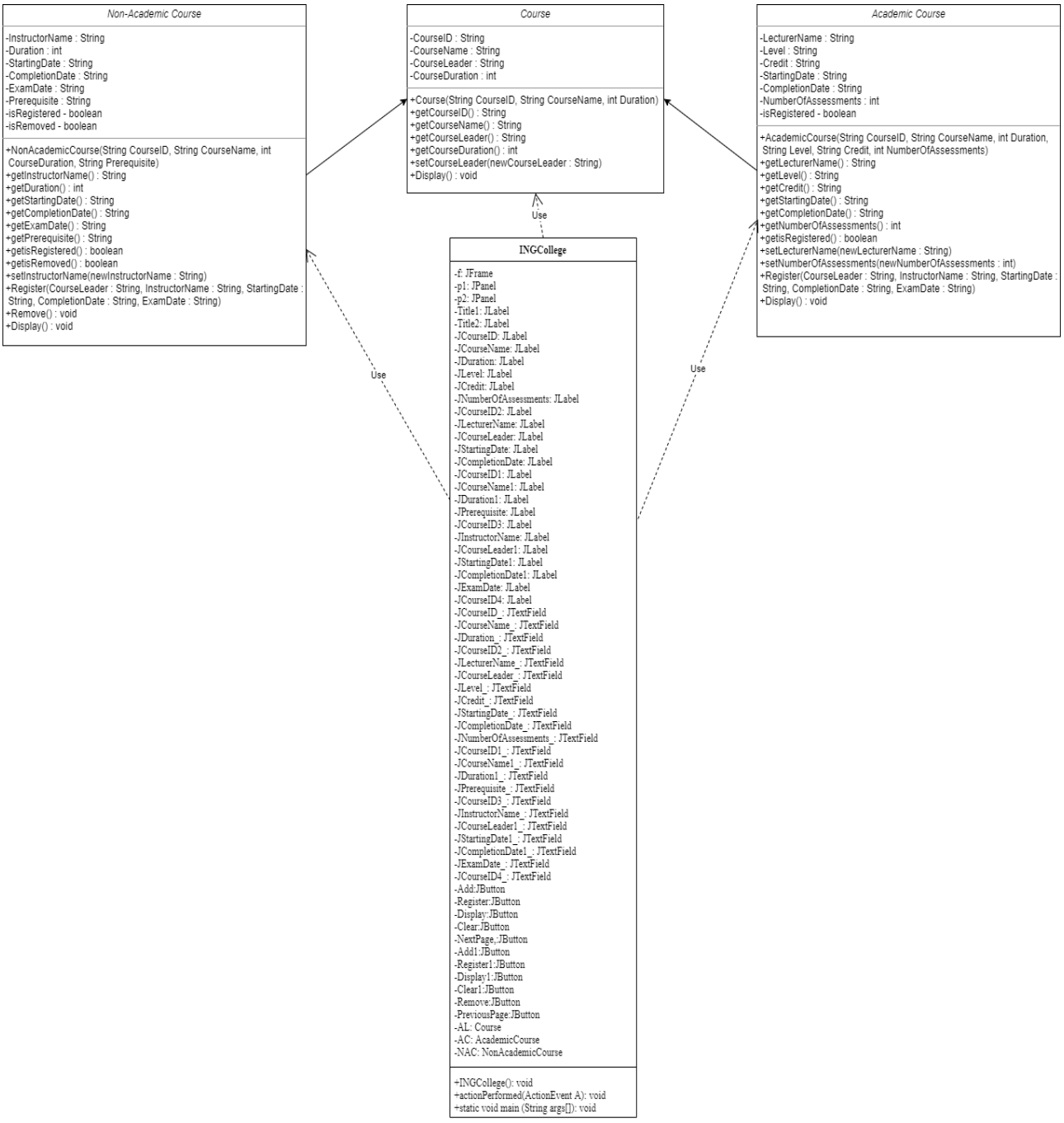

Figure 5 Final Class Diagram

3. Pseudocode

3.1 Pseudocode for Course Class

CREATE class Course

DECLARE instance variable CourseID of String type
DECLARE instance variable CourseName of String type
DECLARE instance variable CourseLeader of String type
DECLARE instance variable CourseDuration of int type

CREATE Constructor Course(**PASS** parameter CourseID of String type, CourseName of String type, CourseDuration of int type)
ASSIGN parameter CourseID to instance variable CourseID
ASSIGN parameter CourseName to instance variable CourseName
ASSIGN parameter CourseDuration to instance variable CourseDuration
ASSIGN CourseLeader to null

CREATE method getCourseID with return type String
RETURN CourseID

CREATE method getCourseName with return type String
RETURN CourseName

CREATE method getCourseLeader with return type String
RETURN CourseLeader

CREATE method getCourseDuration with return type int
RETURN CourseDuration

CREATE method setCourseLeader (**PASS** parameter newCourseLeader of String Type)
ASSIGN parameter CourseLeader to instance variable newCourseLeader

CREATE method Display
PRINT CourseID
PRINT CourseName
PRINT CourseDuration
IF (CourseLeader is null)
PRINT CourseLeader
ENDIF

3.2 Pseudocode for Academic Course Class

CREATE class AcademicCourse

DECLARE instance variable LecturerName of String type
DECLARE instance variable Level of String type
DECLARE instance variable Credit of String type
DECLARE instance variable StartingDate of String type
DECLARE instance variable CompletionDate of String type
DECLARE instance variable NumberOfAssessments of int type
DECLARE instance variable isRegistered of boolean type

CREATE Constructor AcademicCourse (**PASS** parameter CourseID of String type, CourseName of String type, CourseDuration of int type, Level of String type, Credit of String type, NumberOfAssessments of int type)
CALL constructor from parent class
ASSIGN parameter CourseID to instance variable CourseID
ASSIGN parameter CourseName to instance variable CourseName
ASSIGN parameter CourseDuration to instance variable CourseDuration
ASSIGN parameter Level to instance variable Level
ASSIGN parameter NumberOfAssessments to instance variable NumberOfAssessments
ASSIGN parameter Credit to instance variable Credit
ASSIGN LecturerName to null
ASSIGN StartingDate to null
ASSIGN CompletionDate to null
ASSIGN isRegistered to false

CREATE method getLecturerName with return type String
RETURN LecturerName

CREATE method getLevel with return type String
RETURN Level

CREATE method getCredit with return type String
RETURN Credit

CREATE method getStartingDate with return type String
RETURN StartingDate

CREATE method getCompletionDate with return type String
RETURN CompletionDate

CREATE method getNumberOfAssessments with return type int
RETURN NumberOfAssessments

CREATE method getisRegistered with return type boolean
    RETURN isRegistered

CREATE method setLecturerName (**PASS** parameter newLecturerName of String Type)
    ASSIGN parameter newLecturerName to instance variable LecturerName

CREATE method setNumberOfAssessments (**PASS** parameter newNumberOfAssessments of String Type)
    ASSIGN parameter newNumberOfAssessments to instance variable NumberOfAssessments

CREATE method Register (**PASS** parameter CourseLeader of String type, LecturerName of String type, StartingDate of String type, CompletionDate of String type)
    IF (isRegistered is true)
        PRINT "Academic course is already registered"
        PRINT LecturerName
        PRINT StartingDate
        PRINT CompletionDate
    ELSE
        CALL setCourseLeader method from parent class
        ASSIGN parameter LecturerName to instance variable LecturerName
        ASSIGN parameter StartingDate to instance variable StartingDate
        ASSIGN parameter CompletionDate to instance variable CompletionDate
        ASSIGN isRegistered to true

CREATE method Display
    CALL Display method from parent class
    IF (isRegistered is true)
        PRINT LecturerName
        PRINT Level
        PRINT Credit
        PRINT StartingDate
        PRINT CompletionDate
        PRINT NumberOfAssessments
    ENDIF

3.3 Pseudocode for Non-Academic Course Class

CREATE NonAcademicCourse

DECLARE instance variable InstructorName of String type
DECLARE instance variable CourseDuration of int type
DECLARE instance variable StartingDate of String type
DECLARE instance variable CompletionDate of String type
DECLARE instance variable ExamDate of String type
DECLARE instance variable Prerequisite of String type
DECLARE instance variable isRegistered of boolean type
DECLARE instance variable isRemoved of boolean type

CREATE Constructor NonAcademicCourse (**PASS** parameter CourseID of String type, CourseName of String type, CourseDuration of int type, Prerequisite of String type)

CALL constructor from parent class
ASSIGN parameter CourseID to instance variable CourseID
ASSIGN parameter CourseName to instance variable CourseName
ASSIGN parameter CourseDuration to instance variable CourseDuration
ASSIGN parameter Prerequisite to instance variable Prerequisite
ASSIGN StartingDate to null
ASSIGN CompletionDate to null
ASSIGN ExamDate to null
ASSIGN isRegistered to false
ASSIGN isRemoved to false

CREATE method getInstructorName with return type String
RETURN InstructorName

CREATE method getCourseDuration with int type String
RETURN CourseDuration

CREATE method getStartingDate with return type String
RETURN StartingDate

CREATE method getCompletionDate with return type String
RETURN CompletionDate

CREATE method getExamDate with return type String
RETURN ExamDate

CREATE method getPrerequisite with return type String
RETURN Prerequisite

CREATE method getisRegistered with return type boolean

RETURN isRegistered

CREATE method getisRemoved with return type boolean
RETURN isRemoved

CREATE method setInstructorName (**PASS** parameter newInstructorName of String Type)
IF (isRegistered is true)
PRINT “Since instructor is already registered, it is not possible”
ELSE
ASSIGN parameter newInstructorName to instance variable InstructorName

CREATE method Register (**PASS** parameter CourseLeader of String type, InstructorName of String type, StartingDate of String type, CompletionDate of String type, ExamDate of String type)
IF (isRegistered is true)
PRINT “Non-academic course is already registered”
ELSE
CALL method setCourseLeader from parent class
ASSIGN parameter InstructorName to instance variable LecturerName
ASSIGN parameter StartingDate to instance variable StartingDate
ASSIGN parameter CompletionDate to instance variable CompletionDate
ASSIGN parameter ExamDate to instance variable ExamDate
ASSIGN isRegistered to true
ASSIGN isRemoved to false

CREATE method Remove
IF (isRemoved is true)
PRINT “Non-Academic course is already removed.”
ELSE
CALL setCourseLeader from parent class(**PASS** null in instance variable)
ASSIGN instance variable InstructorName to null
ASSIGN instance variable StartingDate to null
ASSIGN instance variable CompletionDate to null
ASSIGN instance variable ExamDate to null
ASSIGN instance variable isRegistered to false
ASSIGN instance variable isRemoved to true

CREATE method Display
CALL Display method from parent class
IF (isRegistered is true)
PRINT InstructorName
PRINT StartingDate

PRINT CompletionDate
PRINT ExamDate
PRINT Prerequisite
ENDIF

3.4 Pseudocode for INGCollege Class

IMPORT all from javax.swing
IMPORT all from java.awt
IMPORT all from java.awt.event
IMPORT ArrayList from java.util

DEFINE class INGCollege which implements ActionListener

DECLARE instance variable f of type JFrame
DECLARE instance variables p1 and p2 of type JPanel
DECLARE instance variables Title1, Title2, JCourseID, JCourseName, JDuration, JLevel, JCredit, JNumberOfAssessments, JCourseID2, JLecturerName, JCourseLeader,JStartingDate, JCompletionDate, JCourseID1, JCourseName1, JDuration1, JPrerequisite, JCourseID3, JInstructorName, JCourseLeader1, JStartingDate1, JCompletionDate1, JExamDate, JCourseID4 of type JLabel
DECLARE instance variables JCourseID_, JCourseName_, JDuration_, JCourseID2_, JLecturerName_, JCourseLeader_, JLevel_, JCredit_, JStartingDate_, JCompletionDate_, JNumberOfAssessments_, JCourseID1_, JCourseName1_, JDuration1_, JPrerequisite_, JCourseID3_, JInstructorName_, JCourseLeader1_, JStartingDate1_, JCompletionDate1_, JExamDate_, JCourseID4_ of type JTextField
DECLARE instance variables Add, Register, Display, Clear, NextPage, Add1, Register1, Display1, Clear1, Remove,PreviousPage of type JButton

INITIALIZE new ArrayList as AL which holds objects of Course
DECLARE object AC of Academic Course
DECLARE object NAC of Non-Academic Course

DEFINE a constructor INGCollege

INITIALIZE JFrame as f as Course Registration
INITIALIZE JPanels as p1 and p2

INITIALIZE JLabel Title1 as Academic Course
INITIALIZE JLabel JCourseID as Course ID :
INITIALIZE JLabel JCourseName as Course Name :
INITIALIZE JLabel JDuration as Duration :
INITIALIZE JLabel JLevel as Level :
INITIALIZE JLabel JCredit as Credit :

INITIALIZE JLabel JNumberOfAssessments as Number of Assessments :

INITIALIZE JLabel Title2 as Non-Academic Course
INITIALIZE JLabel JCourseID1 as Course ID :
INITIALIZE JLabel JCourseName1 as Course Name :
INITIALIZE JLabel JDuration1 as Duration :
INITIALIZE JLabel JPrerequisite1 as Prerequisite :

INITIALIZE JLabel JCourseID2 as Course ID :
INITIALIZE JLabel JLecturerName as Lecturer Name :
INITIALIZE JLabel JCourseLeader as Course Leader :
INITIALIZE JLabel JStartingDate as Starting Date :
INITIALIZE JLabel JCompletionDate as Completion Date :

INITIALIZE JLabel JCourseID3 as CouresID :
INITIALIZE JLabel JInstructorName as InstructorName :
INITIALIZE JLabel JCourseLeader1 as CourseLeader :
INITIALIZE JLabel JStartingDate1 as Starting Date :
INITIALIZE JLabel JCompletionDate1 as Completion Date :
INITIALIZE JLabel JExamDate as Exam Date :

INITIALIZE JLabel JCourseID4 as Course ID :

INITIALIZE JTextField JCourseID_
INITIALIZE JTextField JCourseName_
INITIALIZE JTextField JDuration_
INITIALIZE JTextField JLevel_
INITIALIZE JTextField JCredit_

INITIALIZE JTextField JCourseID1_
INITIALIZE JTextField JCourseName1_
INITIALIZE JTextField JDuration1_
INITIALIZE JTextField JPrerequisite_

INITIALIZE JTextField JCourseID2_
INITIALIZE JTextField JLecturerName_
INITIALIZE JTextField JCourseLeader_
INITIALIZE JTextField JStartingDate_
INITIALIZE JTextField JCompletionDate_

INITIALIZE JTextField JCourseID3_
INITIALIZE JTextField JInstructorName_
INITIALIZE JTextField JCourseLeader1_
INITIALIZE JTextField JStartingDate1_
INITIALIZE JTextField JCompletionDate1_
INITIALIZE JTextField JExamDate_

INITIALIZE JTextField JCourseID4_

SET f visibility to True
SET p1 visibility to True
SET p2 visibility to False

SET layout of f to null
SET layout of p1 to null
SET layout of p2 to null

SET resizable of f to false
SET size of f to 860, 460
SET size of p1 to 860, 460
SET size of p2 to 860, 460

INITIALIZE JButton Add as Add
INITIALIZE JButton Register as Register
INITIALIZE JButton Display as Display
INITIALIZE JButton Clear as Clear
INITIALIZE JButton NextPage as Next Page

INITIALIZE JButton Add1 as Add
INITIALIZE JButton Register1 as Register
INITIALIZE JButton Display1 as Display
INITIALIZE JButton Clear1 as Clear
INITIALIZE JButton Remove as Remove
INITIALIZE JButton PreviousPage as Previous Page

SET font of Title1 to Arial, BOLD, 25
SET font of Title2 to Arial, BOLD, 25

SET bounds of Tilte1 to (300, 20, 400, 50)
SET bounds of JCourseID to (40, 100, 120, 25)
SET bounds of JCourseName to (430, 100, 120, 25)
SET bounds of JDuration to (40, 150, 120, 25)
SET bounds of JLevel to (430, 150, 120, 25)
SET bounds of JCredit to (40, 200, 120, 25)
SET bounds of JNumberOfAssessments to (430, 200, 150, 25)

SET bounds of Tilte1 to (300, 20, 400, 50)
SET bounds of JCourseID1 to (40, 100, 120, 25)
SET bounds of JCourseName1 to (430, 100, 120, 25)
SET bounds of JDuration1 to (40, 150, 120, 25)
SET bounds of JPrerequisite to (430, 150, 120, 25)

SET bounds of JCourseID2 to (40, 310, 120, 25)
SET bounds of JLecturerName to (430, 310, 120, 25)
SET bounds of JCourseLeader to (40, 360, 120, 25)
SET bounds of JStartingDate to (430, 360, 120, 25)
SET bounds of JCompletionDate to (40, 410, 120, 25)

SET bounds of JCourseID3 to (40, 260, 120, 25)
SET bounds of JInstructorName to (430, 260, 120, 25)
SET bounds of JCourseLeader1 to (40, 310, 120, 25)
SET bounds of JStartingDate1 to (430, 310, 120, 25)
SET bounds of JCompletionDate1 to (40, 360, 120, 25)
SET bounds of JExamDate to (430, 360, 120, 25)

SET bounds of JCourse4 to (40, 460, 120, 25)

SET bounds of JCourseID_ to (160, 100, 180, 25)
SET bounds of JCourseName_ to (585, 100, 180, 25)
SET bounds of JDuration_ to (160, 150, 180, 25)
SET bounds of JLevel_ to (585, 150, 180, 25)
SET bounds of JCredit_ to (160, 200, 180, 25)
SET bounds of JNumberOfAssessments_ to (585, 200, 180, 25)

SET bounds of JCourseID1_ to (160, 100, 180, 25)
SET bounds of JCourseName1_ to (585, 100, 180, 25)
SET bounds of JDuration1_ to (160, 150, 180, 25)
SET bounds of JPrerequisite_ to (585, 150, 180, 25)

SET bounds of JCourseID2_ to (160, 310, 180, 25)
SET bounds of JLecturerName_ to (585, 310, 180, 25)
SET bounds of JCourseLeader_ to (160, 360, 180, 25)
SET bounds of JStartingDate_ to (585, 360, 180, 25)
SET bounds of JCompletionDate_ to (160, 410, 180, 25)

SET bounds of JCourseID3_ to (160, 260, 180, 25)
SET bounds of JInstructorName_ to (585, 260, 180, 25)
SET bounds of JCourseLeader1_ to (160, 310, 180, 25)
SET bounds of JStartingDate1_ to (585, 310, 180, 25)
SET bounds of JCompletionDate1_ to (160, 360, 180, 25)
SET bounds of JExamDate_ to (585, 360, 180, 25)

SET bounds of JCourseID4_ to (160, 460, 180, 25)

SET bounds of Add to (585, 250, 120, 25)
SET bounds of Register to (585, 460, 120, 25)
SET bounds of Display to (220, 510, 120, 25)
SET bounds of Clear to (380, 510, 120, 25)

SET bounds of NextPage to (540, 510, 120, 25)

SET bounds of Add1 to (585, 200, 120, 25)
SET bounds of Register1 to (585, 410, 120, 25)
SET bounds of Remove to (160, 510, 120, 25)
SET bounds of Display1 to (220, 560, 120, 25)
SET bounds of Clear1 to (380, 560, 120, 25)
SET bounds of PreviousPage to (540, 560, 120, 25)

ADD Title1 to p1
ADD JCourseID to p1
ADD JCourseName to p1
ADD JDuration to p1
ADD JLevel to p1
ADD JCredit to p1
ADD JNumberOfAssessments to p1
ADD JCourseID_ to p1
ADD JCourseName_ to p1
ADD JDuration_ to p1
ADD JLevel_ to p1
ADD JCredit_ to p1
ADD JNumberOfAssessments_ to p1
ADD Add to p1
ADD JCourseID2 to p1
ADD JLecturerName to p1
ADD JCourseLeader to p1
ADD JStartingDate to p1
ADD JCompletionDate to p1
ADD JCourseID2_ to p1
ADD JLecturerName_ to p1
ADD JCourseLeader_ to p1
ADD JStartingDate_ to p1
ADD JCompletonDate_ to p1
ADD Register to p1
ADD Display to p1
ADD Clear to p1
ADD NextPage to p1

ADD Title2 to p2
ADD JCourseID1 to p2
ADD JCourseName1 to p2
ADD JDuration1 to p2
ADD JPrerequisite to p2
ADD JCourseID1_ to p2
ADD JCourseName1_ to p2
ADD JDuration1_ to p2

ADD JPrerequisite_ to p2
ADD Add1 to p2
ADD JCourseID3 to p2
ADD JInstructorName to p2
ADD JCourseLeader1 to p2
ADD JStartingDate1 to p2
ADD JCompletionDate1 to p2
ADD JExamDate to p2
ADD JInstructorName_ to p2
ADD JCourseLeader1_ to p2
ADD JStartingDate1_ to p2
ADD JCompletionDate1_ to p2
ADD JExamDate_ to p2
ADD Register1 to p2
ADD JCourseID4 to p2
ADD JCourseID4_ to p2
ADD Remove to p2
ADD Display1 to p2
ADD Clear1 to p2
ADD PreviousPage to p2

ADD p1 to f
ADD p2 to f
SET Default Close Operation to JFrame Exit On Close

ADD ActionListener to Add
ADD ActionListener to Add1
ADD ActionListener to Register
ADD ActionListener to Register1
ADD ActionListener to Display
ADD ActionListener to Display1
ADD ActionListener to Remove
ADD ActionListener to Clear
ADD ActionListener to Clear1
ADD ActionListener to NextPage
ADD ActionListener to PreviousPage

END METHOD

DEFINE method actionPerformed(Action Event A)

IF VALUE of method getSource() is equal to Add

ASSIGN String variable CourseID as empty
ASSIGN String variable CourseName as empty
ASSIGN int variable Duration as null

ASSIGN String variable Level as empty
ASSIGN int variable Credit as null
ASSIGN int variable NumberOfAssessments as null

TRY
ASSIGN CourseID as value of JCourseID_
ASSIGN CourseName as value of JCourseName_
ASSIGN Duration as int value of JDuration_
ASSIGN Level as value of JLevel_
ASSIGN Credit as int value of JCredit_
ASSIGN NumberOfAssessments as int value of JNumberOfAssessments_
ASSIGN boolean SameAC as false

FOR all object C of Course in ArrayList AL

IF VALUE of method getCourseID() is equal to CourseID
ASSIGN SameAC to true

END FOR

IF SameAC is equal to false
CREATE new object AC of Academic Course (**PASS** parameters CourseID, CourseName, Duration, Level, Credit, NumberOfAssessments)
ADD AC in arraylist AL
DISPLAY MessageDialog in f "The Academic Course has been added."
END IF

ELSE
DISPLAY MessageDialog in f "The Academic Course has already been added."
END ELSE

END TRY

CATCH NumberFormatException e
DISPLAY ErrorMessageDialog in f "Please fill up the form properly."
END CATCH

END IF

ELSE IF VALUE of method getSource() is equal to Add1

ASSIGN String variable CourseID as empty
ASSIGN String variable CourseName as empty
ASSIGN int variable Duration as null
ASSIGN String variable Prerequisite as empty

TRY
ASSIGN CourseID as value of JCourseID1_
ASSIGN CourseName as value of JCourseName1_
ASSIGN Duration as int value of JDuration1_
ASSIGN Prerequisite as value of Prerequisite_
ASSIGN boolean SameNAC as false

FOR all object C of Course in ArrayList AL

IF VALUE of methodgetCourseID() is equal to CourseID
ASSIGN SameNAC to true
END IF

END FOR

IF SameNAC is equal to false
CREATE new object AC of NonAcademic Course (**PASS** parameters CourseID, CourseName, Duration, Prerequisite)
ADD NAC in arraylist AL
DISPLAY MessageDialog in f "The Non-Academic Course has been added."
END IF

ELSE
DISPLAY MessageDialog in f "The Non-Academic Course has already been added."
END ELSE

END TRY

CATCH NumberFormatException e
DISPLAY ErrorMessageDialog in f "Please fill up the form properly."
END CATCH

END ELSE IF

ELSE IF VALUE of method getSource() is equal to Register

ASSIGN String variable CourseID as empty
ASSIGN String variable CourseLeader as empty
ASSIGN String variable LecturerName as empty
ASSIGN String variable StartingDate as empty
ASSIGN String variable CompletionDate as empty

TRY
ASSIGN CourseID as value of JCourseID2_
ASSIGN CourseLeader as value of JCourseLeader_
ASSIGN LecturerName as value of JLecturerName_
ASSIGN StartingDate as value of JStartingDate_
ASSIGN CompletionDate as value of JCompletionDate_
ASSIGN boolean SameAC1 as false

FOR all object CO of Course in ArrayList AL

IF VALUE of method getCourseID() is equal to CourseID
ASSIGN SameAC1 as true

IF object CO is instance of Academic Course
DECLARE variable AC of type Academic Course, **CAST** object CO to Academic Course and **STORE** value in AC

IF VALUE of method getisRegistered() equals to true
DISPLAY MessageDialog in f “The Academic Course has already been added”
END IF

ELSE
CALL method Register (**PASS** parameters LecturerName, CourseLeader, StartingDate, CompletionDate)
DISPLAY MessageDialog in f “The Academic Course has been registered.”
END ELSE
END IF
END IF

ELSE

DISPLAY WarningMessageDialog in f “The CourseID do not match.”
END ELSE

END FOR

END TRY

CATCH NumberFormatException e
DISPLAY ErrorMessageDialog in f “Please fill up the form properly.”
END CATCH

END ELSE IF

ELSE IF value of method getSource() is equal to Register1

ASSIGN String variable CourseID as empty
ASSIGN String variable CourseLeader as empty
ASSIGN String variable InstructorName as empty
ASSIGN String variable StartingDate as empty
ASSIGN String variable CompletionDate as empty
ASSIGN String variable ExamDate as empty

TRY
ASSIGN CourseID as value of JCourseID3_
ASSIGN CourseLeader as value of JCourseLeader1_
ASSIGN LecturerName as value of JLecturerName_
ASSIGN StartingDate as value of JStartingDate1_
ASSIGN CompletionDate as value of JCompletionDate1_
ASSIGN ExamDate as value of JExamDate_
ASSIGN boolean SameNAC1 as false

FOR all object CO of Course in ArrayList AL

IF VALUE of method getCourseID() is equal to CourseID
ASSIGN SameNAC1 as true

IF object CO is instance of Academic Course
DECLARE variable NAC of type NonAcademic Course, **CAST** object CO to NonAcademic Course and **STORE** value in NAC

IF VALUE of method getisRegistered() equals to true

DISPLAY MessageDialog in f “The Non-Academic Course has already been added”

END IF

ELSE

CALL method Register (**PASS** parameters InstructorName, CourseLeader, StartingDate, CompletionDate, ExamDate)

DISPLAY MessageDialog in f “The Non-Academic Course has been registered.”

END ELSE

END IF

END IF

ELSE

DISPLAY WarningMessageDialog in f “The CourseID do not match.”

END ELSE

END FOR

END TRY

CATCH NumberFormatException e

DISPLAY ErrorMessageDialog in f “Please fill up the form properly.”

END CATCH

END ELSE IF

ELSE IF VALUE of getSource is equals to Remove

ASSIGN String variable CourseID as value of JCourseID4_

TRY

FOR all object C in ArrayList AL

IF VALUE of method getCourseID() is equal to CourseID

IF object C is instance of NonAcademicCourse

DECLARE variable NAC of type NonAcademicCourse, **CAST** object C to

NonAcademicCourse and **STORE** the value in NAC

IF VALUE of method getisRemoved is equal to false
CALL method Remove()
DISPLAY MessageDialog in f "The course has been removed."
END IF

ELSE IF VALUE of getisRemoved is equal to true
DISPLAY MessageDialog in f "The course has already been removed."
END ELSE IF
END IF
END IF

ELSE
DISPLAY WarningMessageDialog in f "Enter Valid CourseID."
END ELSE

END FOR

END TRY

CATCH Exception e
DISPLAY ErrorMessageDialog in f "Please fill up the form properly."
END CATCH

END ELSE IF

ELSE IF VALUE of method getSource() is equal to Display

FOR all object CO of Course in ArrayList AL

IF object CO is instance of AcademicCourse
DECLARE variable AC of type AcademicCourse **CAST** object CO to AcademicCourse and **STORE** the value in AC
CALL method display() of AcademicCourse
END IF
END FOR

END ELSE IF

ELSE IF VALUE of method getSource() is equal to Display1

FOR all object CO of Course in ArrayList AL

IF object CO is instance of NonAcademicCourse
DECLARE variable NAC of type NonAcademicCourse
CAST object CO to NonAcademicCourse and **STORE** the value in NAC
CALL method display() of NonAcademicCourse
END IF

END FOR

END ELSE IF

ELSE IF VALUE of method getSource() is equal to Clear

CALL method setText(**PASS** parameter empty String) for JCourseID_
CALL method setText(**PASS** parameter empty String) for JCourseName_
CALL method setText(**PASS** parameter empty String) for JDuration_
CALL method setText(**PASS** parameter empty String) for JLevel_
CALL method setText(**PASS** parameter empty String) for JCredit_
CALL method setText(**PASS** parameter empty String) for JNumberOfAssessments_
CALL method setText(**PASS** parameter empty String) for JCourseID2_
CALL method setText(**PASS** parameter empty String) for JLecturerName_
CALL method setText(**PASS** parameter empty String) for JCourseLeader_
CALL method setText(**PASS** parameter empty String) for JStartingDate_
CALL method setText(**PASS** parameter empty String) for JCompletionDate_

END ELSE IF

ELSE IF VALUE of method getSource() is equal to Clear1

CALL method setText(**PASS** parameter empty String) for JCourseID1_
CALL method setText(**PASS** parameter empty String) for JCourseName1_
CALL method setText(**PASS** parameter empty String) for JDuration1_

CALL method setText(**PASS** parameter empty String) for JPrerequisite_
CALL method setText(**PASS** parameter empty String) for JCourseID3_
CALL method setText(**PASS** parameter empty String) for JInstructorName_
CALL method setText(**PASS** parameter empty String) for JCourseLeader1_
CALL method setText(**PASS** parameter empty String) for JStartingDate1_
CALL method setText(**PASS** parameter empty String) for JCompletionDate1_
CALL method setText(**PASS** parameter empty String) for JExamDate_
CALL method setText(**PASS** parameter empty String) for JCourseID4_

END ELSE IF

ELSE IF VALUE of method getSource() is equal to NextPage

SET p1 visibility to False
SET p2 visibility to True

END ELSE IF

ELSE IF VALUE of method getSource() is equal to PreviousPage

SET p2 visibility to False
SET p1 visibility to True

END ELSE IF

CREATE method static void main (**PASS** parameter String args[ ])

CALL constructor INGCollege()

END METHOD

END CLASS

4. Method Description

There are two main methods used in INGCollege class with the methods of course, academic course and non-academic course. The methods are listed below:

- **INGCollege Class**

  - **actionPerformed (ActionEvent A)**

This method is used to handle the events that are made in the program. Without the action event, the buttons are rendered useless. There is an action event for many buttons in the GUI. They are given below.

1. **Add button for Academic Course**

   When the add button is clicked by the user, it gets the value of CourseID, Course name, duration, level, credit, and number of assessments. Then stores it in the object of Academic Course and the object is stored in the array list. A specific message will be shown saying the academic course has been added. If the CourseId is already added then a specific message is displayed. Similarly, if the user inputs the wrong datatype the specific message will be shown.

2. **Add button for Non-Academic Course**

   Similarly, as the add button for the academic course when the add button is clicked by the user, it gets the value of CourseID, Course name, duration and prerequisite. Then stores it in the object of Non-Academic Course and the object is stored in the array list. A specific message will be shown saying the non-academic course has been added. If the CourseId is already added then a specific message is displayed. Similarly, if the user inputs the wrong datatype the specific message will be shown.

3. **Register button for Academic Course**

When the register button is clicked by the user, it gets the value of Course Leader, Lecturer Name, Starting Date, and Completion Date and stores it in the object of Academic Course. Then the object is stored in the array list. A specific message will be shown saying the academic course has been registered. The method Register from Academic Course is used to register the course. If the same CourseID is added, a message will be shown that it is already registered. If a different CourseID that is not added is inputted for registration, a specific message will be shown. Finally, if the user inputs the wrong datatype the specific message will be shown.

4. **Register button for Non-Academic Course**

When the register button is clicked by the user, it gets the value of Course Leader, Instructor Name, Starting Date, Completion Date, and ExamDate and stores it in the object of Non-Academic Course. Then the object is stored in the array list. A specific message will be shown saying the non-academic course has been registered. The method Register from Non-Academic Course is used to register the course. If the same CourseID is added, a message will be shown that it is already registered. If a different CourseID that is not added is inputted for registration, a specific message will be shown. Finally, if the user inputs the wrong datatype the specific message will be shown.

5. **Remove button for Non-Academic Course**

When this button is clicked, the inputted CourseID will be removed and a message will be shown. The method Remove from Non-Academic Course is used to remove the course. If the CourseID is

already removed, it will display the specific message and similarly, if the inputted CourseID is not registered then a suitable message will be shown.

6. **Display button for Academic Course**

   When this button is clicked, the data stored in the object of Academic Course is displayed in the terminal.

7. **Display button for Non-Academic Course**

   When this button is clicked, the data stored in the object of Non-Academic Course is displayed in the terminal.

8. **Clear button for Academic Course**

   When the button is clicked, the values inputted in the text fields of panel p1 are cleared. A method set text is used to clear the values of the text fields.

9. **Clear button for Non-Academic Course**

   When the button is clicked, the values inputted in the text fields of panel p2 are cleared. A method set text is used to clear the values of the text fields.

10. **Next Page button**

    When this button is clicked, the visibility of p2 is set to true and p1 is set to false. This helps to input the values for the non-academic course.

11. **Previous Page button**

    When this button is clicked, the visibility of p1 is set to true and p2 is set to false. This helps to input the values for the academic course.

- **Main method**

  This method is used to create an object of the INGCollege. All the codes written within the constructor of INGCollege are called through this method.

- **Course Class**

  - **getCourseID():**

    This method is used to access the private attributes and return the values of CourseID. It returns the values in String datatype.

  - **getCourseName():**

    This method is used to access the private attributes and return the values of CourseName. It returns the values in String datatype.

  - **getCourseDuration():**

    This method is used to access the private attributes and return the values of CourseDuration. It returns the values in int datatype.

  - **getCourseLeader():**

    This method is used to access the private attributes and return the values of CourseLeader. It returns the values in String datatype.

  - **setCourseLeader(newCourseLeader):**

    This method is used to set the value of CourseLeader. It takes a parameter and assigns it as String datatype.

  - **Display():**

    This method is used to print out the values of the course class. It displays the details of the course class

- **AcademicCourse Class**

  - **getLecturerName():**
    This method is used to access the private attributes and return the values of LecturerName. It returns the values in String datatype.

  - **getLevel():**
    This method is used to access the private attributes and return the values of Level. It returns the values in String datatype.

  - **getCredit()**
    This method is used to access the private attributes and return the values of Credit. It returns the values in String datatype.

  - **getStartingDate():**
    This method is used to access the private attributes and return the values of StartingDate. It returns the values in String datatype.

  - **getCompletionDate():**
    This method is used to access the private attributes and return the values of CompletionDate. It returns the values in String datatype.

  - **getNumberOfAssessments():**
    This method is used to access the private attributes and return the values of NumberOfAssessments. It returns the values in int datatype.

  - **getisRegistered():**
    This method is used to access the private attributes and return the values of isRegistered. It returns the values in boolean datatype.

- **setLecturerName(newLecturerName):**
  This method is used to set the value of LecturerName. It takes a parameter and assigns it as String datatype.

- **setNumberOfAssessments(newNumberOfAssessments):**
  This method is used to set the value of NumberOfAssessments. It takes a parameter and assigns it as an int datatype.

- **Register(String CourseLeader, String LecturerName, String StartingDate, String Completion Date)**
  This method is used to register a course. It takes parameters and checks if it is registered then sets the value from the parameters if it is not registered.

- **Display():**
  This method is used to print out the values of the AcademicCourse class. This method also uses the display method from its parent class. It displays the details of the AcademicCourse class.

- **NonAcademicCourse Class**

  - **getInstructorName():**

    This method is used to access the private attributes and return the values of InstructorName. It returns the values in String datatype.

  - **getStartingDate():**

    This method is used to access the private attributes and return the values of StartingDate. It returns the values in String datatype.

- **getCompletionDate():**

  This method is used to access the private attributes and return the values of CompletionDate. It returns the values in String datatype.

- **getExamDate():**

  This method is used to access the private attributes and return the values of ExamDate. It returns the values in String datatype.

- **getPrerequisite():**

  This method is used to access the private attributes and return the values of Prerequisite. It returns the values in String datatype.

- **getisRegistered():**

  This method is used to access the private attributes and return the values of isRegistered. It returns the values in boolean datatype.

- **getisRemoved():**

  This method is used to access the private attributes and return the values of isRemoved. It returns the values in boolean datatype.

- **setInstructorName(newInstructorName):**
  This method is used to set the value of InstructorName. It takes a parameter and assigns it as String datatype.

- **Register (String CourseLeader, String InstructorName, String StartingDate, String CompletionDate, String ExamDate)**

This method is used to register a Non-Academic course. It takes parameters and checks if it is registered then sets the value from the parameters if it is not registered.

- **Remove():**

  This method is used to remove a course. If the course has already been removed it will display a certain message if not then it will remove the course from the records.

- **Display():**

  This method is used to print out the values of the NonAcademicCourse class. This method also uses the display method from its parent class. It displays the details of the NonAcademicCourse class.

5. Testing

5.1 Test 1 – To test the program can be compiled and run using the command prompt.

Test Number	1
Objective:	To test the program can be compiled and run using the command prompt.
Action:	→ The command prompt is opened and the directory is changed to where the file is located. → javac INGCollege.java is inputted in the command prompt. → java INGCollege is inputted in the command prompt.
Expected Result:	The program should run and the GUI should be displayed.
Actual Result:	The program ran and the GUI was displayed.
Conclusion:	The test is successful.

Table 1 Test Table 1

Output Result:

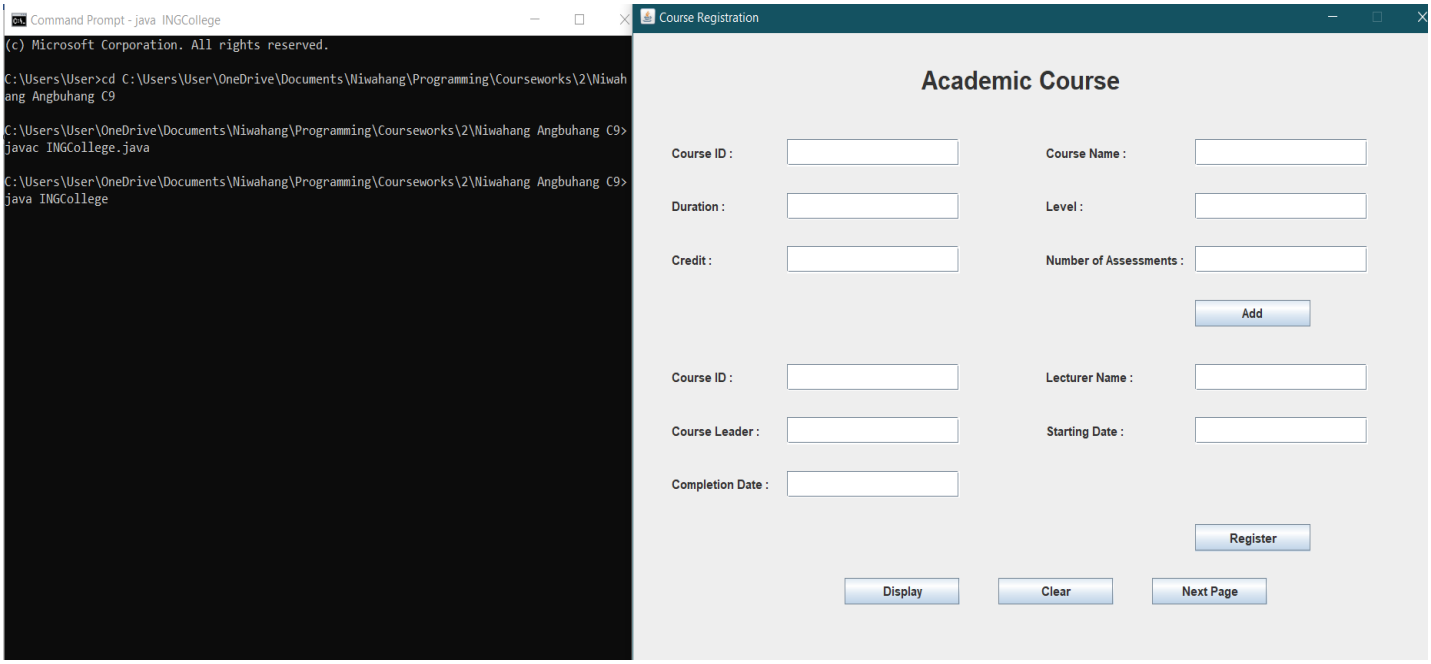

Figure 6 Command Prompt Test

5.2 Test 2(a) – To add and register Course for Academic Course

Test Number	2(a)
Objective:	To add and register a course for Academic Course
Action:	→ The following values are inserted in the text fields of the Academic Course panel. Course ID = 103 Course Name = Psychology Course Duration = 3 Level = 4 Credit = 120 Number of Assessments = 6 → The Add button is clicked → The following values are inserted in the text fields of the Academic Course panel. Course ID = 103 Lecturer Name = Jose Course Leader = Savic Starting Date = 12th May 2021 Completion Date = 28th April 2024 → The Register button is clicked. → The Display button is clicked.
Expected Result:	The Academic Course should be added and registered.
Actual Result:	The Academic Course was added and registered.
Conclusion:	The test is successful.

Table 2 Test Table 2

Output Result:

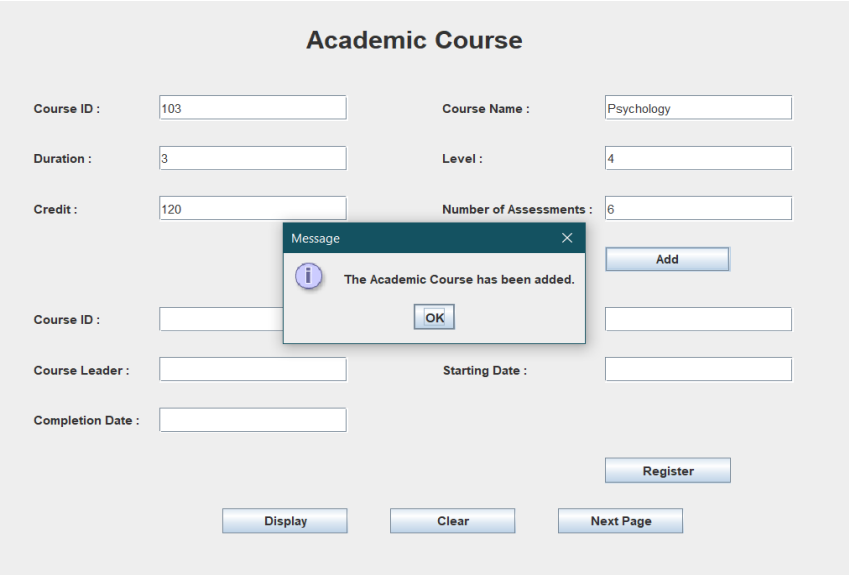

Figure 7 Academic Course is added

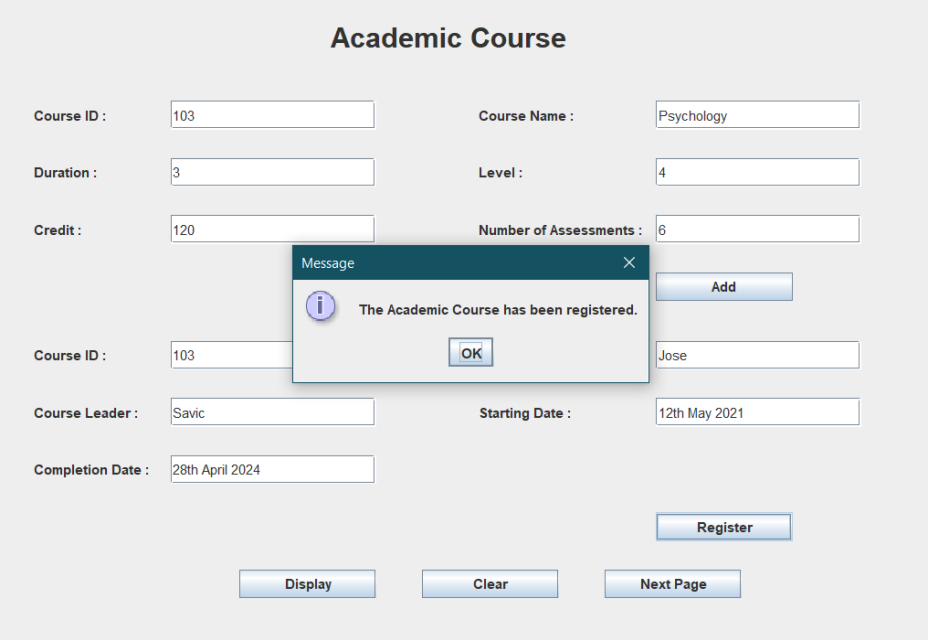

Figure 8 Academic Course is registered

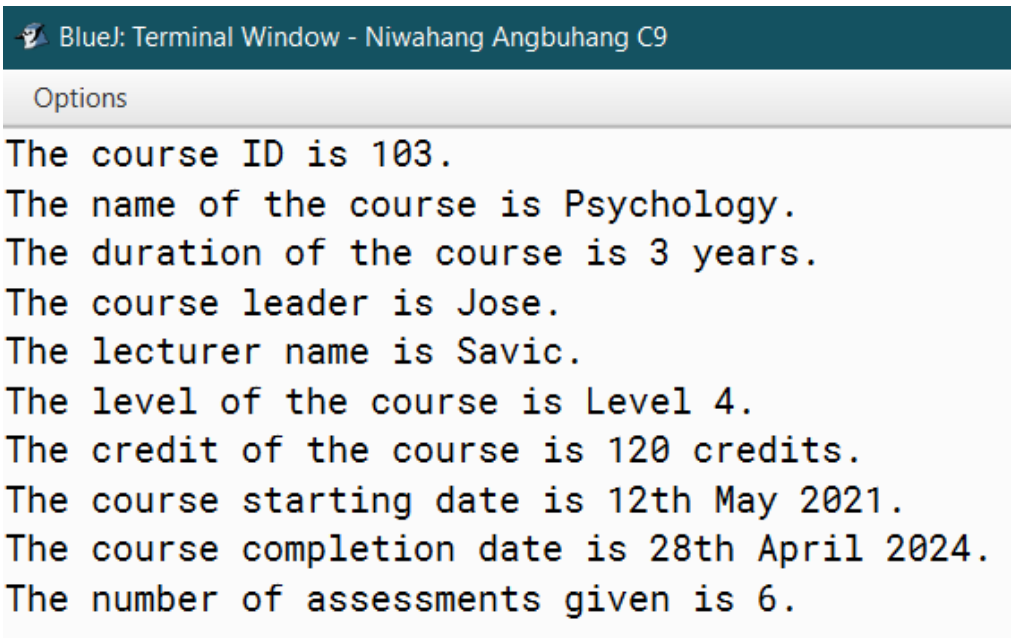

Figure 9 Academic Course is displayed

5.3 Test 2(b) – To add, register, and remove course for Non-Academic Course

Test Number	2(b)
Objective:	To add and register and remove course for Non-Academic Course
Action:	→ Next page button is clicked. → The following values are inserted in the text fields of the Non-Academic Course panel. Course ID = 321 Course Name = C++ Course Duration = 2 Prerequisite = Fundamentals of Programming → The Add button is clicked → The following values are inserted in the text fields of the Non-Academic Course panel. Course ID = 321 Instructor Name = Pooran Course Leader = Rima Starting Date = 21st April 2021 Completion Date = 16th February 2023 Exam Date = 8th March 2023 → The Register button is clicked. → The following values are inserted in the text fields of the Non-Academic Course panel. Course ID = 321 → The Display button is clicked. → The Remove button is clicked. → The Display button is clicked.
Expected Result:	The Non-Academic Course should be added, registered, and removed.
Actual Result:	The Non-Academic Course was added, registered, and removed.
Conclusion:	The test is successful.

Table 3 Test Table 3

Output Result:

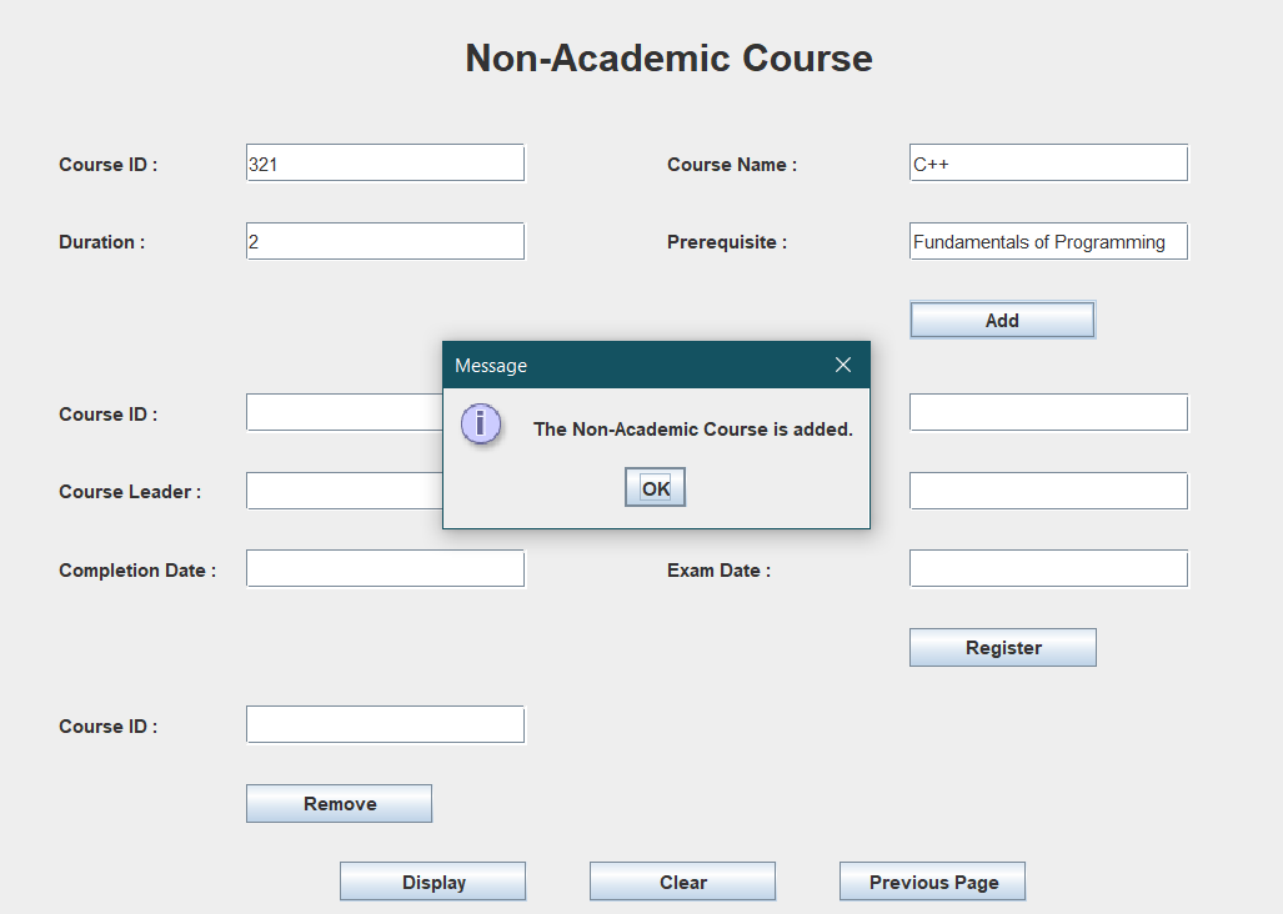

Figure 10 Non-Academic Course is added

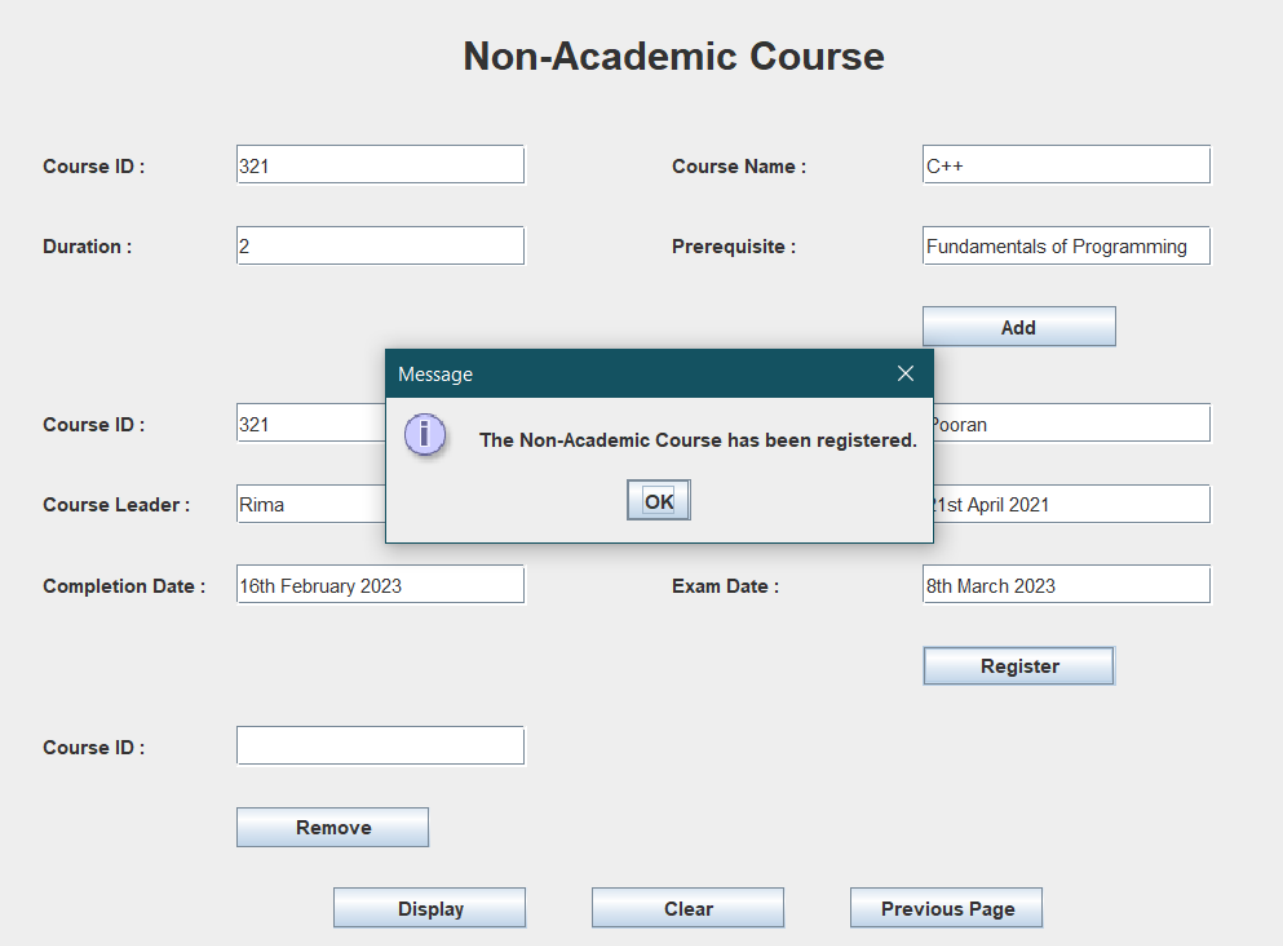

Figure 11 Non-Academic Course is registered.

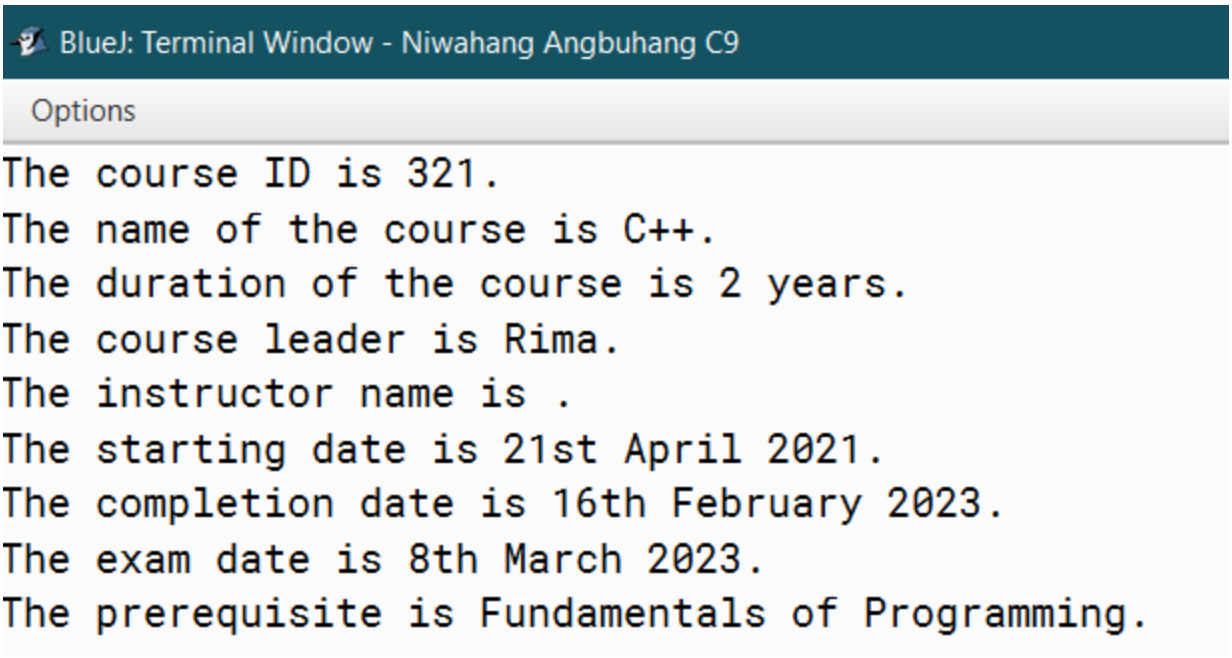

Figure 12 Non-Academic course is displayed

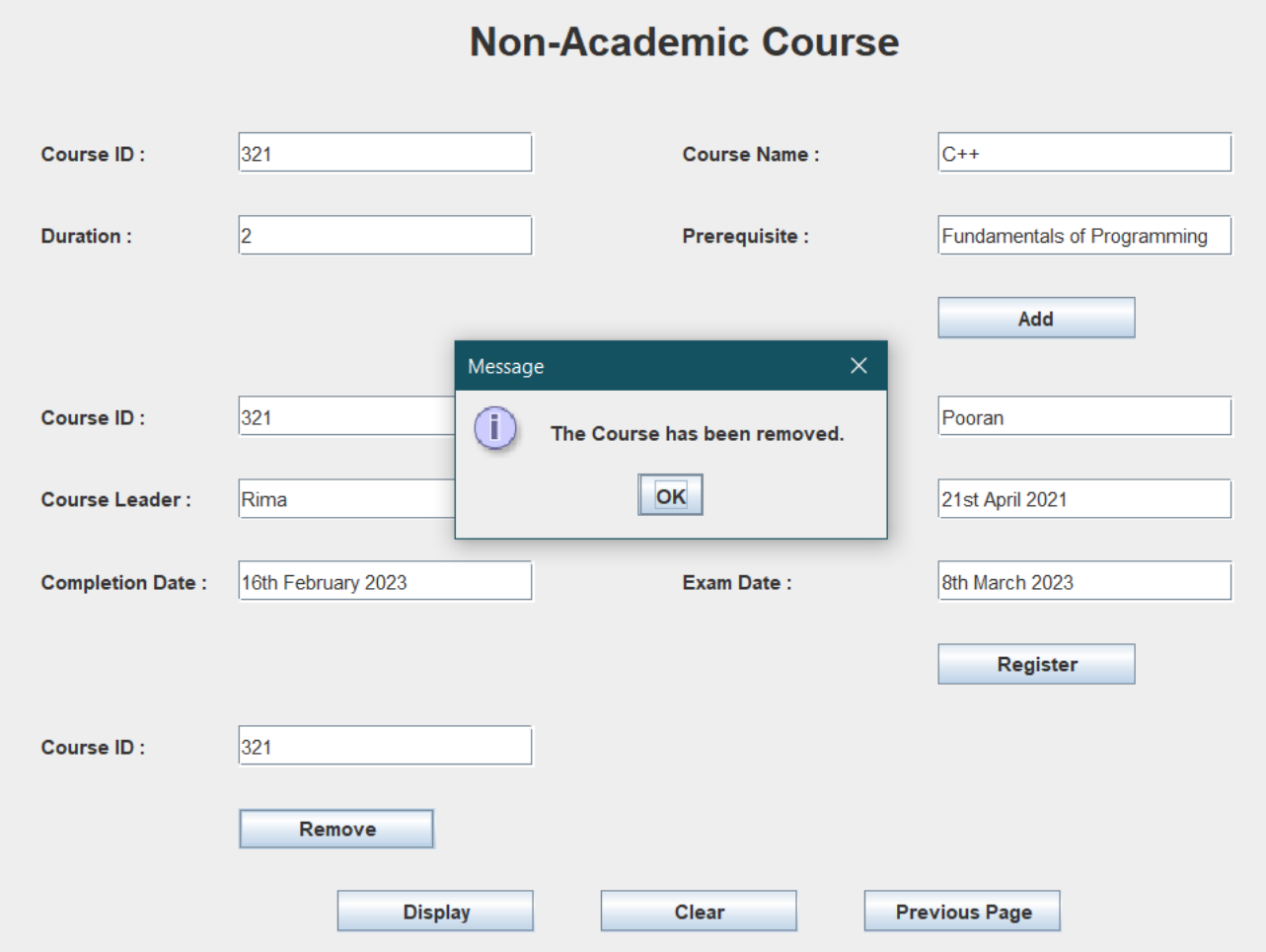

Figure 13 Non-Academic course is removed

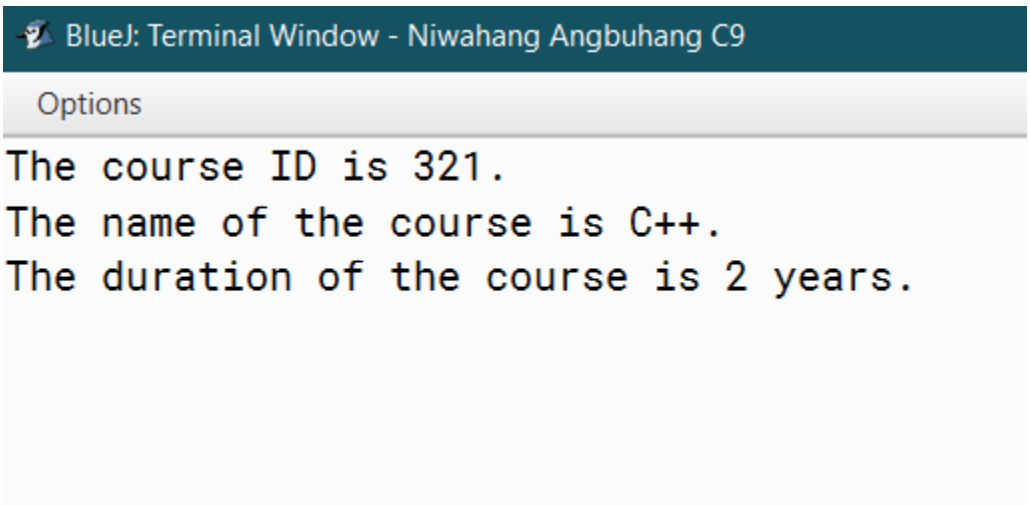

Figure 14 Non-Academic course is displayed after removing

5.4 Test 3(a) – To add duplicate Course ID

Test Number	**3(a)**
Objective:	To add duplicate Course ID
Action:	→ The following values are inserted in the text fields of the Academic Course panel. Course ID = 103 Course Name = Psychology Course Duration = 3 Level = 4 Credit = 120 Number of Assessments = 6 → The Add button is clicked → The following values are inserted in the text fields of the Academic Course panel. Course ID = 103 Course Name = Business Marketing Course Duration = 2 Level = 5 Credit = 60 Number of Assessments = 3 → The Add button is clicked
Expected Result:	A dialog box should appear saying the course is already added.
Actual Result:	A dialog box appeared saying the course is already added.
Conclusion:	The test is successful.

Table 4 Test Table 4

Output Result:

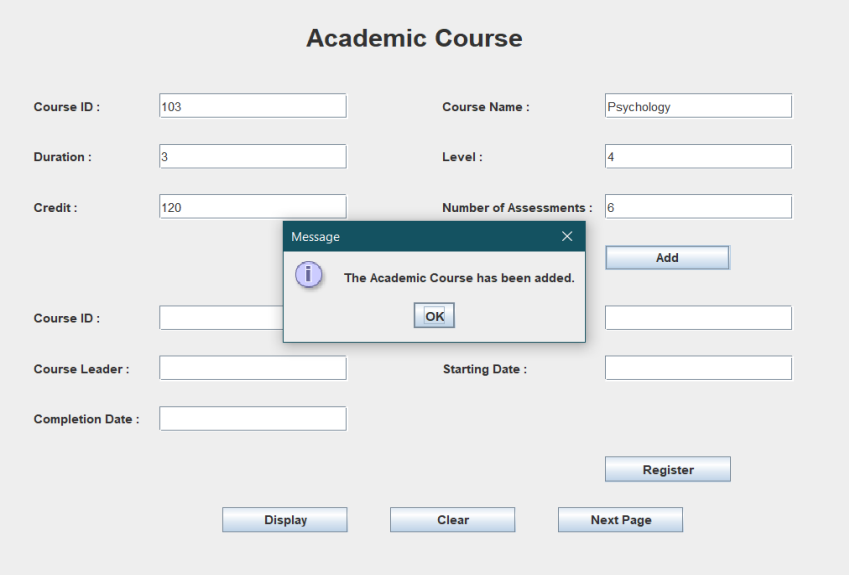

Figure 15 Academic Course is added

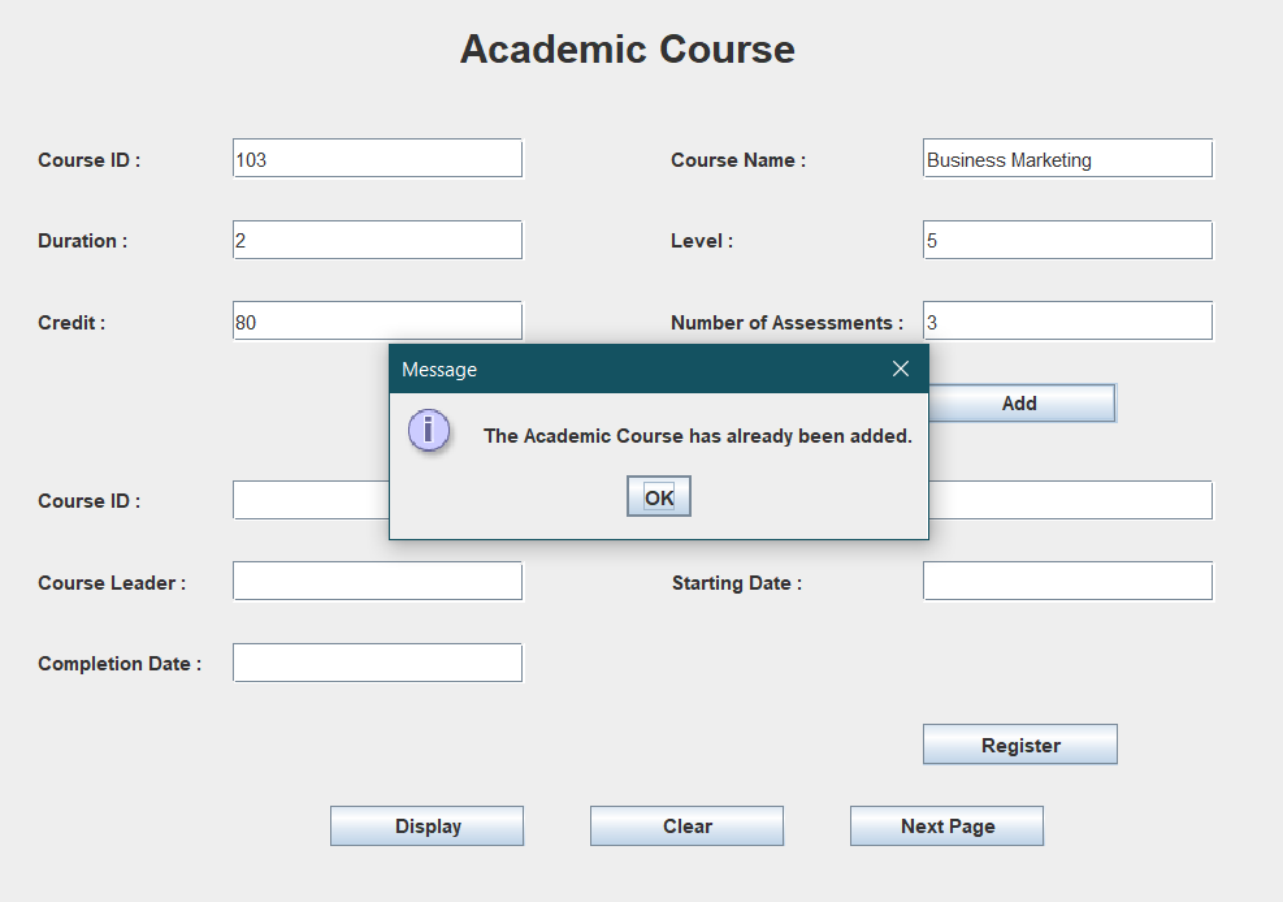

Figure 16 Academic Course is added again

5.5 Test 3(b) – To register already registered course

Test Number	**3(b)**
Objective:	To register already registered course
Action:	→ The following values are inserted in the text fields of the Academic Course panel. Course ID = 103 Course Name = Psychology Course Duration = 3 Level = 4 Credit = 120 Number of Assessments = 6 → The Add button is clicked → The following values are inserted in the text fields of the Academic Course panel. Course ID = 103 Lecturer Name = Jose Course Leader = Savic Starting Date = 12th May 2021 Completion Date = 28th April 2024 → The Register button is clicked. → The Register button is clicked again.
Expected Result:	A dialog box should appear saying the course is already registered.
Actual Result:	A dialog box appeared saying the course is already registered.
Conclusion:	The test is successful.

Table 5 Test Table 5

Output Result:

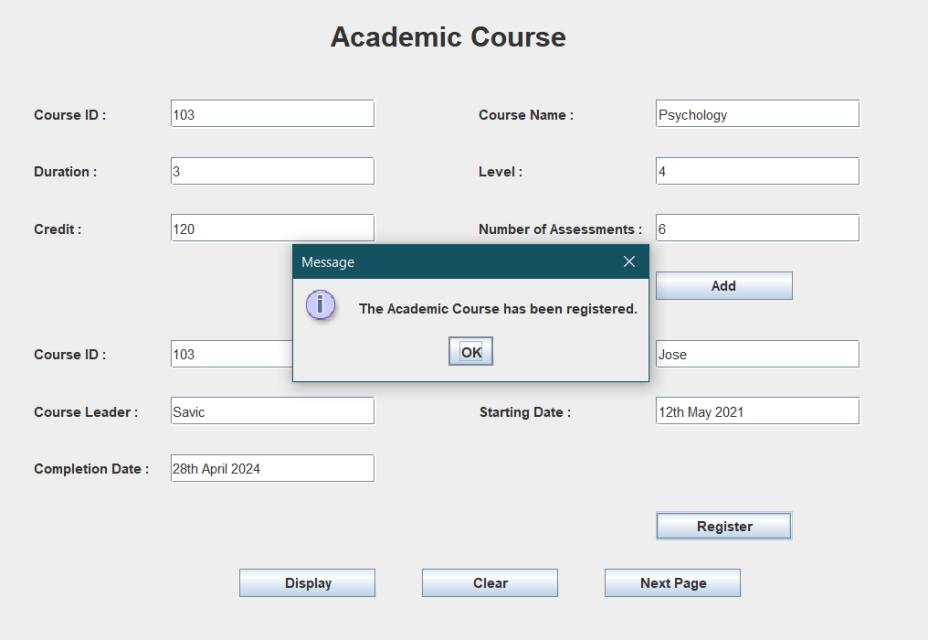

Figure 17 Academic Course is registered

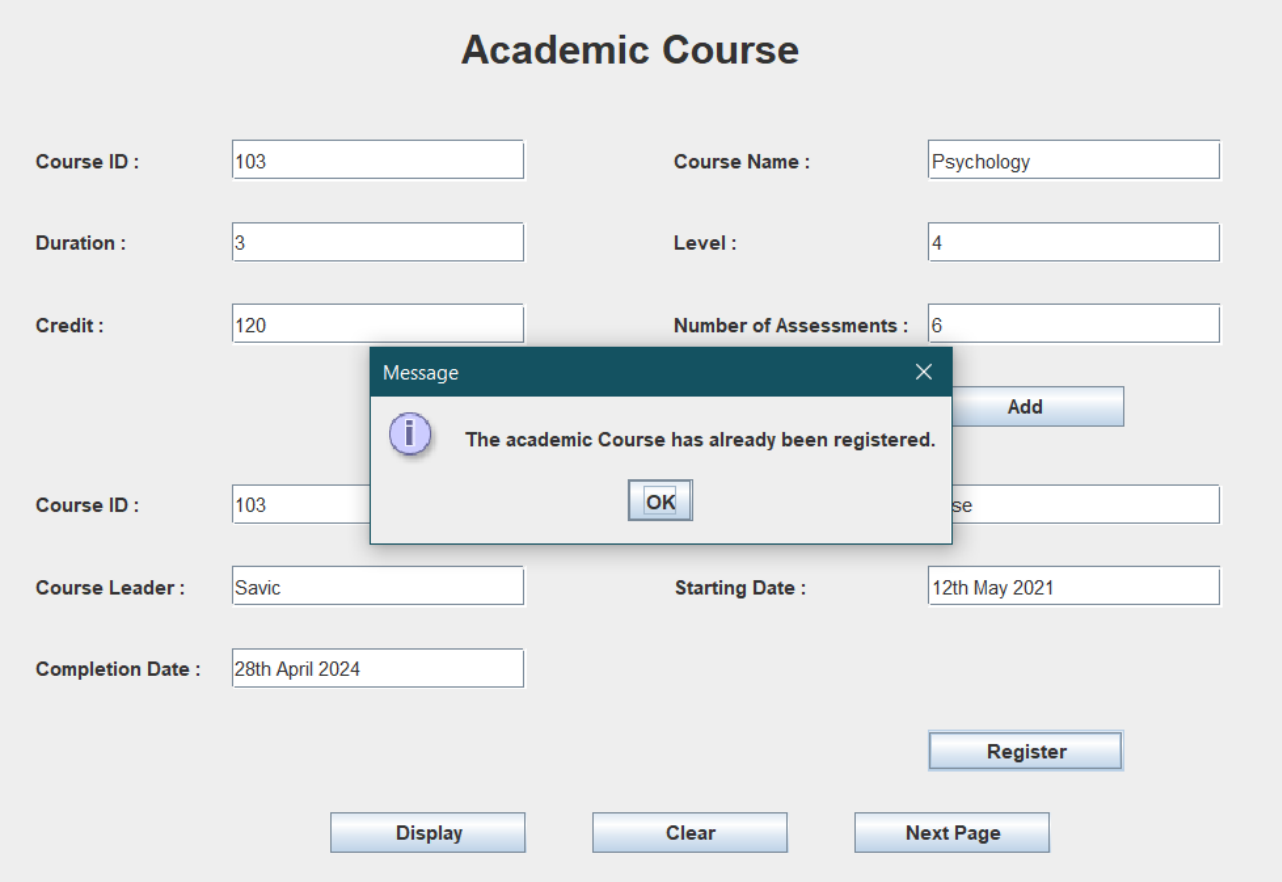

Figure 18 Academic Course is registered again

5.6 Test 3(c) – To remove Non-Academic Course which is already removed

Test Number	**3(c)**
Objective:	To remove Non-Academic Course which is already removed
Action:	→ Next page button is clicked. → The following values are inserted in the text fields of the Non-Academic Course panel. Course ID = 321 Course Name = C++ Course Duration = 2 Prerequisite = Fundamentals of Programming → The Add button is clicked → The following values are inserted in the text fields of the Non-Academic Course panel. Course ID = 321 Instructor Name = Pooran Course Leader = Rima Starting Date = 21st April 2021 Completion Date = 16th February 2023 Exam Date = 8th March 2023 → The Register button is clicked. → The following values are inserted in the text fields of the Non-Academic Course panel. Course ID = 321 → The Remove button is clicked. → The Remove button is clicked again
Expected Result:	A dialog box should appear saying the course is already removed.
Actual Result:	A dialog box appeared saying the course is already removed.
Conclusion:	The test is successful.

Table 6 Test Table 6

Output Result:

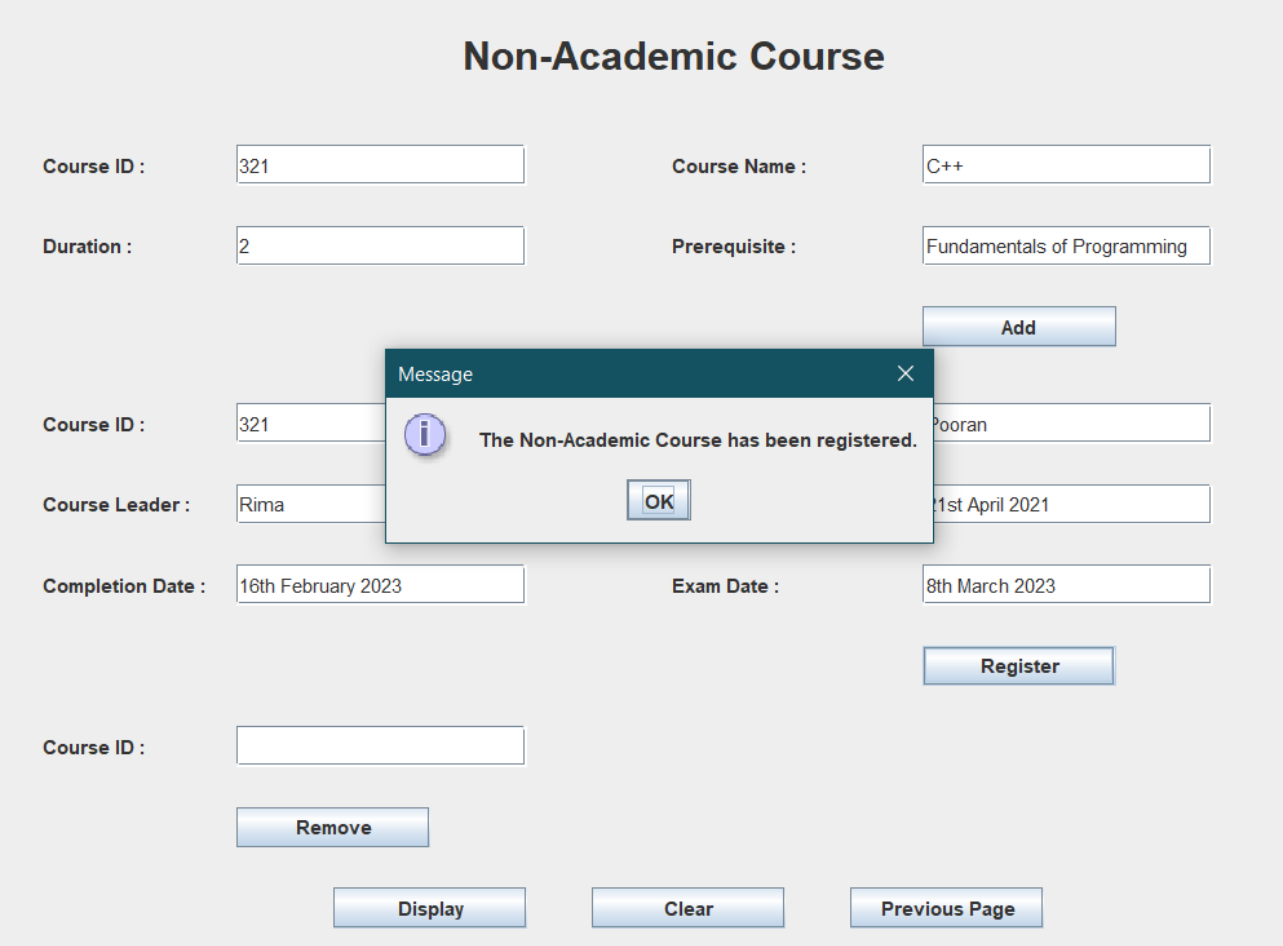

Figure 19 Non-Academic Course is registered

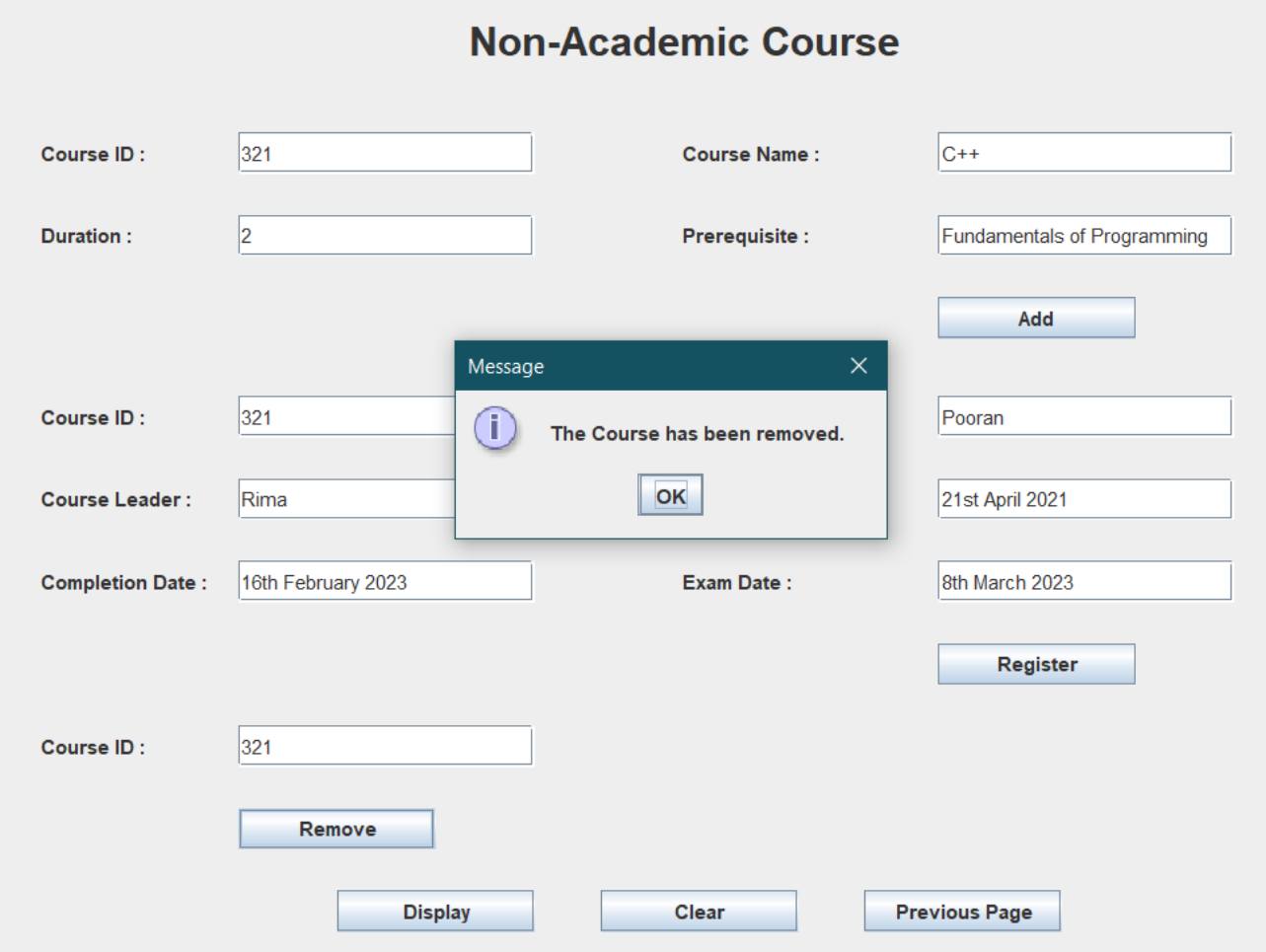

Figure 20 Non-Academic Course is removed

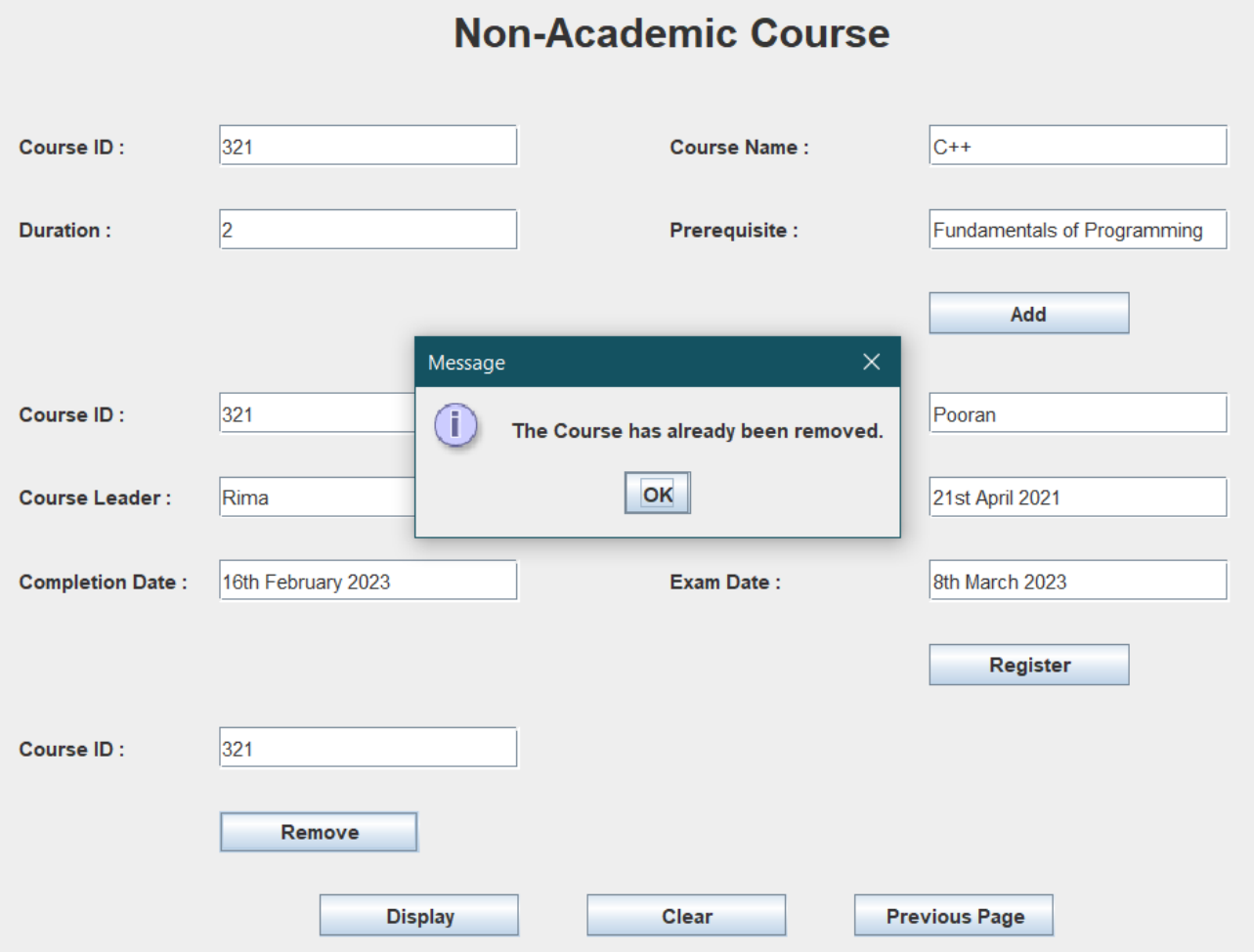

Figure 21 Non-Academic Course is removed again

6. Different Error Detection and Correction

There were some errors encountered during the coding process.

6.1 Syntax Error

Syntax error occurred when doing most of the coding. The semi-colon was forgotten at the end to terminate a line. It was fixed by adding the semi-colon at the end of the line.

```
JDuration = new JLabel("Duration : ");
JLevel = new JLabel("Level : ");
JCredit = new JLabel("Credit : ")
JNumberOfAssessments = new JLabel("Number o
```

Figure 22 Syntax Error Detection

```
JDuration = new JLabel("Duration : "
JLevel = new JLabel("Level : ");
JCredit = new JLabel("Credit : ");
JNumberOfAssessments = new JLabel("N
```

Figure 23 Syntax Error Correction

6.2 Semantic Error

When writing the GUI, a semantic error occurred when course duration was mistakenly assigned as a string after parsing it to int. It was fixed by correcting the string to int.

```
String Duration="";
String Prerequisite = "";
try
{
    CourseID = JCourseID1_.getText();
    CourseName = JCourseName1_.getText();
    Duration = Integer.parseInt(JDuration1_.getText());
    Prerequisite = JPrerequisite_.getText();
```

Figure 24 Semantic Error Detection

```
String CourseName = "";
int Duration=0;
String Prerequisite = "";
try
{
    CourseID = JCourseID1_.getText();
    CourseName = JCourseName1_.getText();
    Duration = Integer.parseInt(JDuration1_.getText());
    Prerequisite = JPrerequisite_.getText();
    boolean SameNAC = false;
```

Figure 25 Semantic Error Correction

6.3 Logical Error

An error occurred when running the program. When running the program, the program was not displayed accordingly to the code that was written. Later, I found out that I had not set the layout of the frame and panels. I fixed the error by setting the layouts of the frame and panels to null.

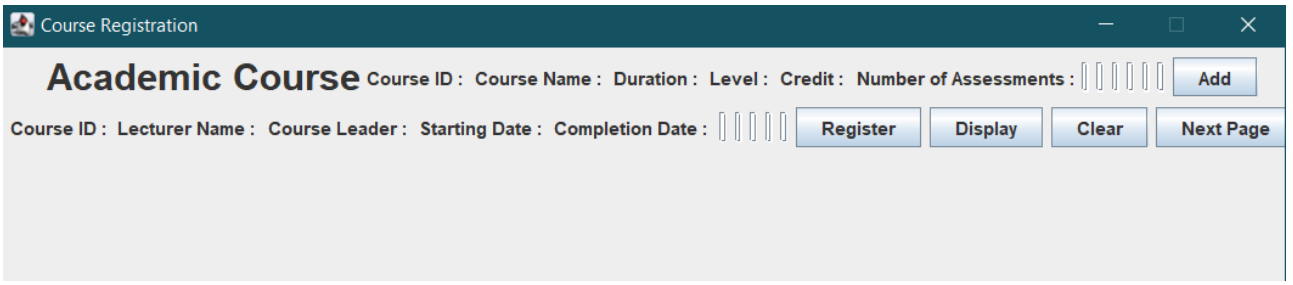

Figure 26 Logical Error Detection

```
f.setLayout(null);
p1.setLayout(null);
p2.setLayout(null);
```

Figure 27 Logical Error Correction

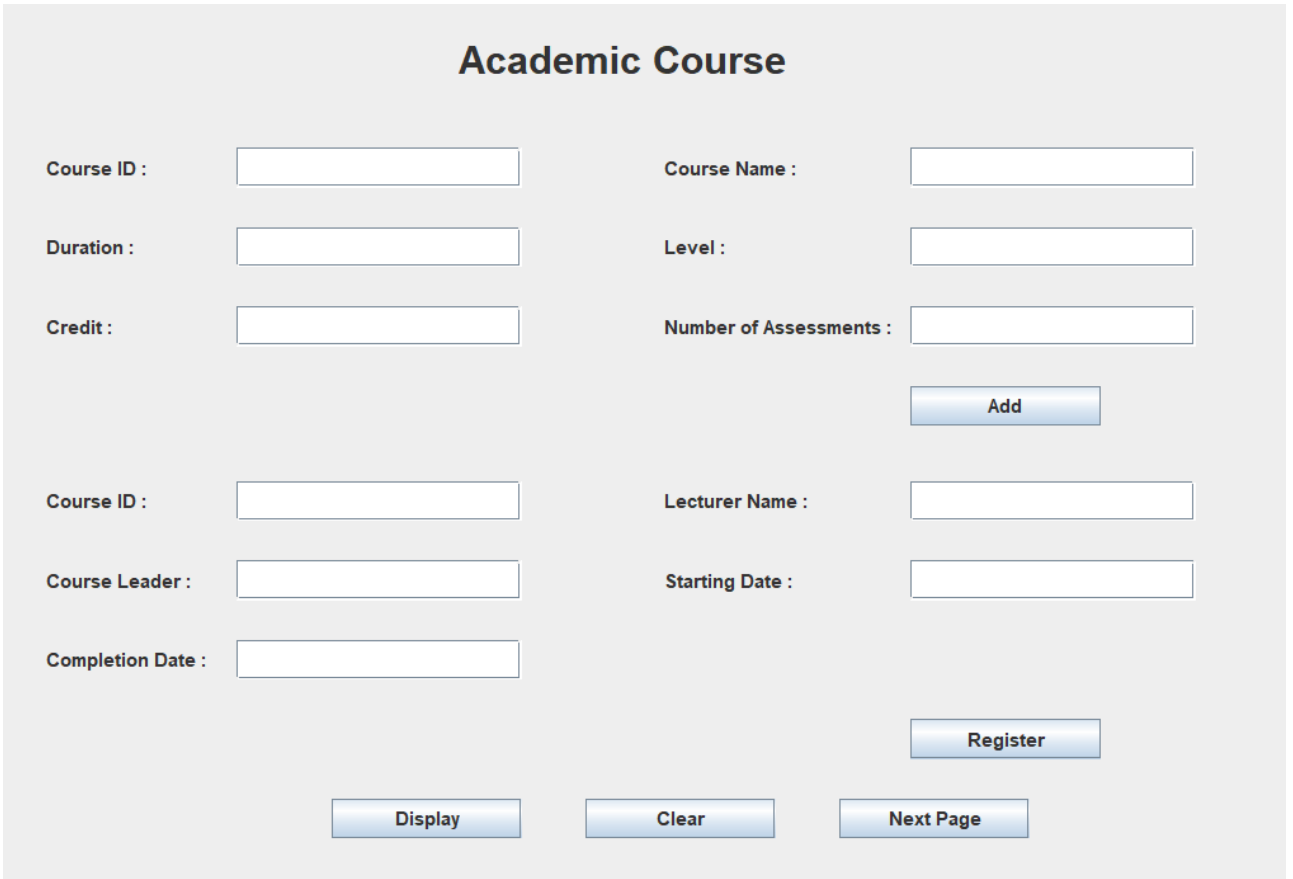

Figure 28 Logical Error Correction(2)

6.4 Run Time Error

When entering data in the GUI, a string value was entered instead of an integer data type. An error occurred in the terminal. The error was fixed by adding the code inside the try and catch block.

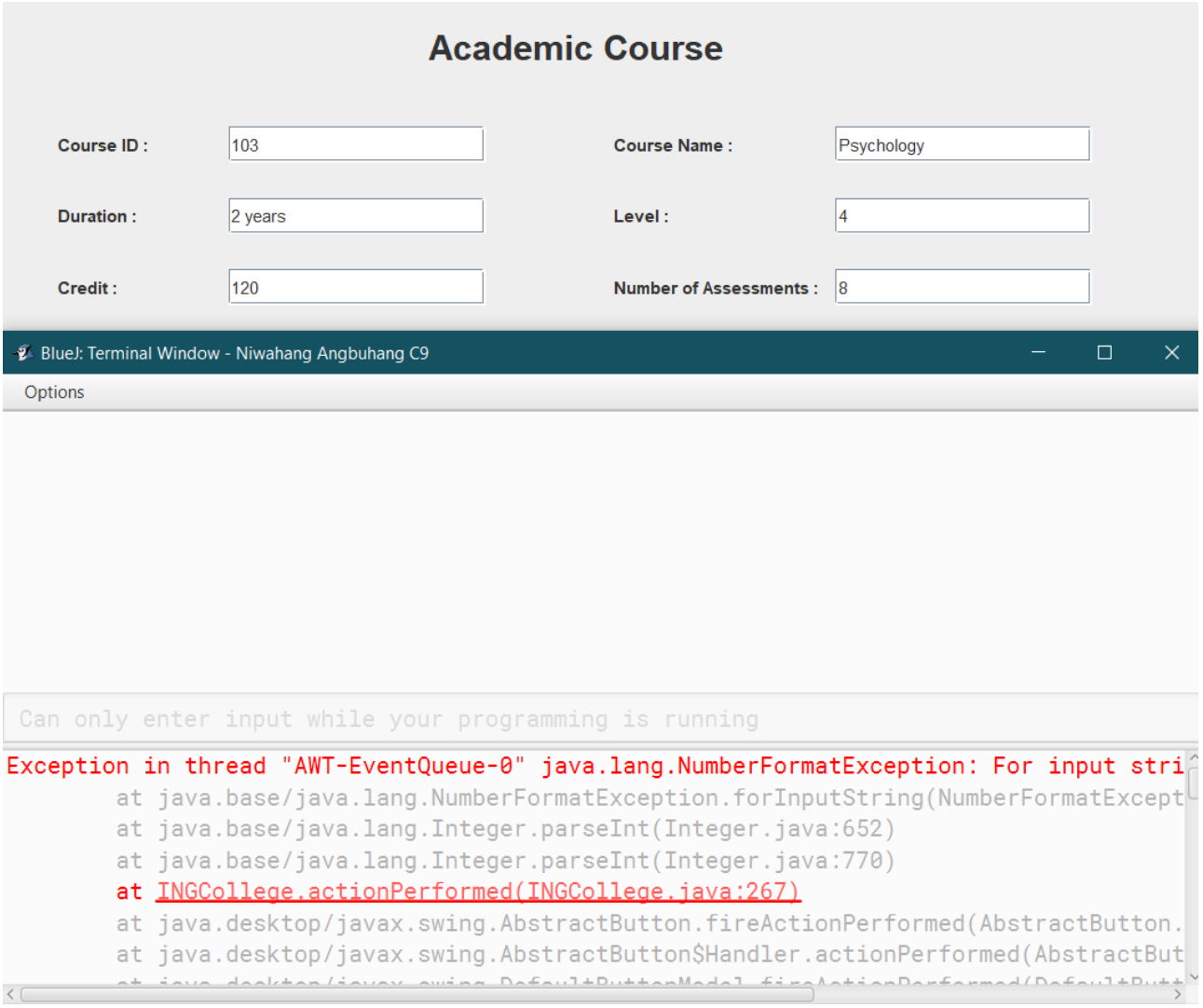

Figure 29 Run Time Error

```
}
catch(NumberFormatException e)
{
    JOptionPane.showMessageDialog(p1,"Please fill up the form properly !","ERROR",JOptionPane.ERROR_MESSAGE);
}
```

Figure 30 Run Time Error Correction

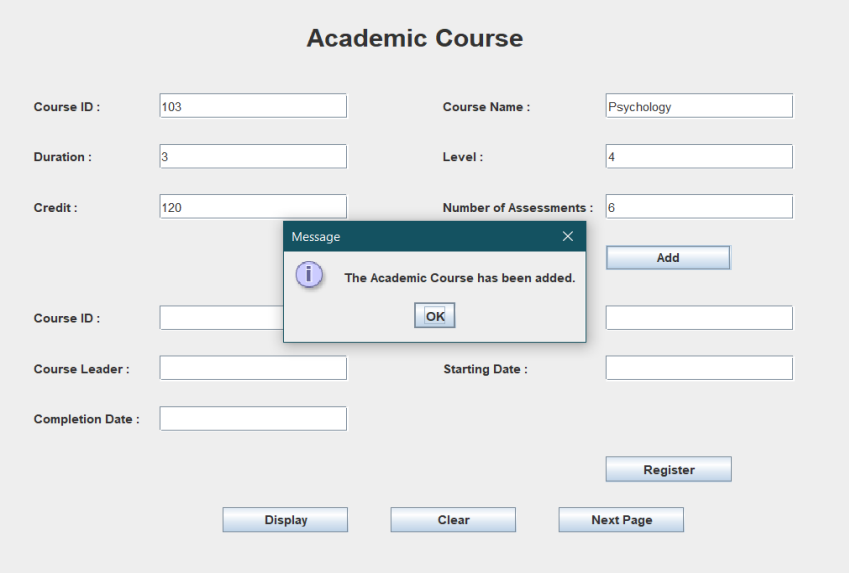

Figure 31 Run Time Error Correction (2)

7. Conclusion

Java is a programming language and platform for computing. It allows us to create many useful applications. Many applications we use widely are made using Java as the programming language. It is valuable in making applications that serve the most recent innovative advancement. The coursework we were given focuses mostly on the GUI Interface.

At first, I got somewhat restless and anxious too, when I saw the second coursework. When compared with the first coursework, the second coursework was hard for me to finish on schedule. I was stressed that if I could complete my coursework on schedule. In the first coursework, we were approached to register courses for academic and non-academic courses. This coursework is in one way or another as the first coursework. In this coursework, we were approached to make a GUI for the Academic and Non-Academic Course. We were assigned to make action buttons to add, register, and remove the courses. We needed to use JPanels, JFrame, JTexFields, JLabels etc to make a design for the GUI. These themes were new for me. The hardest part for me to finish this coursework was to make a design for the GUI. A lot of time was consumed to pick a design so that it looks user-friendly and easy to use. However, I had a great time doing this coursework. This was my first GUI program used in the real world.

There were tons of challenges while doing the coursework. The subject was practically new for me, as I said previously, however with the steady help of the teachers and going through the internet I was able to finish the coursework on schedule. It was troublesome as the entire world was battling with COVID-19. The stage was truly hard for us all. It was difficult for me to do this coursework. Online classes are not as effective as physical classes. So, I had to watch the lecture, tutorial, and workshop recordings of the class many times to understand. There were a few bugs while doing this coursework however with the assistance of the teachers and some of my friends, I was able to debug the bugs. This coursework has increased my knowledge of Java and GUI much more than before.

Appendix

Course Class

```
public class Course
{
	private String CourseID;
	private String CourseName;
	private String CourseLeader;
	private int CourseDuration;

	public Course(String CourseID, String CourseName, int CourseDuration)
	{
	    this.CourseID = CourseID;
	    this.CourseName = CourseName;
	    this.CourseDuration = CourseDuration;
	    CourseLeader = "";
	}

	public String getCourseID()
	{
	    return CourseID;
	}

	public String getCourseName()
	{
	    return CourseName;
	}

	public String getCourseLeader()
	{
	    return CourseLeader;
	}

	public int getCourseDuration()
	{
	    return CourseDuration;
	}

	public void setCourseLeader(String newCourseLeader)
	{
	    this.CourseLeader = newCourseLeader;
	}
```

```
    public void Display()
    {
        System.out.println("The course ID is " + CourseID + ".");
        System.out.println("The name of the course is " + CourseName + ".");
        System.out.println("The duration of the course is " + CourseDuration + " years.");
        if (CourseLeader!="")
        {
            System.out.println("The course leader is " + CourseLeader + ".");
        }
    }
}
```

AcademicCourse Class

```
public class AcademicCourse extends Course
{
    private String LecturerName;
    private String Level;
    private int Credit;
    private String StartingDate;
    private String CompletionDate;
    private int NumberOfAssessments;
    private boolean isRegistered;

    public AcademicCourse(String CourseID, String CourseName, int CourseDuration,
String Level, int Credit, int NumberOfAssessments)
    {
        super(CourseID, CourseName, CourseDuration);
        this.Level = Level;
        this.NumberOfAssessments = NumberOfAssessments;
        this.Credit = Credit;
        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 newLecturerName)
    {
        this.LecturerName = newLecturerName;
    }

    public void setNumberOfAssessments(int newNumberOfAssessments)
    {
        this.NumberOfAssessments = newNumberOfAssessments;
    }

    public void Register(String CourseLeader, String LecturerName, String StartingDate,
String CompletionDate)
    {
        if(isRegistered == true)
        {
            System.out.println("The academic course has already been registered.");
            System.out.println("The lecturer name is " + LecturerName + ".");
            System.out.println("The course starting date is " + StartingDate + ".");
            System.out.println("The course completion date is " + CompletionDate + ".");
        }
```

```
        else
        {
            super.setCourseLeader(CourseLeader);
            this.LecturerName = LecturerName;
            this.StartingDate = StartingDate;
            this.CompletionDate = CompletionDate;
            isRegistered = true;
        }
    }

    public void display()
    {
        super.Display();
        if(isRegistered == true)
        {
            System.out.println("The lecturer name is " + LecturerName + ".");
            System.out.println("The level of the course is Level " + Level + ".");
            System.out.println("The credit of the course is " + Credit + " credits.");
            System.out.println("The course starting date is " + StartingDate + ".");
            System.out.println("The course completion date is " + CompletionDate + ".");
            System.out.println("The number of assessments given is " +
NumberOfAssessments + ".");
        }
    }
}
```

NonAcademicCourse Class

```
public class NonAcademicCourse extends Course
{
    private String InstructorName;
    private String StartingDate;
    private String CompletionDate;
    private String ExamDate;
    private String Prerequisite;
    private boolean isRegistered;
    private boolean isRemoved;

    public NonAcademicCourse(String CourseID, String CourseName, int CourseDuration,
String Prerequisite)
    {
        super(CourseID, CourseName, CourseDuration);
        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;
    }

    public boolean getisRegistered()
    {
        return isRegistered;
    }

    public boolean getisRemoved()
    {
        return isRemoved;
    }

    public void setInstructorName(String newInstructorName)
    {
        if(isRegistered == true)
        {
            System.out.println("Since the instructor name is already registered, it is not
possible to change it.");
        }
        else
```

```
    {
        this.InstructorName = newInstructorName;
    }
}

public void Register(String CourseLeader, String InstructorName, String StartingDate,
String CompletionDate, String ExamDate)
{
    if(isRegistered == true)
    {
        System.out.println("The non-academic course has already been registered.");

    }
    else
    {
        super.setCourseLeader(CourseLeader);
        this.setInstructorName(InstructorName);
        this.StartingDate = StartingDate;
        this.CompletionDate = CompletionDate;
        this.ExamDate = ExamDate;
        isRegistered = true;
        isRemoved = false;
    }
}

public void Remove()
{
    if(isRemoved == true)
    {
        System.out.println("The non-academic course has already been 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("The instructor name is " + InstructorName + ".");
            System.out.println("The starting date is " + StartingDate + ".");
            System.out.println("The completion date is " + CompletionDate + ".");
            System.out.println("The exam date is " + ExamDate + ".");
            System.out.println("The prerequisite is " + Prerequisite + ".");
        }
    }
}
```

INGCollege

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
public class INGCollege implements ActionListener
{
    private JFrame f;
    private JPanel p1,p2;
    private JLabel Title1, Title2, JCourseID, JCourseName, JDuration, JLevel, JCredit,
JNumberOfAssessments, JCourseID2, JLecturerName, JCourseLeader,JStartingDate,
JCompletionDate, JCourseID1, JCourseName1, JDuration1, JPrerequisite, JCourseID3,
JInstructorName, JCourseLeader1, JStartingDate1, JCompletionDate1, JExamDate,
JCourseID4;
    private JTextField JCourseID_, JCourseName_, JDuration_, JCourseID2_,
JLecturerName_, JCourseLeader_, JLevel_, JCredit_, JStartingDate_, JCompletionDate_,
JNumberOfAssessments_, JCourseID1_, JCourseName1_, JDuration1_, JPrerequisite_,
JCourseID3_, JInstructorName_, JCourseLeader1_, JStartingDate1_,
JCompletionDate1_, JExamDate_, JCourseID4_;
    private JButton Add, Register, Display, Clear, NextPage, Add1, Register1, Display1,
Clear1, Remove,PreviousPage;

    private ArrayList<Course>AL = new ArrayList<Course>();
    private AcademicCourse AC;
    private NonAcademicCourse NAC;

    public INGCollege()
    {
        f = new JFrame("Course Registration");
        p1 = new JPanel();
        p2 = new JPanel();

        Title1 = new JLabel("Academic Course");
        JCourseID = new JLabel("Course ID : ");
```

```
JCourseName = new JLabel("Course Name : ");
JDuration = new JLabel("Duration : ");
JLevel = new JLabel("Level : ");
JCredit = new JLabel("Credit : ");
JNumberOfAssessments = new JLabel("Number of Assessments :");

Title2 = new JLabel("Non-Academic Course");
JCourseID1 = new JLabel("Course ID : ");
JCourseName1 = new JLabel("Course Name : ");
JDuration1 = new JLabel("Duration : ");
JPrerequisite = new JLabel("Prerequisite : ");

JCourseID2 = new JLabel("Course ID : ");
JLecturerName = new JLabel("Lecturer Name : ");
JCourseLeader = new JLabel("Course Leader : ");
JStartingDate = new JLabel("Starting Date : ");
JCompletionDate = new JLabel("Completion Date : ");

JCourseID3 = new JLabel("Course ID : ");
JInstructorName = new JLabel("Instructor Name : ");
JCourseLeader1 = new JLabel("Course Leader : ");
JStartingDate1 = new JLabel("Starting Date : ");
JCompletionDate1 = new JLabel("Completion Date : ");
JExamDate = new JLabel("Exam Date : ");

JCourseID4 = new JLabel("Course ID : ");

JCourseID_ = new JTextField();
JCourseName_ = new JTextField();
JDuration_ = new JTextField();
JLevel_ = new JTextField();
JCredit_ = new JTextField();
JNumberOfAssessments_ = new JTextField();

JCourseID1_ = new JTextField();
JCourseName1_ = new JTextField();
JDuration1_ = new JTextField();
JPrerequisite_ = new JTextField();

JCourseID2_ = new JTextField();
JLecturerName_ = new JTextField();
JCourseLeader_ = new JTextField();
JStartingDate_ = new JTextField();
JCompletionDate_ = new JTextField();

JCourseID3_ = new JTextField();
```

```
JInstructorName_ = new JTextField();
JCourseLeader1_ = new JTextField();
JStartingDate1_ = new JTextField();
JCompletionDate1_ = new JTextField();
JExamDate_ = new JTextField();

JCourseID4_ = new JTextField();

f.setVisible(true);
p1.setVisible(true);
p2.setVisible(false);

f.setLayout(null);
p1.setLayout(null);
p2.setLayout(null);

f.setResizable(false);
f.setSize(860,640);
p1.setSize(860,640);
p2.setSize(860,640);

Add = new JButton("Add");
Register = new JButton("Register");
Display = new JButton("Display");
Clear = new JButton("Clear");
NextPage = new JButton("Next Page");

Add1 = new JButton("Add");
Register1 = new JButton("Register");
Display1 = new JButton("Display");
Clear1 = new JButton("Clear");
Remove = new JButton("Remove");
PreviousPage = new JButton("Previous Page");

Title1.setFont(new Font("Arial",Font.BOLD,25));
Title2.setFont(new Font("Arial",Font.BOLD,25));

Title1.setBounds(300,20,400,50);
JCourseID.setBounds(40,100,120,25);
JCourseName.setBounds(430,100,120,25);
JDuration.setBounds(40,150,120,25);
JLevel.setBounds(430,150,120,25);
JCredit.setBounds(40,200,120,25);
JNumberOfAssessments.setBounds(430,200,150,25);

Title2.setBounds(300,20,400,50);
```

```
JCourseID1.setBounds(40,100,120,25);
JCourseName1.setBounds(430,100,120,25);
JDuration1.setBounds(40,150,120,25);
JPrerequisite.setBounds(430,150,120,25);

JCourseID2.setBounds(40,310,120,25);
JLecturerName.setBounds(430,310,120,25);
JCourseLeader.setBounds(40,360,120,25);
JStartingDate.setBounds(430,360,120,25);
JCompletionDate.setBounds(40,410,120,25);

JCourseID3.setBounds(40,260,120,25);
JInstructorName.setBounds(430,260,120,25);
JCourseLeader1.setBounds(40,310,120,25);
JStartingDate1.setBounds(430,310,120,25);
JCompletionDate1.setBounds(40,360,120,25);
JExamDate.setBounds(430,360,120,25);

JCourseID4.setBounds(40,460,120,25);

JCourseID_.setBounds(160,100,180,25);
JCourseName_.setBounds(585,100,180,25);
JDuration_.setBounds(160,150,180,25);
JLevel_.setBounds(585,150,180,25);
JCredit_.setBounds(160,200,180,25);
JNumberOfAssessments_.setBounds(585,200,180,25);

JCourseID1_.setBounds(160,100,180,25);
JCourseName1_.setBounds(585,100,180,25);
JDuration1_.setBounds(160,150,180,25);
JPrerequisite_.setBounds(585,150,180,25);

JCourseID2_.setBounds(160,310,180,25);
JLecturerName_.setBounds(585,310,180,25);
JCourseLeader_.setBounds(160,360,180,25);
JStartingDate_.setBounds(585,360,180,25);
JCompletionDate_.setBounds(160,410,180,25);

JCourseID3_.setBounds(160,260,180,25);
JInstructorName_.setBounds(585,260,180,25);
JCourseLeader1_.setBounds(160,310,180,25);
JStartingDate1_.setBounds(585,310,180,25);
JCompletionDate1_.setBounds(160,360,180,25);
JExamDate_.setBounds(585,360,180,25);

JCourseID4_.setBounds(160,460,180,25);
```

```
Add.setBounds(585,250,120,25);
Register.setBounds(585,460,120,25);
Display.setBounds(220,510,120,25);
Clear.setBounds(380,510,120,25);
NextPage.setBounds(540,510,120,25);

Add1.setBounds(585,200,120,25);
Register1.setBounds(585,410,120,25);
Remove.setBounds(160,510,120,25);
Display1.setBounds(220,560,120,25);
Clear1.setBounds(380,560,120,25);
PreviousPage.setBounds(540,560,120,25);

p1.add(Title1);
p1.add(JCourseID);
p1.add(JCourseName);
p1.add(JDuration);
p1.add(JLevel);
p1.add(JCredit);
p1.add(JNumberOfAssessments);
p1.add(JCourseID_);
p1.add(JCourseName_);
p1.add(JDuration_);
p1.add(JLevel_);
p1.add(JCredit_);
p1.add(JNumberOfAssessments_);
p1.add(Add);
p1.add(JCourseID2);
p1.add(JLecturerName);
p1.add(JCourseLeader);
p1.add(JStartingDate);
p1.add(JCompletionDate);
p1.add(JCourseID2_);
p1.add(JLecturerName_);
p1.add(JCourseLeader_);
p1.add(JStartingDate_);
p1.add(JCompletionDate_);
p1.add(Register);
p1.add(Display);
p1.add(Clear);
p1.add(NextPage);

p2.add(Title2);
p2.add(JCourseID1);
p2.add(JCourseName1);
```

```
        p2.add(JDuration1);
        p2.add(JPrerequisite);
        p2.add(JCourseID1_);
        p2.add(JCourseName1_);
        p2.add(JDuration1_);
        p2.add(JPrerequisite_);
        p2.add(Add1);
        p2.add(JCourseID3);
        p2.add(JInstructorName);
        p2.add(JCourseLeader1);
        p2.add(JStartingDate1);
        p2.add(JCompletionDate1);
        p2.add(JExamDate);
        p2.add(JCourseID3_);
        p2.add(JInstructorName_);
        p2.add(JCourseLeader1_);
        p2.add(JStartingDate1_);
        p2.add(JCompletionDate1_);
        p2.add(JExamDate_);
        p2.add(Register1);
        p2.add(JCourseID4);
        p2.add(JCourseID4_);
        p2.add(Remove);
        p2.add(Display1);
        p2.add(Clear1);
        p2.add(PreviousPage);

        f.add(p1);
        f.add(p2);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        Add.addActionListener(this);
        Add1.addActionListener(this);
        Register.addActionListener(this);
        Register1.addActionListener(this);
        Display.addActionListener(this);
        Display1.addActionListener(this);
        Remove.addActionListener(this);
        Clear.addActionListener(this);
        Clear1.addActionListener(this);
        NextPage.addActionListener(this);
        PreviousPage.addActionListener(this);
    }

    public void actionPerformed(ActionEvent A)
    {
```

```
        if(A.getSource()==Add)
        {
            String CourseID="";
            String CourseName="";
            int Duration = 0;
            String Level = "";
            int Credit = 0;
            int NumberOfAssessments = 0;
            try
            {
                CourseID = JCourseID_.getText();
                CourseName = JCourseName_.getText();
                Duration = Integer.parseInt(JDuration_.getText());
                Level = JLevel_.getText();
                Credit = Integer.parseInt(JCredit_.getText());
                NumberOfAssessments = Integer.parseInt(JNumberOfAssessments_.getText());
                boolean SameAC = false;
                for(Course C:AL)
                {
                    if (C.getCourseID().equals(CourseID))
                    {
                        SameAC = true;
                    }
                }
                if(SameAC==false)
                {
                    AC = new
AcademicCourse(CourseID,CourseName,Duration,Level,Credit,NumberOfAssessments);
                    AL.add(AC);
                    JOptionPane.showMessageDialog(f,"The Academic Course has been
added.");
                }
                else
                {
                    JOptionPane.showMessageDialog(f,"The Academic Course has already been
added.");
                }
            }
            catch(NumberFormatException e)
            {
                JOptionPane.showMessageDialog(p1,"Please fill up the form properly
!","ERROR",JOptionPane.ERROR_MESSAGE);
            }
        }
        else if (A.getSource()==Add1)
        {
```

```
            String CourseID = "";
            String CourseName = "";
            int Duration=0;
            String Prerequisite = "";
            try
            {
               CourseID = JCourseID1_.getText();
               CourseName = JCourseName1_.getText();
               Duration = Integer.parseInt(JDuration1_.getText());
               Prerequisite = JPrerequisite_.getText();
               boolean SameNAC = false;
               for(Course C:AL)
               {
                  if(C.getCourseID().equals(CourseID))
                  {
                     SameNAC = true;
                  }
               }
               if(SameNAC == false)
               {
                  NAC = new NonAcademicCourse(CourseID, CourseName, Duration,
Prerequisite);
                  AL.add(NAC);
                  JOptionPane.showMessageDialog(f,"The Non-Academic Course is added.");
               }
               else
               {
                  JOptionPane.showMessageDialog(f,"The Non-Academic Course has already
been added");
               }
            }
            catch(NumberFormatException e)
            {
               JOptionPane.showMessageDialog(f,"Please fill up the form properly
!","ERROR",JOptionPane.ERROR_MESSAGE);
            }
         }

         else if (A.getSource()==Register)
         {
            String CourseID = "";
            String CourseLeader = "";
            String LecturerName = "";
            String StartingDate = "";
            String CompletionDate = "";
            try
```

```
        {
            CourseID = JCourseID2_.getText();
            CourseLeader = JCourseLeader_.getText();
            LecturerName = JLecturerName_.getText();
            StartingDate = JStartingDate_.getText();
            CompletionDate = JCompletionDate_.getText();
            boolean SameAC1 = false;
            for(Course CO:AL)
            {
                if(AC.getCourseID().equals(CourseID))
                {
                    SameAC1 = true;
                    if (CO instanceof AcademicCourse)
                    {
                        AC = (AcademicCourse) CO;
                        if (AC.getisRegistered()==true)
                        {
                            JOptionPane.showMessageDialog(f,"The academic Course has
already been registered.");
                        }
                        else
                        {
                            AC.Register(LecturerName, CourseLeader, StartingDate,
CompletionDate);
                            JOptionPane.showMessageDialog(f,"The Academic Course has been
registered.");
                        }
                    }
                }
                else
                {
                    JOptionPane.showMessageDialog(f,"The CourseID do not
match.","Alert",JOptionPane.WARNING_MESSAGE);
                    break;
                }
            }
        }

        catch (NumberFormatException e)
        {
            JOptionPane.showMessageDialog(p1,"Please  fill up the forms properly
!","ERROR",JOptionPane.ERROR_MESSAGE);
        }
    }

    else if (A.getSource()==Register1)
```

```
    {
        String CourseID = "";
        String CourseLeader = "";
        String InstructorName = "";
        String StartingDate = "";
        String CompletionDate = "";
        String ExamDate = "";
        try
        {
            CourseID = JCourseID3_.getText();
            CourseLeader = JCourseLeader1_.getText();
            InstructorName = JLecturerName_.getText();
            StartingDate = JStartingDate1_.getText();
            CompletionDate = JCompletionDate1_.getText();
            ExamDate = JExamDate_.getText();
            boolean SameNAC1 = false;
            for(Course CO:AL)
            {
                if(NAC.getCourseID().equals(CourseID))
                {
                    SameNAC1 = true;
                    if (CO instanceof NonAcademicCourse)
                    {
                        NAC = (NonAcademicCourse)CO;
                        if (NAC.getisRegistered()==true)
                        {
                            JOptionPane.showMessageDialog(f,"The Non-Academic Course has
already been registered.");
                        }
                        else
                        {
                            NAC.Register(CourseLeader, InstructorName, StartingDate,
CompletionDate, ExamDate);
                            JOptionPane.showMessageDialog(f,"The Non-Academic Course has
been registered.");
                        }
                    }
                }
                else
                {
                    JOptionPane.showMessageDialog(f,"The CourseID do not
match.","Alert",JOptionPane.WARNING_MESSAGE);
                    break;
                }
            }
        }
```

```
            catch(NumberFormatException e)
            {
                JOptionPane.showMessageDialog(f,"Please fill up the form properly
!","ERROR",JOptionPane.ERROR_MESSAGE);
            }
        }
        else if (A.getSource()==Remove)
        {
            String CourseID = JCourseID4_.getText();
            try{
                for(Course C:AL){
                    if(NAC.getCourseID().equals(CourseID))
                    {
                        if(C instanceof NonAcademicCourse)
                        {
                            NAC=(NonAcademicCourse)C;
                            if(NAC.getisRemoved()==false)
                            {
                                NAC.Remove();
                                JOptionPane.showMessageDialog(f,"The Course has been
removed.");
                            }
                            else if(NAC.getisRemoved()==true)
                            {
                                JOptionPane.showMessageDialog(f,"The Course has already been
removed.");
                            }
                        }
                    }
                    else
                    {
                        JOptionPane.showMessageDialog(f,"Enter valid
CourseID","Alert",JOptionPane.WARNING_MESSAGE);
                        break;
                    }
                }
            }
            catch (Exception e)
            {
                JOptionPane.showMessageDialog(f,"Please fill up the form properly
!","ERROR",JOptionPane.ERROR_MESSAGE);
            }
        }
        else if (A.getSource()==Display)
        {
            for(Course CO:AL)
```

```
        {
            if(CO instanceof AcademicCourse)
            {
                AcademicCourse AC = (AcademicCourse)CO;
                AC.Display();
            }
        }
    }
    else if (A.getSource()==Display1)
    {
        for(Course CO:AL)
        {
            if(CO instanceof NonAcademicCourse)
            {
                NonAcademicCourse NAC = (NonAcademicCourse)CO;
                NAC.Display();
            }
        }
    }

    else if (A.getSource()==Clear)
    {
        JCourseID_.setText("");
        JCourseName_.setText("");
        JDuration_.setText("");
        JLevel_.setText("");
        JCredit_.setText("");
        JNumberOfAssessments_.setText("");
        JCourseID2_.setText("");
        JLecturerName_.setText("");
        JCourseLeader_.setText("");
        JStartingDate_.setText("");
        JCompletionDate_.setText("");
    }
    else if (A.getSource()==Clear1)
    {
        JCourseID1_.setText("");
        JCourseName1_.setText("");
        JDuration1_.setText("");
        JPrerequisite_.setText("");
        JCourseID3_.setText("");
        JInstructorName_.setText("");
        JCourseLeader1_.setText("");
        JStartingDate1_.setText("");
        JCompletionDate1_.setText("");
        JExamDate_.setText("");
```

```
            JCourseID4_.setText("");
        }
        else if(A.getSource()==NextPage)
        {
            p1.setVisible(false);
            p2.setVisible(true);
        }
        else if(A.getSource()==PreviousPage)
        {
            p2.setVisible(false);
            p1.setVisible(true);
        }
    }

    public static void main(String[]args)
    {
        new INGCollege();
    }
}
```