LONDON METROPOLITAN UNIVERSITY

islington college

(इस्लिङ्टन कलेज)

Module Code & Module Title

CC4001NI Programming

COURSEWORK-2

Assessment Weightage & Type

30% Individual Coursework

Semester and Year

Spring 2021

Student Name: Aashna Shrestha

Group: C13

London Met ID: 20048800

College ID: NP01CP4S210103

Assignment Due Date: 20th August, 2021

Assignment Submission Date: 20th 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 submission 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

The project was assigned as a coursework for the module CS4001NI Programming. It includes a Graphical User Interface coded in Java. The Java program was written in BlueJ. BlueJ is an Integrated Development Environment created by the Kings College London. It allows the users to work with Java and supports interaction with objects. Once the program was compiled, the documentation was written as a report. The report, which contains the method description, class diagram, pseudocode and testing of different functions, has been compiled in MS Word.

The GUI allows users to input details of any academic or non-academic course. The user can then add the course to an ArrayList and register it. The GUI contains JLabels for the headings and sub headings of the form, JTextFields where users can input the required details, JComboBox to choose the date when a course starts, completes or the exam is held and JButtons with different functions such as add, register or display the courses and clear all the text fields.

The code consists a method to build the GUI, accessor methods which returns the user input, a method to execute certain tasks when particular buttons are clicked, and a main method where the method to build GUI will be called. The current project has been integrated with a previously done project. The methods to perform certain tasks like add or remove any course were already written in the previous code. The current project executes the methods according to the user input. It also restricts them from performing some actions to avoid duplications and exceptions.

2. Class Diagram

INGCollege
- frame : JFrame - mainPanel : JPanel - addCoursePanel : JPanel - registerPanel : JPanel - title : JLabel - academicTitle : JLabel - nonAcademicTitle : JLabel - courseOption : JLabel - courseIdLabel : JLabel - courseNameLabel : JLabel - durationLabel : JLabel - levelLabel : JLabel - creditLabel : JLabel - assessmentLabel : JLabel - prerequisiteLabel : JLabel - idRegisterLabel : JLabel - leaderLabel : JLabel - lecturerLabel : JLabel - startDateLabel : JLabel - completionDateLabel : JLabel - examDateLabel : JLabel - courseIdField : JTextField - courseName : JTextField - durationField : JTextField - levelField : JTextField

- creditField : JTextField
- assessmentField : JTextField
- prerequisiteField : JTextField
- idRegisterField : JTextField
- leaderField : JTextField
- lecturerField : JTextField
- instructorField : JTextField

- startYear : JComboBox
- startMonth : JComboBox
- startDay : JComboBox
- completionYear : JComboBox
- completionMonth : JComboBox
- completionDay : JComboBox
- examYear : JComboBox
- examMonth : JComboBox
- examDay : JComboBox

- academicButton : JButton
- nonAcademicButton : JButton
- addAcademicButton : JButton
- addNonAcademicButton : JButton
- registerAcademicButton : JButton
- registerNonAcademicButton : JButton
- removeNonAcademicButton : JButton
- displayAcademicButton : JButton
- displayNonAcademicButton : JButton
- clearButton : JButton

- start_year : String
- start_month : String

- start_day : String
- startDate : String
- completion_year : String
- completion_month : String
- completion_day : String
- completionDate : String
- exam_year : String
- exam_month : String
- exam_day : String
- examDate : String

ing : INGCollege

- courseList : Course
- academicObject : AcademicCourse
- nonAcademicObject : NonAcademicCourse

+ INGCollege()
+ courseGui() : void
+ getCourseId() : String
+ getCourseName() : String
+ getDuration() : String
+ getLevel() : String
+ getCredit() : String
+ getNoOfAssessments() : String
+ getPrerequisite() : String
+ getRegisterId() : String
+ getCourseLeader() : String
+ getLecturer() : String
+ getInstructor() : String
+getStartDate() : String
+ getCompletionDate() : String
+ getExamDate() : String
+ actionPerformed(e : ActionEvent) : void

Table 1 Class Diagram

3. Pseudocode

Class INGCollege

START

CREATE class IngCollege

DECLARE instance variable frame as JFrame,

mainPanel, addCoursePanel, registerPanel as JPanel,

title, courseOption, academicTitle, nonAcademicTitle, courseIdLabel, courseNameLabel, durationLabel, levelLabel, creditLabel, assessmentLabel, prerequisiteLabel, idRegisterLabel, leaderLabel, lecturerLabel, instructorLabel, startDateLabel, completionDateLabel, examDateLabel as JLabel,

courseIdField, courseNameField, durationField, levelField, creditField, assessmentField, prerequisiteField, idRegisterField, leaderField, lecturerField, instructorField as JTextField,
academicButton, nonAcademicButton, addAcademicButton, addNonAcademicButton, registerAcademicButton, registerNonAcademicButton, removeButton,displayAcademicButton, displayNonAcademicButton, clearButton as JButton,
startYear, startMonth, startDay, completionYear, completionMonth, completionDay, examYear, examMonth, examDay as JComboBox
start_year, start_month, start_day, startDate, completion_year, completion_month, completion_day, completionDate, exam_year, exam_month, exam_day, examDate as String
academicObject as AcademicCourse,
nonAcademicObject as NonAcademicCourse
DECLARE static variable ing as INGCollege
INITIALIZE an ArrayList of Course type to courseList
INITIALIZE ArrayList to courseList

CREATE courseGUI()
DO
    INITIALIZE JFrame to frame
    INITIALIZE JPanel to mainPanel
    INITIALIZE Color of rgb 27, 38, 66 to mainBackground
    INITIALIZE Color of rgb 191, 193, 199 to innerPanel
    INITIALIZE Color of rgb 247, 129, 2 to buttonColor
    SET the layout of mainPanel to null
    SET the background color of mainPanel to mainBackground
    INITIALIZE Font to the variable titleFont
    INITIALIZE Font to the variable mainFont
    INITIALIZE Font to the variable comboBoxFont

INITIALIZE JLabel to the reference variable title

SET the text of title to “Course Registration”

SET the bounds of x axis to 285 px, y axis to 20 px, width to 250 px and height to 50 px for title

SET the font of title to titleFont

SET the foreground color of title to white

ADD title to mainPanel

INITIALIZE JLabel to courseOption

SET the text of courseOption to “Choose your course!”

SET the font of courseOption to mainFont

SET the foreground color of courseOption to White

SET the bounds of x axis to 320 px, y axis to 90 px, width to 300 px and height to 30 px for courseOption

ADD courseOption to mainPanel

INITIALIZE JButton to academicButton

SET the text of academicButton to “Academic Course”

SET the bounds of x axis to 310 px, y axis to 130 px, width to 180 px and height to 30 px for academicButton

SET the font of academicButton to mainFont

SET the foreground color of academicButton to white

SET the background color of academicButton to buttonColor

CALL the method addActionListener of academicButton with the current object as parameter

ADD academicButton to mainPanel

INITIALIZE JButton to nonAcademicButton

SET the text of nonAcademicButton to “Non Academic Course”

SET the bounds of x axis to 300 px, y axis to 170 px, width to 200 px and height to 30 px for nonAcademicButton

SET the font of nonAcademicButton to mainFont

SET the foreground color of nonAcademicButton to white

SET the background color of nonAcademicButton to buttonColor

CALL the method addActionListener of nonAcademicButton with the current object as parameter

ADD nonAcademicButton to mainPanel

INITIALIZE JPanel to addCoursePanel

SET the background color of addCoursePanel to innerPanel

SET the bounds of x axis to 300 px, y axis to 170 px, width to 200 px and height to 30 px for addCoursePanel

SET the visibility of addCoursePanel to true

SET the layout of addCoursePanel to null

ADD addCoursePanel to mainPanel

INITIALIZE JPanel to registerPanel

SET the background color of registerPanel to innerPanel

SET the bounds of x axis to 410 px, y axis to 220 px, width to 355 px and height to 400px

SET the visibility of registerPanel to true

SET the layour of registerPanel to null

ADD registerPanel to mainPanel

INITIALIZE JLabel to the reference variable courseIdLabel

SET the text of courseIdLabel to “Course ID: ”

SET the bounds of courseIdLabel to 10px x-axis, 20px y-axis, 100px width and 30px height

SET the font of courseIdLabel to mainFont

ADD courseIdLabel to addCoursePanel

INITIALIZE JTextField to the reference variable courseIdField

SET the bounds of courseIdField to 170px x-axis, 20px y-axis, 160px width and 30px height

ADD courseIdField to addCoursePanel

INITIALIZE JLabel to the reference variable courseNameLabel

SET the text of courseNameLabel to “Course Name: ”

SET the bounds of courseNameLabel to 10px x-axis, 70px y-axis, 110px width and 30px height

SET the font of courseNameLabel to mainFont

ADD courseNameLabel to addCoursePanel

INITIALIZE JTextField to the reference variable courseNameField

SET the bounds of courseNameField to 170px x-axis, 70px y-axis, 160px width and 30px height

ADD courseNameField to addCoursePanel

INITIALIZE JLabel to the reference variable durationLabel

SET the text of durationLabel to “Duration: ”

SET the bounds of durationLabel to 10px x-axis, 120px y-axis, 100px width and 30px height

SET the font of durationLabel to mainFont

ADD durationLabel to addCoursePanel

INITIALIZE JTextField to the reference variable durationField

SET the bounds of durationField to 170px x-axis, 120px y-axis, 160px width and 30px height

ADD durationField to addCoursePanel

INITIALIZE JLabel to the reference variable levelLabel

SET the text of levelLabel to “Level: ”

SET the bounds of levelLabel to 10px x-axis, 170px y-axis, 100px width and 30px height

SET the font of levelLabel to mainFont

ADD levelLabel to addCoursePanel

INITIALIZE JTextField to the reference variable levelField

SET the bounds of levelField to 170px x-axis, 170px y-axis, 160px width and 30px height

ADD levelField to addCoursePanel

INITIALIZE JLabel to the reference variable creditLabel

SET the text of creditLabel to “Credit: ”

SET the bounds of creditLabel to 10px x-axis, 220px y-axis, 100px width and 30px height

SET the font of creditLabel to mainFont

ADD creditLabel to addCoursePanel

INITIALIZE JTextField to the reference variable creditField

SET the bounds of creditField to 170px x-axis, 220px y-axis, 160px width and 30px height

ADD creditField to addCoursePanel

INITIALIZE JLabel to the reference variable assessmentLabel

SET the text of assessmentLabel to “No. of Assessments: ”

SET the bounds of assessmentLabel to 10px x-axis, 270px y-axis, 160px width and 30px height

SET the font of assessmentLabel to mainFont

ADD assessmentLabel to addCoursePanel

INITIALIZE JTextField to the reference variable assessmentField

SET the bounds of assessmentField to 170px x-axis, 270px y-axis, 160px width and 30px height

ADD assessmentField to addCoursePanel

INITIALIZE JLabel to the reference variable prerequisiteLabel
SET the text of courseIdLabel to “Prerequisite: ”
SET the bounds of prerequisiteLabel to 10px x-axis, 170px y-axis, 100px width and 30px height
SET the font of prerequisiteLabel to mainFont
ADD prerequisiteLabel to addCoursePanel
SET the visibility of prerequisiteLabel to false

INITIALIZE JTextField to the reference variable prerequisiteField
SET the bounds of prerequisiteField to 170px x-axis, 170px y-axis, 160px width and 30px height
ADD prerequisiteField to addCoursePanel
SET the visibility of prerequisiteField to false

INITIALIZE JButton to the reference variable addAcademicButton
SET the text of addAcademicButton to “Add”
SET the bounds of addAcademicButton to 140px x-axis, 340px y-axis, 80px width and 30px height
SET the font of addAcademicButton to mainFont
CALL the method addActionListener of addAcademicButton with the current object as parameter
ADD addAcademicButton to addCoursePanel

INITIALIZE JButton to the reference variable addNonAcademicButton
SET the text of addNonAcademicButton to “Add”
SET the bounds of addNonAcademicButton to 140px x-axis, 280px y-axis, 80px width and 30px height
SET the font of addNonAcademicButton to mainFont
CALL the method addActionListener of addNonAcademicButton with the current object as parameter

ADD addNonAcademicButton to addCoursePanel

SET the visibility of addNonAcademicButton to false

INITIALIZE JButton to the reference variable removeButton

SET the text of removeButton to “Remove”

SET the bounds of removeButton to 130px x-axis, 320px y-axis, 100px width and 30px height

SET the font of removeButton to mainFont

CALL the method addActionListener of removeButton with the current object as parameter

ADD removeButton to addCoursePanel

SET the visibility of removeButton to false

INITIALIZE JLabel to the reference variable idRegisterLabel

SET the text of idRegisterLabel to “Course ID: ”

SET the bounds of idRegisterLabel to 10px x-axis, 20px y-axis, 100px width and 30px height

SET the font of idRegisterLabel to mainFont

ADD idRegisterLabel to registerPanel

INITIALIZE JTextField to the reference variable idRegisterField

SET the bounds of idRegisterField to 170px x-axis, 20px y-axis, 160px width and 30px height

ADD idRegisterField to registerPanel

INITIALIZE JLabel to the reference variable leaderLabel

SET the text of leaderLabel to “CourseLeader: ”

SET the bounds of leaderLabel to 10px x-axis, 70px y-axis, 120px width and 30px height

SET the font of leaderLabel to mainFont

ADD leaderLabel to registerPanel

INITIALIZE JTextField to the reference variable leaderField
SET the bounds of leaderField to 170px x-axis, 70px y-axis, 160px width and 30px height
ADD leaderField to registerPanel

INITIALIZE JLabel to the reference variable lecturerLabel
SET the text of lecturerLabel to “Lecturer: ”
SET the bounds of lecturerLabel to 10px x-axis, 120px y-axis, 100px width and 30px height
SET the font of lecturerLabel to mainFont
ADD lecturerLabel to registerPanel

INITIALIZE JTextField to the reference variable lecturerField
SET the bounds of lecturerField to 170px x-axis, 120px y-axis, 160px width and 30px height
ADD lecturerField to registerPanel

INITIALIZE JLabel to the reference variable instructorLabel
SET the text of instructorLabel to “Instructor: ”
SET the bounds of instructorLabel to 10px x-axis, 120px y-axis, 100px width and 30px height
SET the font of instructorLabel to mainFont
ADD instructorLabel to registerPanel

INITIALIZE JTextField to the reference variable instructorField
SET the bounds of instructorField to 170px x-axis, 120px y-axis, 160px width and 30px height
ADD instructorField to registerPanel
SET the visibility of instructorField to false

INITIALIZE JLabel to the reference variable startDateLabel

SET the text of startDateLabel to “Start Date: ”

SET the bounds of startDateLabel to 10px x-axis, 170px y-axis, 150px width and 30px height

SET the font of startDateLabel to mainFont

ADD startDateLabel to registerPanel

INITIALIZE an array of length 28 to the reference variable yearList

INITIALIZE the value of a variable year to 2020

INITIALIZE the value of a variable i to 0

FOR i < 26

    INITIALIZE the array year with index i to year

    INCREMENT the value of year

    INCREMENT the value of i

END FOR

INITIALIZE JComboBox to the reference variable startYear

ADD the elements of yearList to startYear

SET the bounds of 160px x-axis, 170px y-axis, 60px width and 30px height to startYear

SET the font comboBoxFont to startYear

ADD startYear to registerPanel

INITIALIZE an array of months to the reference variable month

ADD the elements “January”, "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December" to month

INITIALIZE JComboBox to the reference variable startMonth

ADD the elements of month to startMonth

SET the bounds of 225px x-axis, 170px y-axis, 75px width and 30px height to starMonth

SET the font comboBoxFont to startMonth

ADD startMonth to registerPanel

INITIALIZE an array of length 31 to the reference variable dayList

INITIALIZE the value 1 to an integer variable day

INITIALIZE the value 0 to an integer variable 1

FOR i<=30

    IF day < 10

        INITIALIZE “0” + day to the index i of dayList

    ELSE

        INITILIAZE the String value of day to index I of dayList

    END IF

    INCREMENT the value of day

    INCREMENT the value of i

END FOR

INITIALIZE JLabel to the reference variable completionDateLabel

SET the text of completionDateLabel to “Completion Date: ”

SET the bounds of completionDateLabel to 10px x-axis, 220px y-axis, 140px width and 30px height

SET the font of completionDateLabel to mainFont

ADD completionDateLabel to registerPanel

INITIALIZE JComboBox to the reference variable completionYear

ADD the elements of yearList to completionYear

SET the bounds of 160px x-axis, 220px y-axis, 60px width and 30px height to completionYear

SET the font comboBoxFont to completionYear

ADD completionYear to registerPanel

INITIALIZE JComboBox to the reference variable completionMonth

ADD the elements of month to completionMonth

SET the bounds of 225px x-axis, 220px y-axis, 75px width and 30px height to completionMonth

SET the font comboBoxFont to completionMonth

ADD completionMonth to registerPanel

INITIALIZE JComboBox to the reference variable completionDay

ADD the elements of dayList to completionDay

SET the bounds of 305px x-axis, 220px y-axis, 40px width and 30px height to completionDay

SET the font comboBoxFont to completionDay

ADD completionDay to registerPanel

INITIALIZE JLabel to the reference variable examDateLabel

SET the text of examDateLabel to “Exam Date: ”

SET the bounds of examDateLabel to 10px x-axis, 270px y-axis, 160px width and 30px height

SET the font of examDateLabel to mainFont

ADD examDateLabel to registerPanel

INITIALIZE JComboBox to the reference variable examYear

ADD the elements of yearList to examYear

SET the bounds of 160px x-axis, 270px y-axis, 60px width and 30px height to examYear

SET the font comboBoxFont to examYear

ADD examYear to registerPanel

INITIALIZE JComboBox to the reference variable examMonth

ADD the elements of month to examMonth

SET the bounds of 225px x-axis, 270px y-axis, 75px width and 30px height to examMonth

SET the font comboBoxFont to examMonth

ADD examMonth to registerPanel

INITIALIZE JComboBox to the reference variable examDay

ADD the elements of dayList to examDay

SET the bounds of 305px x-axis, 270px y-axis, 40px width and 30px height to examDay

SET the font comboBoxFont to examDay

ADD examDay to registerPanel

INITIALIZE JButton to the reference variable registerAcademicButton

SET the text of registerAcademicButton to “Register”

SET the bounds of registerAcademicButton to 140px x-axis, 320px y-axis, 100px width and 30px height

SET the font of registerAcademicButton to mainFont

CALL the method addActionListener of registerAcademicButton with the current object as parameter

ADD registerAcademicButton to registerPanel

INITIALIZE JButton to the reference variable registerNonAcademicButton

SET the text of registerNonAcademicButton to “Register”

SET the bounds of registerNonAcademicButton to 140px x-axis, 340px y-axis, 100px width and 30px height

SET the font of registerNonAcademicButton to mainFont

CALL the method addActionListener of registerNonAcademicButton with the current object as parameter

ADD registerNonAcademicButton to registerPanel

SET the visibility of registerNonAcademicButton to false

INITIALIZE JButton to the reference variable displayAcademicButton

SET the text of displayAcademicButton to “Display”

SET the bounds of displayAcademicButton to 285px x-axis, 640px y-axis, 100px width and 30px height

SET the font of displayAcademicButton to mainFont

SET the foreground color of displayAcademicButton to WHITE

SET the background color of displayAcademicButton to buttonColor

CALL the method addActionListener of displayAcademicButton with the current object as parameter

ADD displayAcademicButton to registerPanel

INITIALIZE JButton to the reference variable displayNonAcademicButton

SET the text of displayNonAcademicButton to “Display”

SET the bounds of displayNonAcademicButton to 285px x-axis, 640px y-axis, 100px width and 30px height

SET the font of displayNonAcademicButton to mainFont

SET the foreground color of displayNonAcademicButton to WHITE

SET the background color of displayNonAcademicButton to buttonColor

CALL the method addActionListener of displayNonAcademicButton with the current object as parameter

ADD displayNonAcademicButton to mainPanel

SET the visibility of displayNonAcademicButton to false

INITIALIZE JButton to the reference variable clearButton

SET the text of clearButton to “Clear”

SET the bounds of clearButton to 400px x-axis, 640px y-axis, 100px width and 30px height

SET the font of clearButton to mainFont

SET the foreground color of clearButton to WHITE

SET the background color of clearButton to buttonColor

CALL the method addActionListener of clearButton with the current object as parameter
ADD clearButton to registerPanel

ADD mainPanel to frame
SET the bounds of frame to 300px x-axis, 0px y-axis, 800px width and 730px height
SET the visibility of frame to true
SET resizable of frame to false
END DO

CREATE getCourseId()
DO
    RETURN the text from this.courseIdField
END DO

CREATE getCoursenName()
DO
    RETURN the text from this.courseNameField
END DO

CREATE getDuration()
DO
    CONVERT the text from this.durationField into integer
    RETURN the integer value of this.durationField
END DO

CREATE getlevel()
DO
    RETURN the text from this.levelField
END DO

```
CREATE getCredit()
DO
        RETURN the text from this.creditField
END DO

CREATE getNoOfAssessments()
DO
        CONVERT the text from this.assessmentField into integer
        RETURN the integer value of this.assessmentField
END DO

CREATE getPrerequisite()
DO
        RETURN the text from this.prerequisite Field
END DO

CREATE getRegisterId()
DO
        RETURN the text from this.idRegister Field
END DO

CREATE getCourseLeader()
DO
        RETURN the text from this.leaderField
END DO
CREATE getLecturer()
DO
        RETURN the text from this.lecturerField
END DO
```

CREATE getInstructor()

DO

RETURN the text from this.instructorField

END DO

CREATE getStartDate()

DO

INITIALIZE the text selected from this.startYear to start_year

INITIALIZE the text selected from this.startMonth to start_month

INITIALIZE the text selected from this.startDay to start_day

INITIALIZE the concatenated strings start_year, start_month and start_day to this.startDate

RETURN startDate

END DO

CREATE getCompletionDate()

DO

INITIALIZE the text selected from this.completionYear to completion_year

INITIALIZE the text selected from this. completionMonth to completion_month

INITIALIZE the text selected from this. completionDay to completion _day

INITIALIZE the concatenated strings completion_year, completion_month and completion_day to this. completionDate

RETURN completionDate

END DO

CREATE getExamDate()

DO

INITIALIZE the text selected from this.examYear to exam_year

INITIALIZE the text selected from this. examMonth to exam_month

INITIALIZE the text selected from this. examDay to exam_day

INITIALIZE the concatenated strings exam_year, exam_month and exam_day to this.examDate

RETURN examDate

END DO

CREATE actionPerformed(ActionEvent e)

DO

IF e.getSource() == nonAcademicButton

SET the visibility of levelLabel to false

SET the visibility of creditLabel to false

SET the visibility of assessmentLabel to false

SET the visibility of levelField to false

SET the visibility of creditField to false

SET the visibilitiy of lecturerLabel to false

SET the visibility of lecturerField to false

SET the visibility of assessmentField to false

SET the visibility of prerequisiteLabel to true

SET the visibility of prerequisiteField to true

SET the visibility of addAcademicButton to false

SET the visibility of addNonAcademicButton to true

SET the visibility of removeButton to true

SET the visibility of examDateLabel to true

SET the visibility of examYear to true

SET the visibility of examMonth to true

SET the visibility of examDay to true

SET the visibility of registerNonAcademicButton to true

SET the visibility of registerAcademicButton to true

SET the visibility of instructorLabel to true

SET the visibility of instructorField to true

SET the visibility of nonAcademicTitle to true

SET the visibility of academicTitle to true

```
        SET the visibility of displayAcademicButton to false
        SET the visibility of displayNonAcademicButton to true
    END IF

    ELSE IF e.getSource() == academicButton
        SET the visibility of levelLabel to true
        SET the visibility of creditLabel to true
        SET the visibility of assessmentLabel to true
        SET the visibility of levelField to true
        SET the visibility of creditField to true
        SET the visibilitiy of lecturerLabel to true
        SET the visibility of lecturerField to true
        SET the visibility of assessmentField to true
        SET the visibility of prerequisiteLabel to false
        SET the visibility of prerequisiteField to false
        SET the visibility of addAcademicButton to true
        SET the visibility of addNonAcademicButton to false
        SET the visibility of removeButton to false
        SET the visibility of examDateLabel to false
        SET the visibility of examYear to false
        SET the visibility of examMonth to false
        SET the visibility of examDay to false
        SET the visibility of registerNonAcademicButton to false
        SET the visibility of registerAcademicButton to false
        SET the visibility of instructorLabel to false
        SET the visibility of instructorField to false
        SET the visibility of nonAcademicTitle to false
        SET the visibility of academicTitle to false
        SET the visibility of displayAcademicButton to true
        SET the visibility of displayNonAcademicButton to true
    END IF
```

```
ELSE IF e.getSource() == addAcademicButton
    INITIALIZE courseFound to false
    FOR each element course in courseList
        IF course.getcourseID() is equal to getCourseId() and
        course is an instance of AcademicCourse
        DISPLAY dialog box with the message "The course has
        been added already."
        INIITIALIZE courseFound to true
        END IF
    END FOR
    IF not courseFound or courseList is empty
    TRY
    DO
    IF getCourseId() or getCourseName() or getLevel() or getCredit() is
    empty
        DISPLAY dialog box with the message "Please fill all the
        fields."
        ELSE
            INITIALIZE the object of AcademicCourse to
            academicObject with the parameters getCourseId(),
            getCourseName(), getDuration(), getLevel(),
            getCredit() and getNoOfAssessments()
            ADD academicObject to courseList
            DISPLAY a dialog box with the message "Academic
            Course has been added successfully."
        END IF
    END DO
    CATCH (NumberFormatException ae)
    DO
```

```
                DISPLAY a dialog box with the message “Enter a numerical
                value in duration and number of assessments”
            END DO
            CATCH (NullPointerException ex)
            DO
                DISPLAY a dialog box with the message “Please fill all the
                fields”
            END DO
        END IF
    END IF

    ELSE IF e.getSource() == registerAcademicButton
        INITILIAZE courseFound to false
        IF getRegister() or getCourseLeader or getLecturer is empty
            DISPLAY a dialog box with the message “Please fill all the fields”
        ELSE
            FOR each element course in courseList
                IF course.getcourseID() equals getRegisterId() and course is
                an instance of AcademicCourse
                    INITIALIZE course to academic_obj by downcasting
                    into AcademicCourse
                    INITIALIZE courseFound to true
                    IF academic_obj.getisRegistered() is true
                        DISPLAY a dialog box with the message "The
                        course has already been registered"
                    ELSE
                        CALL the register method of academic_obj
                        with the parameters(getCourseLeader(),
                        getLecturer(), getStartDate(),
                        getCompletionDate)
```

```
                    DISPLAY a dialog box with the message “The
                    course has been registered successfully”
                END IF
            END IF
        END FOR
        IF courseFound is false or courseList is empty
            DISPLAY a dialog box with the message “The course has
            not been added yet”
        END IF
    END IF
END IF

ELSE IF e.getSource() == displayAcademicButton
    INITIALIZE courseFound to false
    FOR each element course of courseList
        IF course is an instance of AcademicCourse
            INITILIAZE course to academic_obj by downcasting into
            AcademicCourse
            CALL the display method of academic_obj
            INTILIAZE courseFound to true
        END IF
    END FOR
    IF courseFound is false and courseList is empty
        DISPLAY a dialog box with the message "Academic Course has
        not been added yet."
    END IF
END IF

ELSE IF e.getSource() == addNonAcademicButton
    INITIALIZE courseFound to false
    FOR each element course in courseList
```

```
        IF course.getcourseID() is  equal to getCourseId() and course is an
        instance of AcademicCourse
            DISPLAY dialog box with the message "The course has
            been added already."
            INIITIALIZE courseFound to true
        END IF
    END FOR
    IF not courseFound or courseList is empty
        TRY
        DO
            IF getCourseId() or getCourseName() or getprerequisite() is
            empty
                DISPLAY dialog box with the message "Please fill all
                the fields."
            ELSE
                INITIALIZE the object of NonAcademicCourse to
                nonAcademicObject with the parameters
                getCourseId(), getCourseName(), getDuration() and
                getprerequisite()
                ADD nonAcademicObject to courseList
                DISPLAY a dialog box with the message " Non
                Academic Course has been added successfully."
            END IF
        END DO
        CATCH (NumberFormatException ae)
        DO
        DISPLAY a dialog box with the message "Enter a numerical value
        in duration"
        END DO
        CATCH (NullPointerException ex)
        DO
```

```
                    DISPLAY a dialog box with the message “Please fill all the
                    fields”
                END DO
            END IF
        END IF

        ELSE IF e.getSource() == registerNonAcademicButton
            INITILIAZE courseFound to false
            IF getRegister() or getCourseLeader() or getLecturer() is empty
                DISPLAY a dialog box with the message “Please fill all the fields”
            ELSE
                FOR each element course in courseList
                    IF course.getcourseID() equals getRegisterId() and course is
                    an instance of NonAcademicCourse
                        INITIALIZE course to nonAcademic_obj by
                        downcasting into NonAcademicCourse
                        INITIALIZE courseFound to true
                        IF nonAcademic_obj.getisRegistered is true
                            DISPLAY a dialog box with the message "The
                            course has already been registered"
                        ELSE
                            CALL the register method of nonAcademic_obj
                            with the parameters getCourseLeader(),
                            getInstructor, getStartDate(),
                            getCompletionDate() and getExamDate()
                            DISPLAY a dialog box with the message “The
                            course has been registered successfully”
                        END IF
                    END IF
                END FOR
                IF courseFound is false or courseList is empty
```

```
                    DISPLAY a dialog box with the message “The course has
                    not been added yet”
            END IF
        END IF
    END IF

    ELSE IF e.getSource() == displayNonAcademicButton
        INITIALIZE courseFound to false
        FOR each element course of courseList
            IF course is an instance of NonAcademicCourse
                INITILIAZE course to nonAcademic_obj by downcasting into
                NonAcademicCourse
                CALL the display method of nonAcademic_obj
                INTILIAZE courseFound to true
            END IF
        END FOR
        IF courseFound is false and courseList is empty
            DISPLAY a dialog box with the message "Non Academic Course
            has not been added yet."
        END IF
    END IF

    ELSE IF e.getSource() == removeButton
        IF getCourseId() is empty
            DISPLAY a dialog box with the message "Please enter the Course
            ID"
        ELSE
            INITIALIZE courseFound to false
            FOR each element course in courseList
                IF course.getcourseID() equals getCourseId() and course is
                an instance of NonAcademicCourse
```

```
                    INITIALIZE courseFound to true
                    INITIALIZE course nonAcademic_obj by downcasting
                        to NonAcademicCourse
                    IF nonAcademic_obj.getisRemoved == true
                        DISPLAY a dialog box with the message "The
                        course has already been removed"
                    ELSE
                        CALL remove method of nonAcademic_obj
                        DISPLAY a dialog box with the message "The
                        course has been removed successfully"
                    END IF
                END FOR
            IF courseFound is false and courseList is empty
                DISPLAY a dialog box with the message "The course has
                not been added yet"
            END IF
        END IF
    END IF

    ELSE IF e.getSource() == clearButton
        SET the text of courseIdField to " "
        SET the text of courseNameField to " "
        SET the text of durationField to " "
        SET the text of levelField to " "
        SET the text of creditField to " "
        SET the text of assessmentField to " "
        SET the text of prerequisiteField to " "
        SET the text of idRegisterField to " "
        SET the text of leaderField to " "
        SET the text of lecturerField to " "
        SET the text of instructorField to " "
```

```
    END IF
END DO
CREATE main(String[] args)
DO
    INITIALIZE an object INGCollege to ing
    CALL courseGui() method of ing
END DO
```

4. Method Description

Method	Description
1. void courseGui()	Contains objects of JFrame, JPanel, JLabel, JTextFields, JComboBox along with their bounds, font, and colors to create a GUI for course registration
2. String getCourseId()	Returns the courseID entered by the user to add a course.
3. String getCourseName()	Returns the course name entered by the user.
4. String getDuration()	Returns the integer value of the duration of the course entered by the user.
5. String getLevel()	Returns the level entered by the user.
6. String getCredit()	Returns the credit entered by the user.
7. String getNoOfAssessments()	Returns the integer value of the number of assessments entered by the user.
8. String getPrerequisite()	Returns the prerequisite entered by the user.
9. String getRegisterId()	Returns the course ID entered by the user to register a course.

10. String getCourseLeader()	Returns the name of the course leader entered by the user.
11. String getLecturer()	Returns the name of the course leader entered by the user.
12. String getInstructor()	Returns the name of the instructor entered by the user.
13. String getStartDate()	• Concatenates the year, month and date entered by the user. • Returns the concatenated date on which the course starts.
14. String getCompletionDate()	• Concatenates the year, month and date entered by the user. • Returns the concatenated date on which the course completes.
15. String getExamDate()	• Concatenates the year, month and date entered by the user. • Returns the concatenated date of exam.
16. void actionPerformed(ActionEvent e)	• Checks which button has been clicked by the user. • Performs certain course of action such as changing the visibility, setting the text, displaying dialog boxes and so on when a particular button is clicked.
16.1 e.getSource() == nonAcademicButton	Visibility of JLabel, JTextField, JButton are changed in the panels when the nonAcademicButton is clicked so that only the fields required for Non Academic Course will be displayed.
16.2 e.getSource() == academicButton	Visibility of JLabel, JTextField, JButton are changed in the panels when the academicButton is clicked so that only the fields required for Academic Course

	will be displayed.
16.3 e.getSource() == addAcademicButton	An object of AcademicCourse is created which takes the course ID, course name, duration, level, credit and number of assessments entered by the user as parameter. The object is then added to the ArrayList: courseList.
16.4 e.getSource() == registerAcademicButton	• Checks if the ID entered by the user has been added to the ArrayList. • Downcasts an object of Course class to AcademicCourse. • Calls the register method from the AcademicCourse class if the ID is found in the ArrayList and the course has not been registered already.
16.5 e.getSource() == displayAcademicButton	• Searches the courseList for instance of AcademicCourse. • Calls the display method from AcademicCourse class to print the details of the academic courses.
16.6 e.getSource() == addNonAcademicButton	• An object of NonAcademicCourse is created which takes the course ID, course name, duration and prerequisite entered by the user as parameter. • The object is then added to the ArrayList: courseList.
16.7 e.getSource() == registerNonAcademicButton	• Checks if the ID entered by the user has been added to the ArrayList. • Downcasts an object of Course class to

	NonAcademicCourse. • Calls the register method from the NonAcademicCourse class if the ID is found in the ArrayList and the course has not been registered already.
16.8 e.getSource() == displayNonAcademicButton	• Searches the courseList for instance of NonAcademicCourse. • Calls the display method from NonAcademicCourse class to print the details of the nonacademic courses.
16.9 e.getSource() == removeNonAcademicButton	• Checks if the ID entered by the user has been added to the courseList and if it is an instance of NonAcademicCourse. • Downcasts an object of Course class to NonAcademicCourse. • If the course is found, remove method is called from the NonAcademicCourse class and the course gets removed.
16.10 e.getSource() == clearButton	Clears all the fields in the GUI by setting the text of all the JTextFields to an empty string(“ ”).

Table 2 Method Description

5. Testing

5.1 Test 1 – Command Prompt Test

Test No.	1
Objective	To check if the program can be compiled and run using command prompt.
Action	• Open the file which contains the .java files. • Open command prompt through the file location. • Enter the following into the command prompt: ➢ Course.java ➢ AcademicCourse.java ➢ NonAcademicCourse.java ➢ INGCollege.java • Once all the classes are compiled enter the following into the command prompt to run the program: ➢ java INGCollege
Expected Result	All the classes will be compiled and the GUI of INGCollege will be displayed on the screen.
Output	• All the classes compiled. • The code of INGCollege could run and the GUI got displayed.
Conclusion	The test was completed successfully.

Table 3 Test to check if the program can be compiled and run using command prompt.

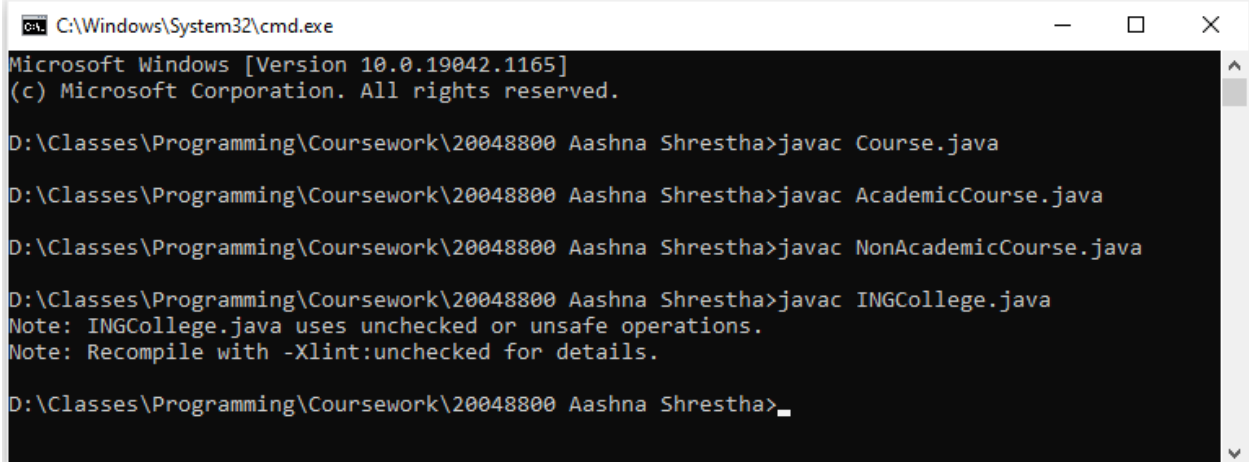

Figure 1 Compiling the classes

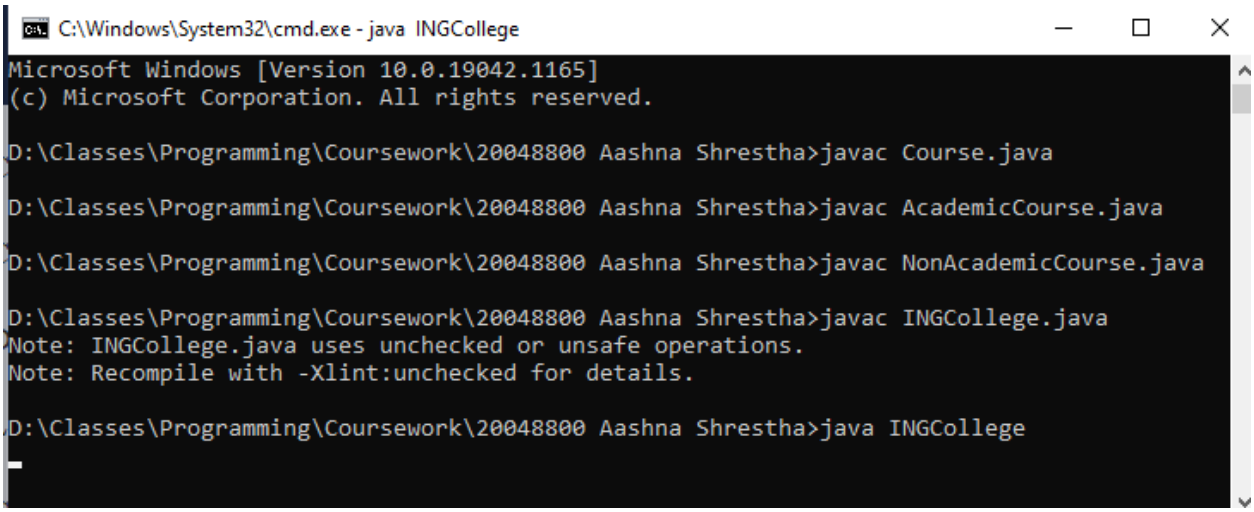

Figure 2 Run the class INGCollege

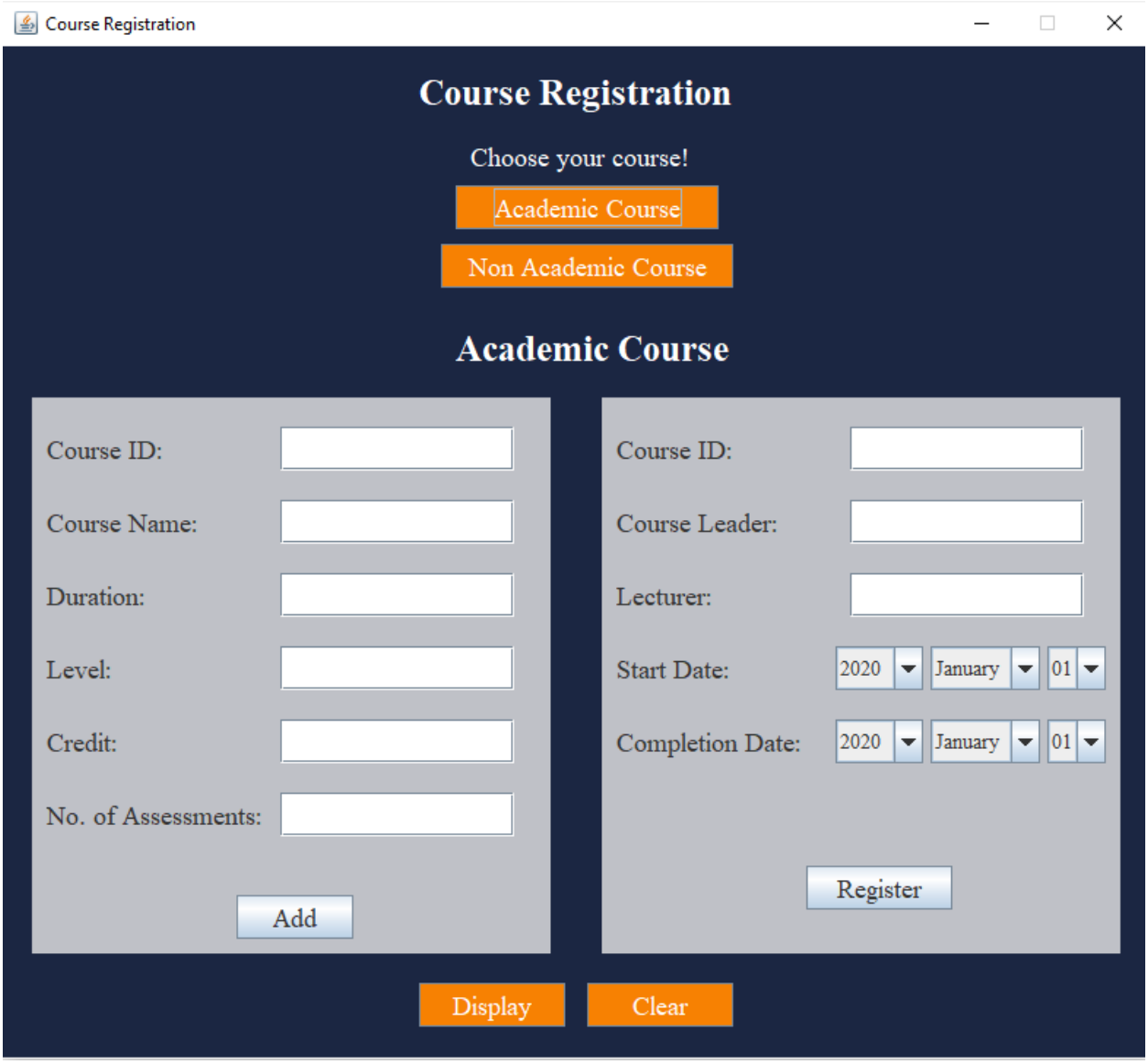

Figure 3 GUI opened through command prompt

5.2 Test 2

5.2.1 Adding Academic Course

Test No.	2.1
Objective	To check if the academic course can be added.
Action	• Call courseGui() method through the main method. • Ensure that the Academic Course is being displayed. • Fill all the fields in the addCoursePanel with the following details: ➢ Course ID – "1" ➢ Course Name – "Programming" ➢ Duration – 2 ➢ Level – "4" ➢ Credit – "3" ➢ No. of Assessments – 2 • Click the button named Add.
Expected Result	The course must be added to the arraylist named courseList and a dialog box will appear to alert the user that the course has been added.
Output	A dialog box appeared with the message "Academic Course has been added successfully."
Conclusion	The test was completed successfully.

Table 4 Test to check if the academic course can be added.

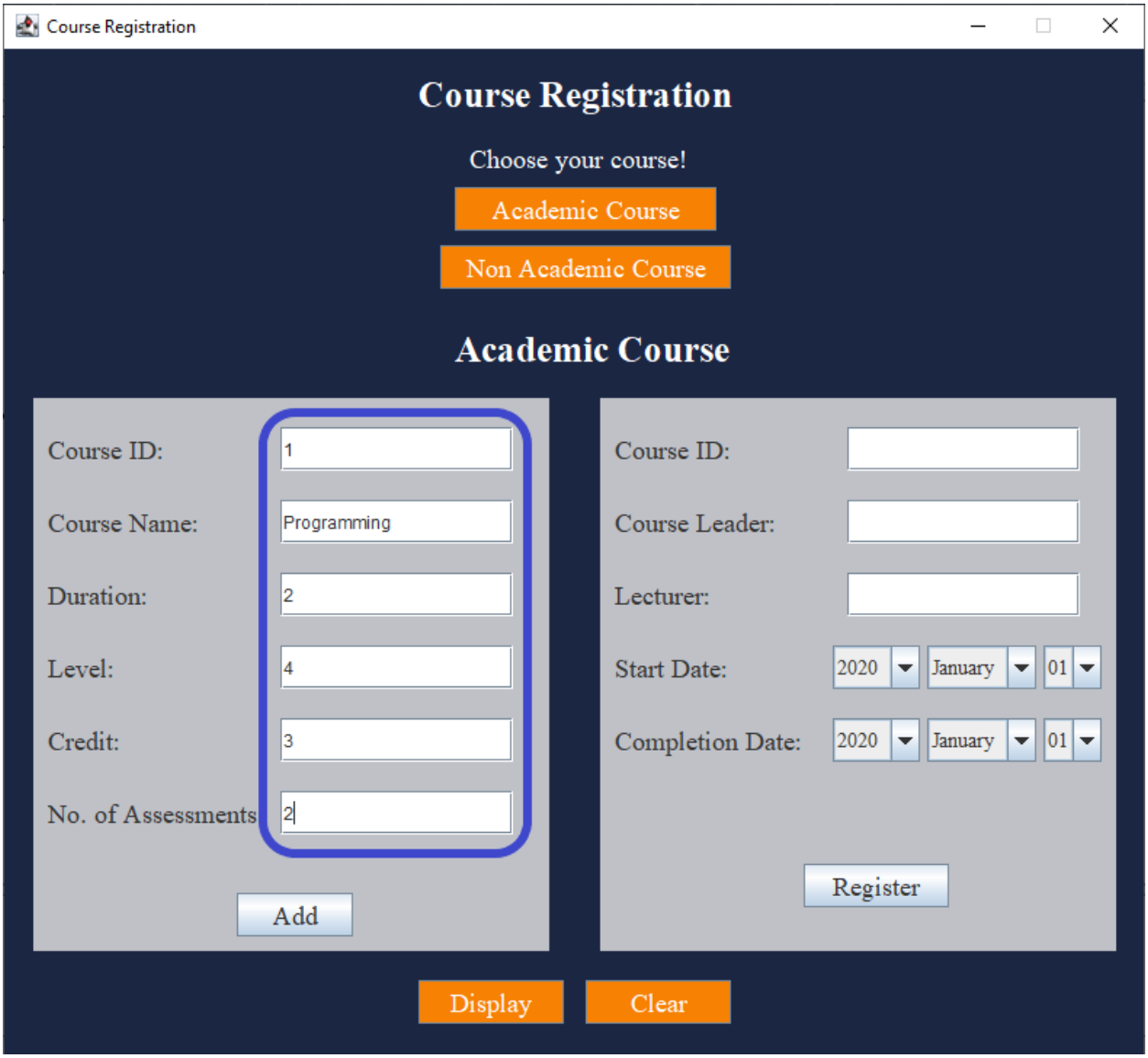

Figure 4 Adding the course details of Academic Course

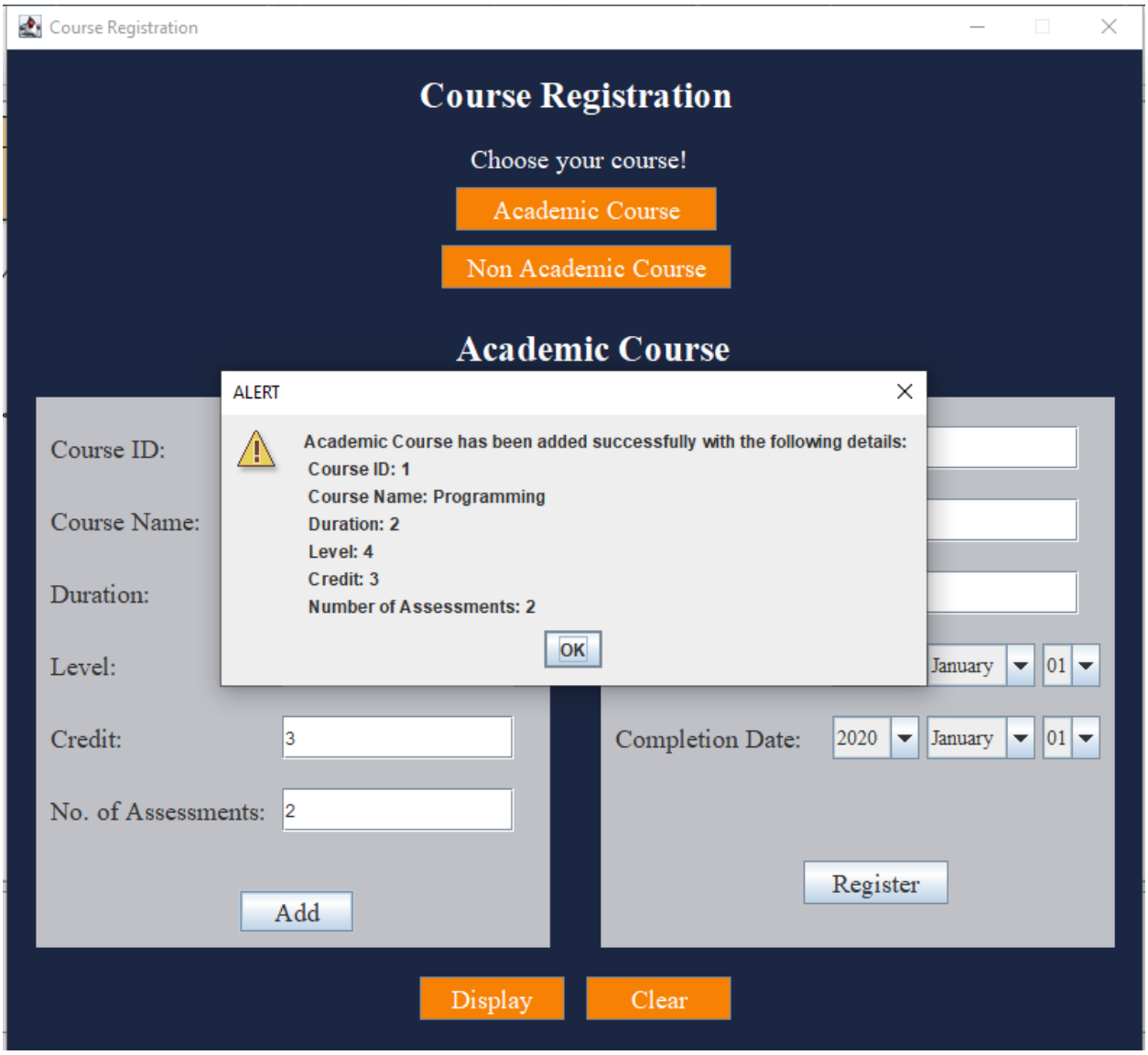

Figure 5 Dialog box alerting the user that the course has been added

5.2.2 Adding Non-Academic Course

Test No.	2.2
Objective	To check if the non-academic course can be added.
Action	• Call courseGui() method through the main method. • Click on the button named Non Academic Course to display the fields required for the non-academic course • Fill the following fields from the addCoursePanel: ➢ Course ID – "2" ➢ Course Name – "Presentation Skills" ➢ Duration – 1 ➢ Prerequisite – "Basics of MS Powerpoint" • Click the button named Add.
Expected Result	The course must be added to the arraylist named courseList and a dialog box will appear to alert the user that the course has been added.
Output	• A dialog box appeared with the message "Non Academic Course has been added successfully."
Conclusion	The test was completed successfully.

Table 5 Test to check if the non-academic course can be added.

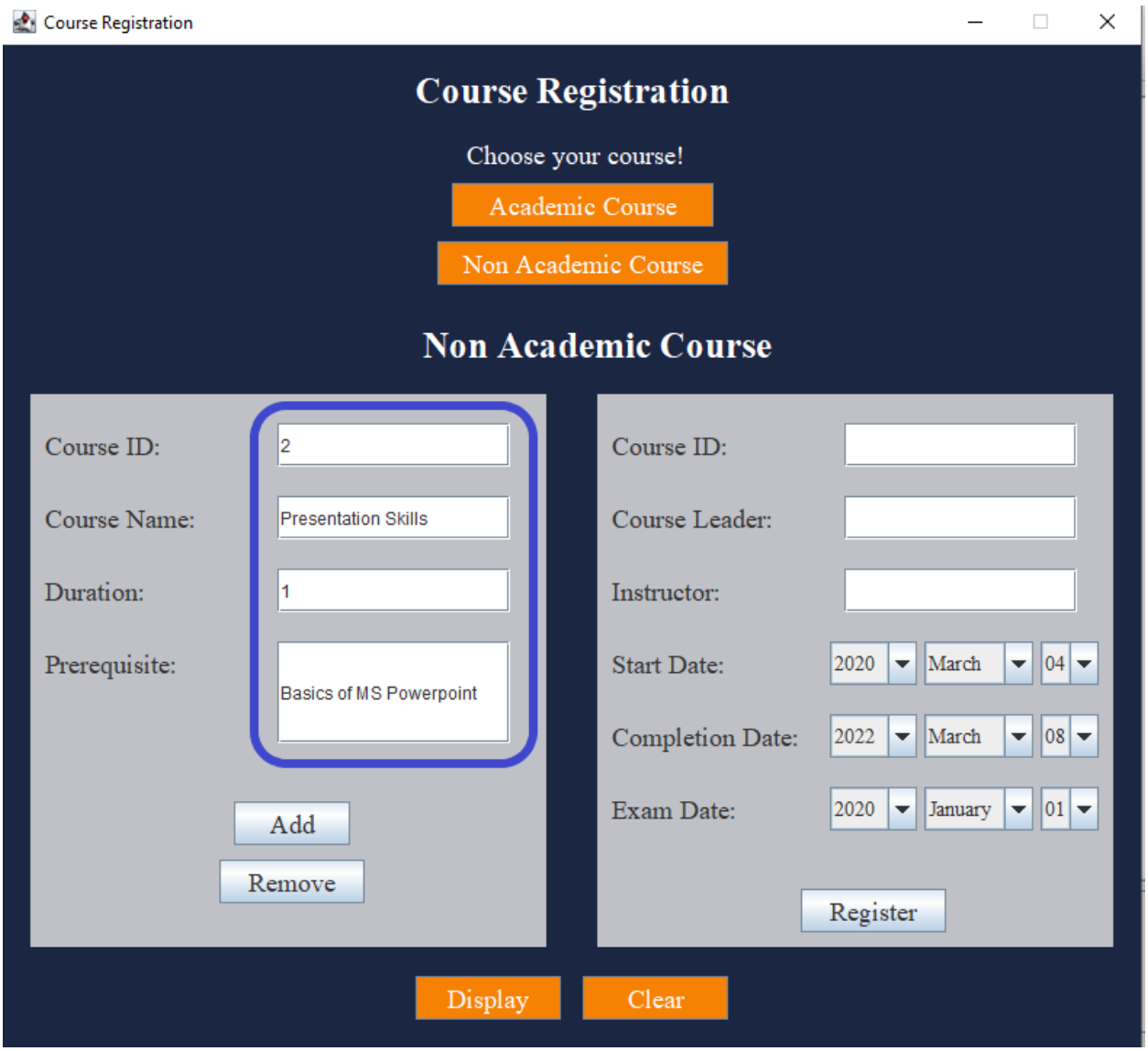

Figure 6 Adding the details of a Non Academic Course

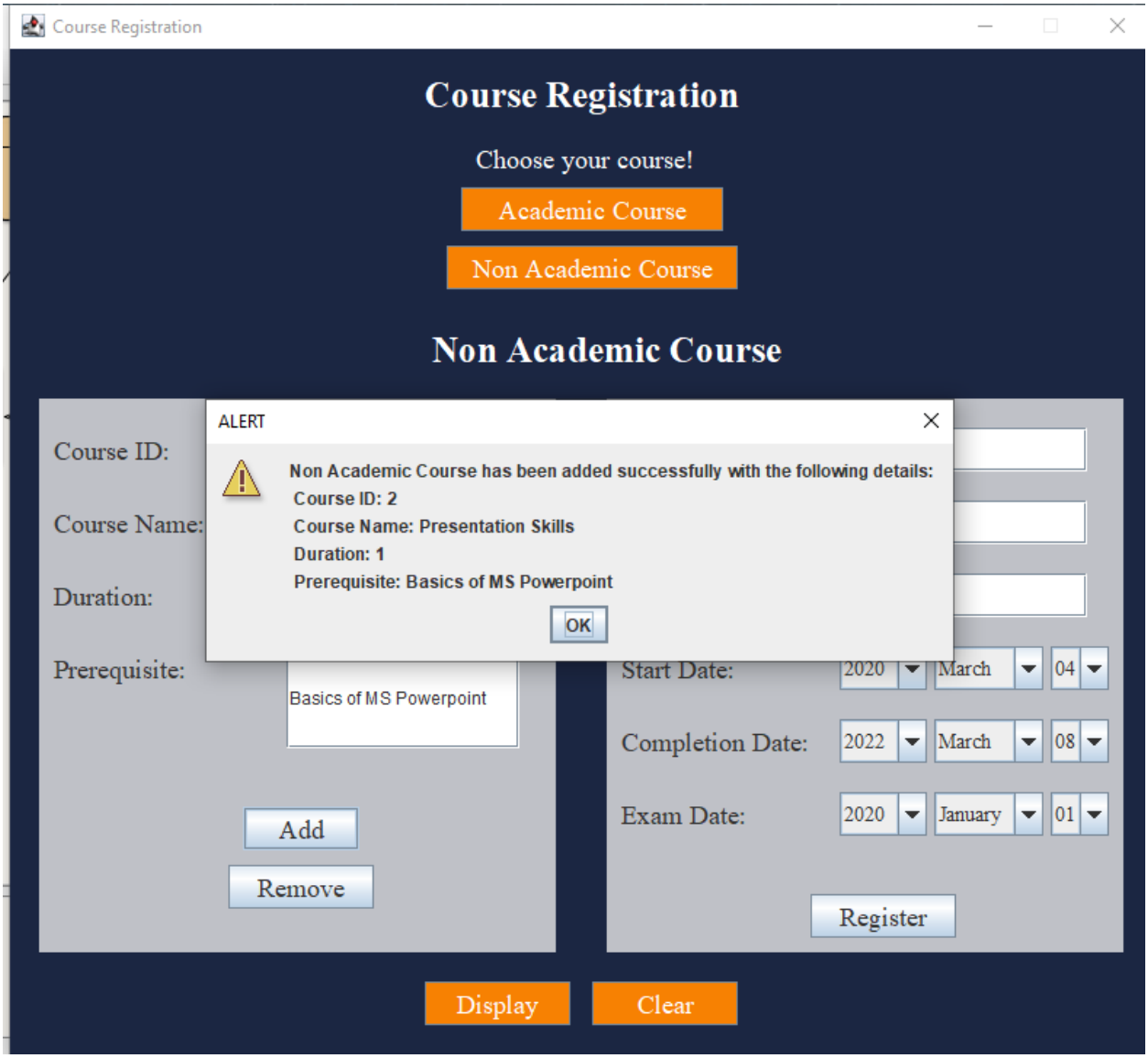

Figure 7 Dialog box alerting the user that the course has been added

5.2.3 Registering Academic Course

Test No.	2.3
Objective	To check if the academic course can be registered.
Action	• Call courseGui() method through the main method. • Ensure that the Academic Course is being displayed. • Add a course to the arraylist with the steps mentioned in test 2.1 • Fill the following fields from the registerPanel: ➢ Course ID – "1" ➢ Course Leader – "John Smith" ➢ Lecturer – "Rachel Reece" ➢ Start Date – "2020 March 04" ➢ Completion Date – "2020 March 08" • Click the button named Register.
Expected Result	• The course must be registered and a dialog box should alert the user that it has been registered. • A terminal must display all the fields.
Output	• A dialog box appeared with the message "The course has been registered successfully." • A terminal displayed the course details.
Conclusion	The test was completed successfully.

Table 6 Test to check if the academic course can be registered.

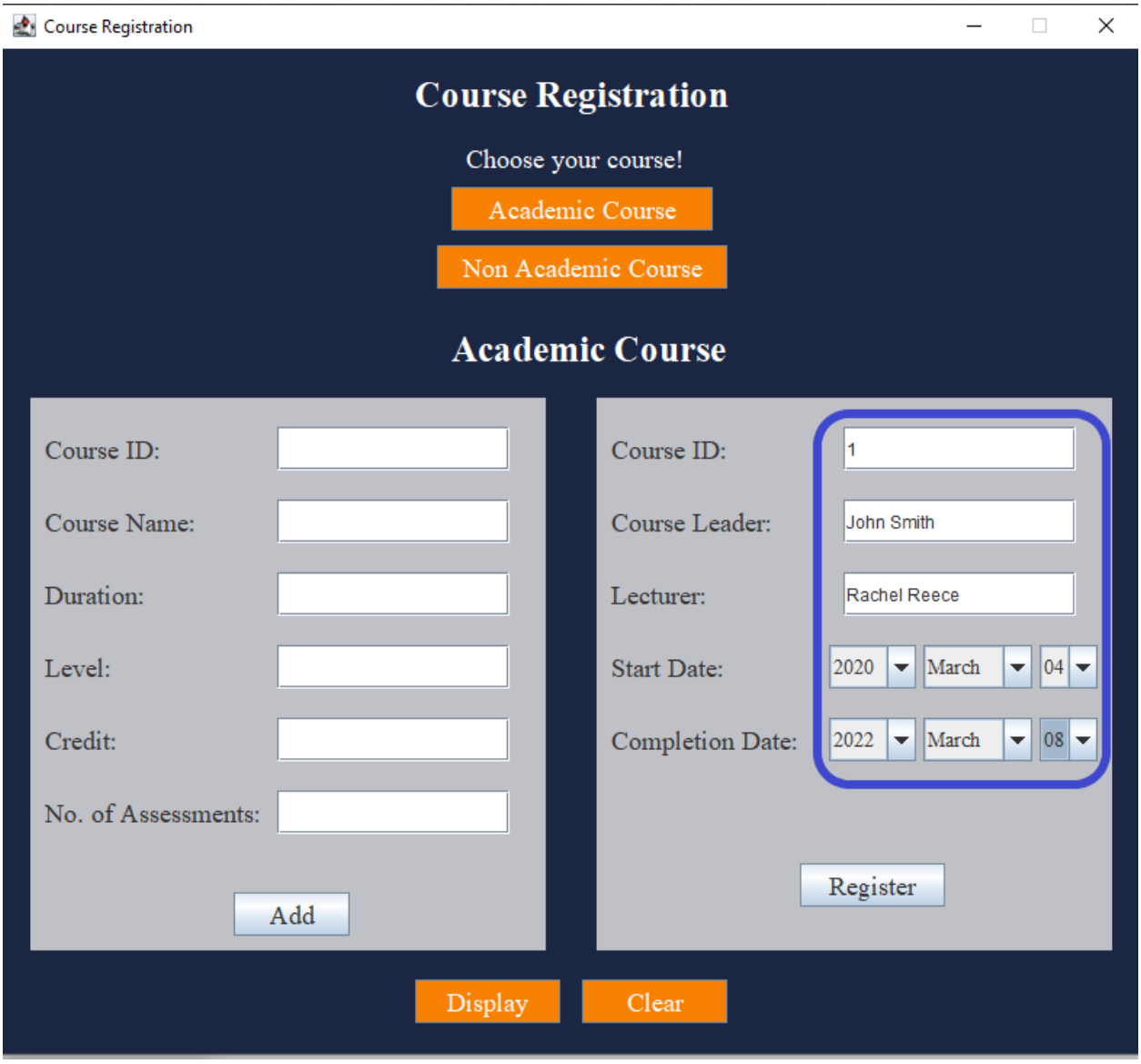

Figure 8 Details to register an academic course

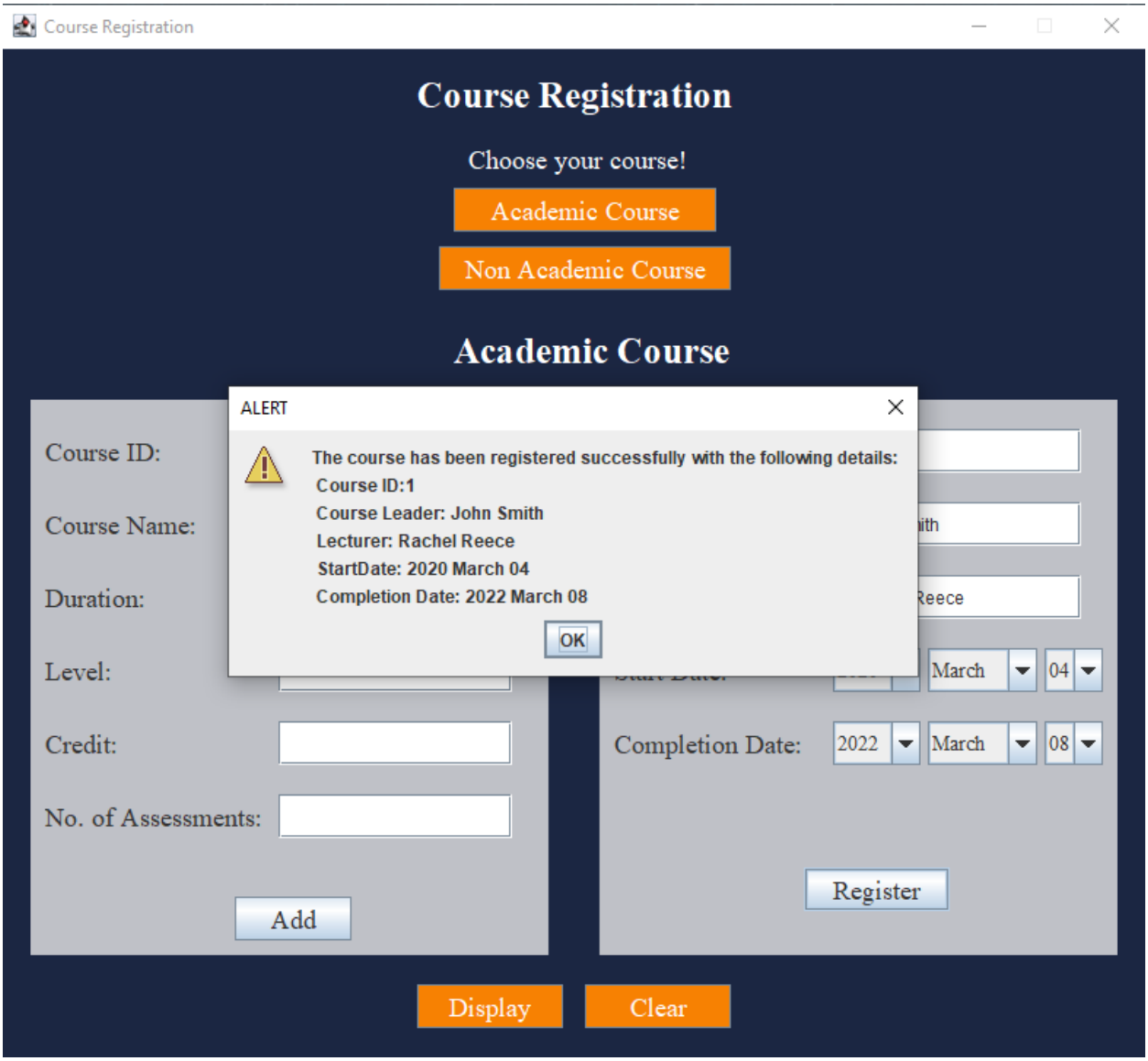

Figure 9 Dialog box alerting the user that the course has been registered

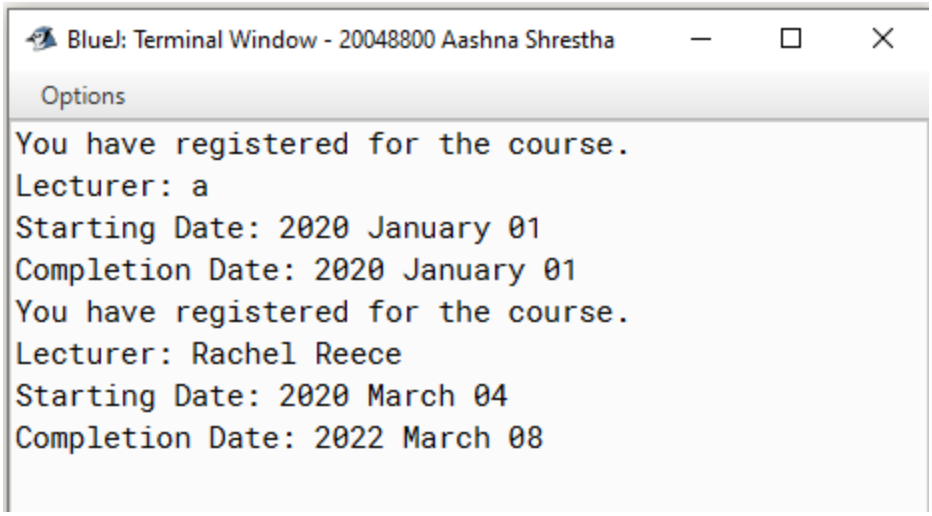

Figure 10 A terminal which displays the course details after the course is registered

5.2.4 Registering Non Academic Course

Test No.	2.4
Objective	To check if the non academic course can be registered.
Action	• Call courseGui() method through the main method. • Click on the button named Non Academic Course to display the fields required for the non-academic course • Add a course to the arraylist with the steps mentioned in test 2.2 • Fill the following fields from the registerPanel: ➢ Course ID – “2” ➢ Course Leader – “Max Brown” ➢ Instructor – “Rita Rose” ➢ Start Date – “2021 June 02” ➢ Completion Date – “2022 May 16”

	➢ Exam Date – "2022 June 03" • Click the button named Register.
Expected Result	The course must be registered and a dialog box should alert the user that it has been registered.
Output	A dialog box appeared with the message "The course has been registered successfully."
Conclusion	The test was completed successfully.

Table 7 Test to check if the non academic course can be registered.

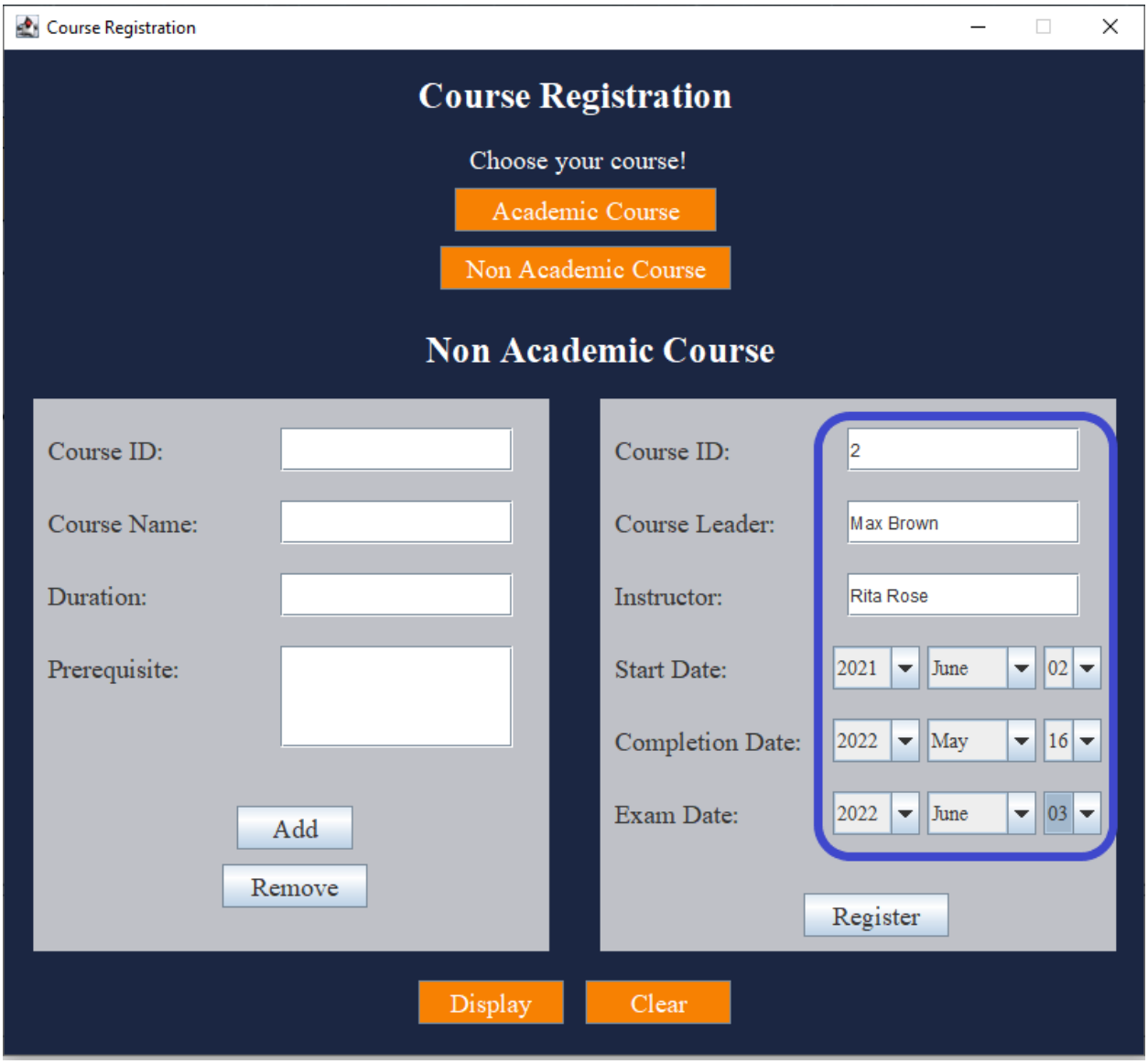

Figure 11 Details to register a Non Academic Course

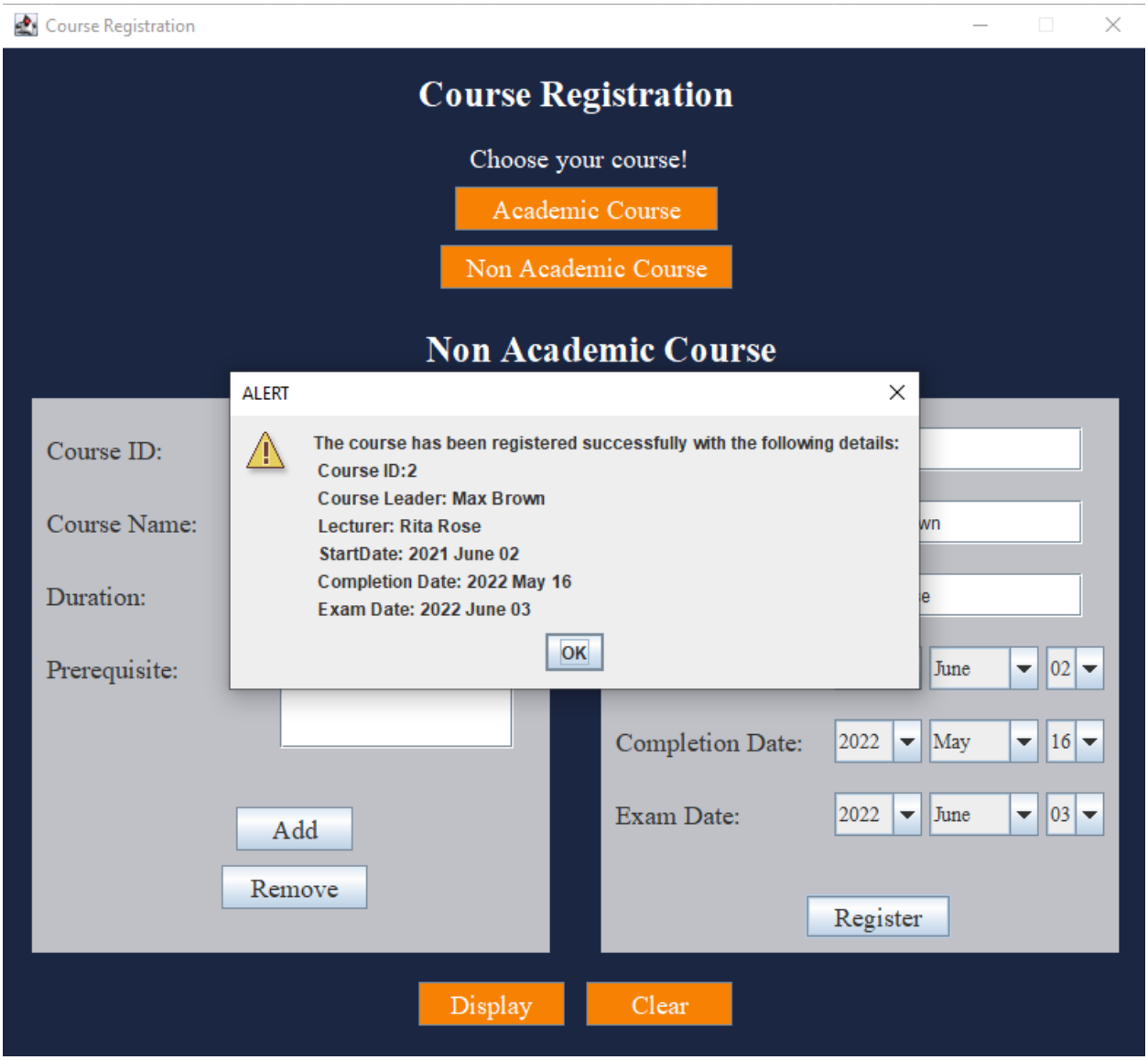

Figure 12 Dialog box alerting the user that the non academic course has been registered

5.2.5 Removing a Non Academic Course

Test No.	2.5
Objective	To check if the non academic course can be removed.
Action	• Call courseGui() method through the main method. • Click on the button named Non Academic Course to display the fields required for the non-academic course • Add a course to the arraylist with the steps mentioned in test 2.2 • Register the course with the steps mentioned in test 2.4 ➢ Fill the Course ID in the addCoursePanel • Click the button named Remove.
Expected Result	The course must be removed and a dialog box should alert the user that it has been removed.
Output	A dialog box appeared with the message “The course has been removed successfully.”
Conclusion	The test was completed successfully.

Table 8 Test to check if the non academic course can be removed.

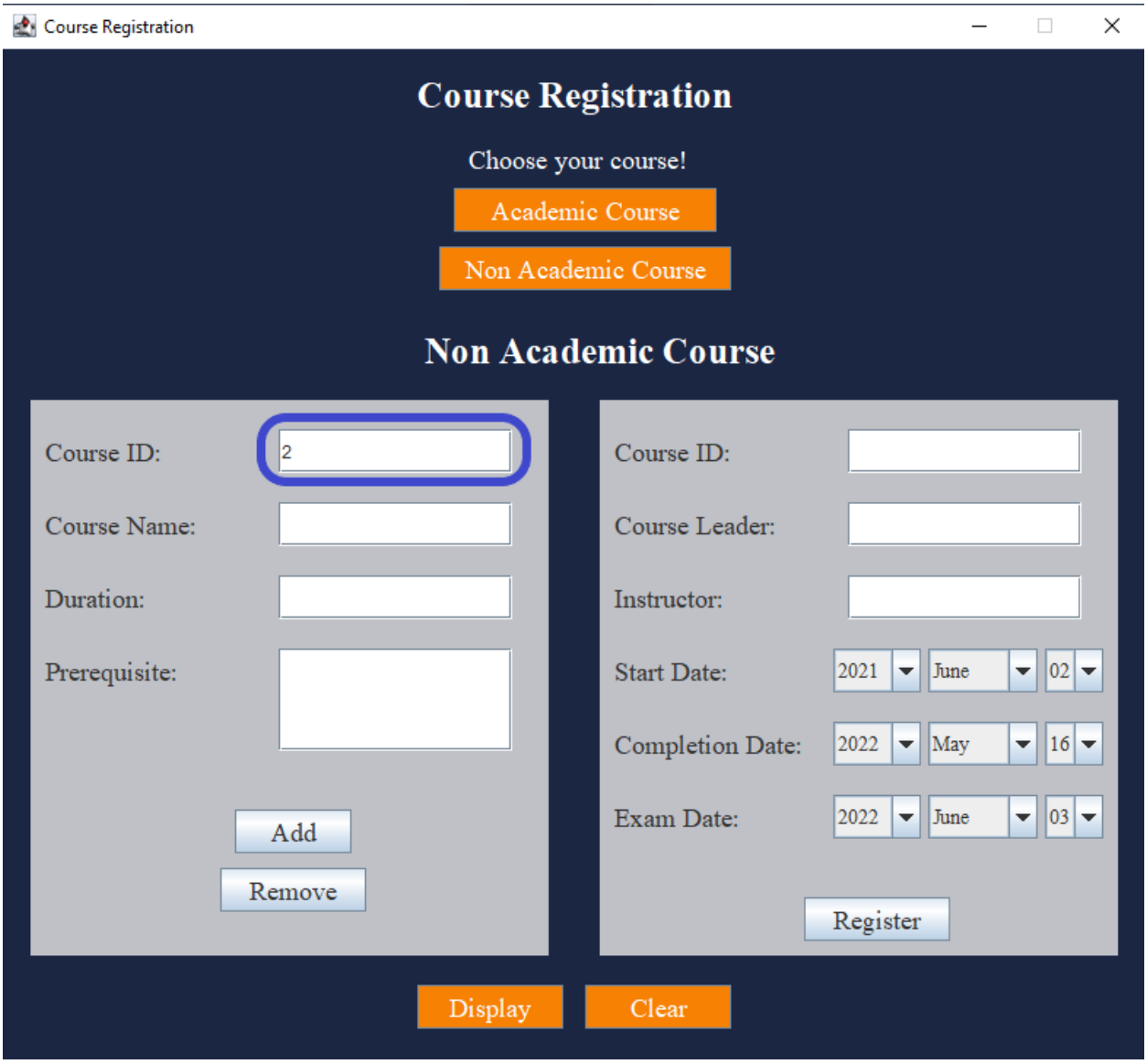

Figure 13 Course ID of the course to be removed

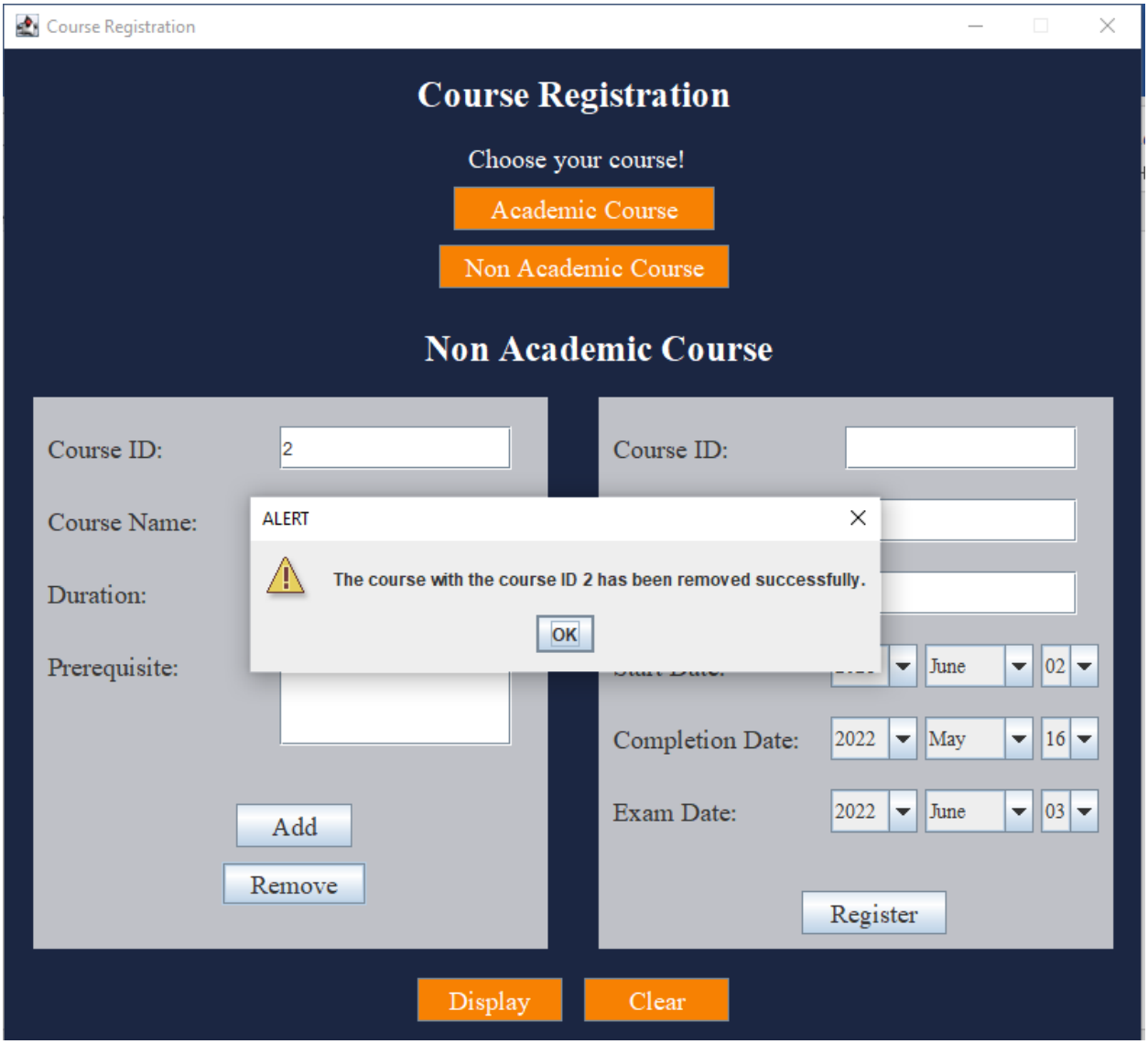

Figure 14 Dialog box alerting the user that the course has been removed successfully

5.3 Test 3

5.3.1 Adding a duplicate course ID for Academic Course

Test No.	3.1
Objective	To check if a course ID of academic course can be added more than once.
Action	• Call courseGui() method through the main method. • Ensure that the Academic Course is being displayed. • Fill the following fields from the addCoursePanel: ➢ Course ID ➢ Course Name ➢ Duration ➢ Level ➢ Credit ➢ No. of Assessments • Click the button named Add. • Once the dialog box appears alerting that the course has been added, add the same values again into the respective fields.
Expected Result	The course must not be added to the arraylist at the second attempt and a dialog box must appear to alert the user.
Output	A dialog box appeared with the message “The course has been added already.
Conclusion	The test was completed successfully.

Table 9 Test to check if a course ID of academic course can be added more than once.

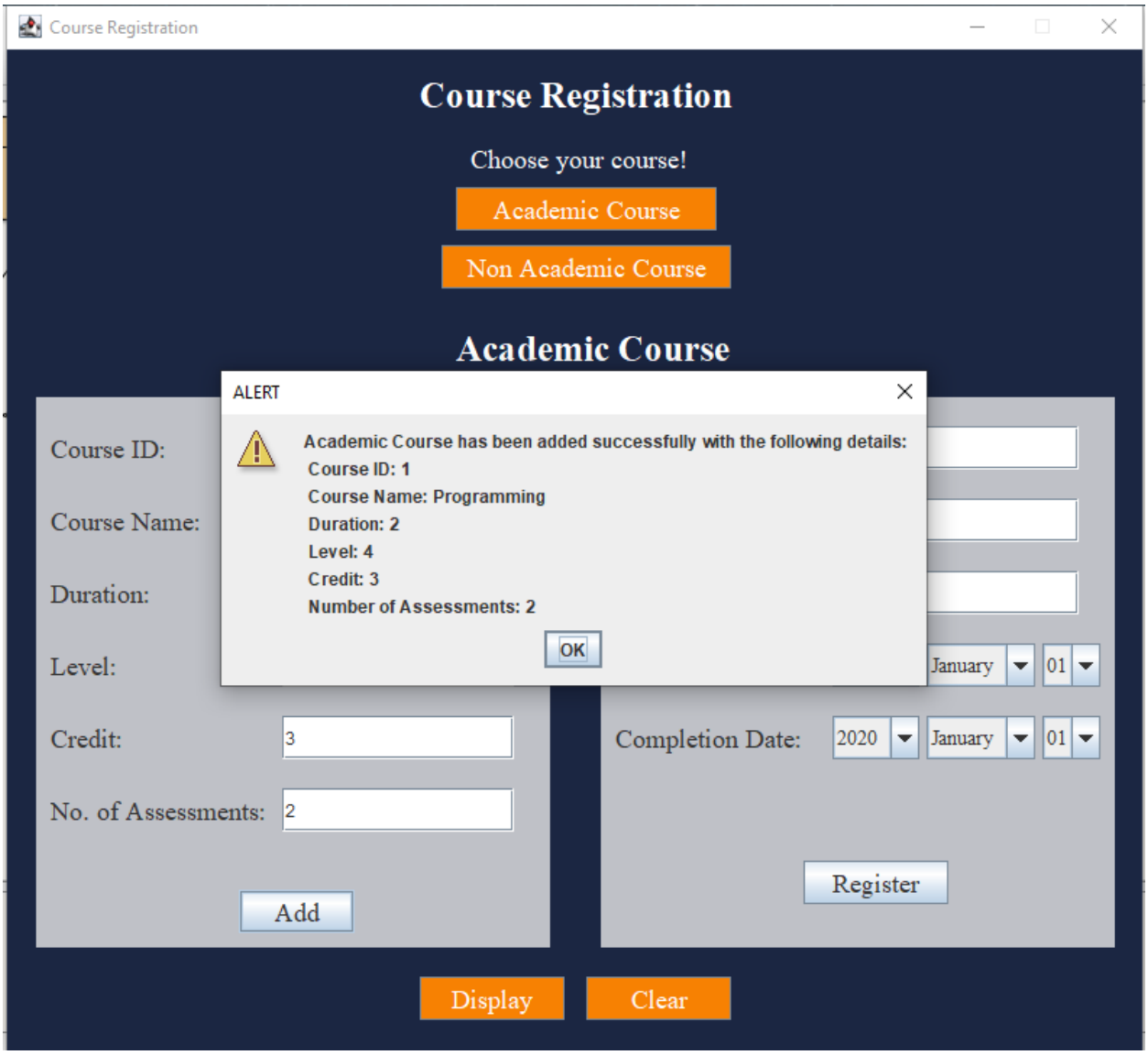

Figure 15 Adding an academic Course

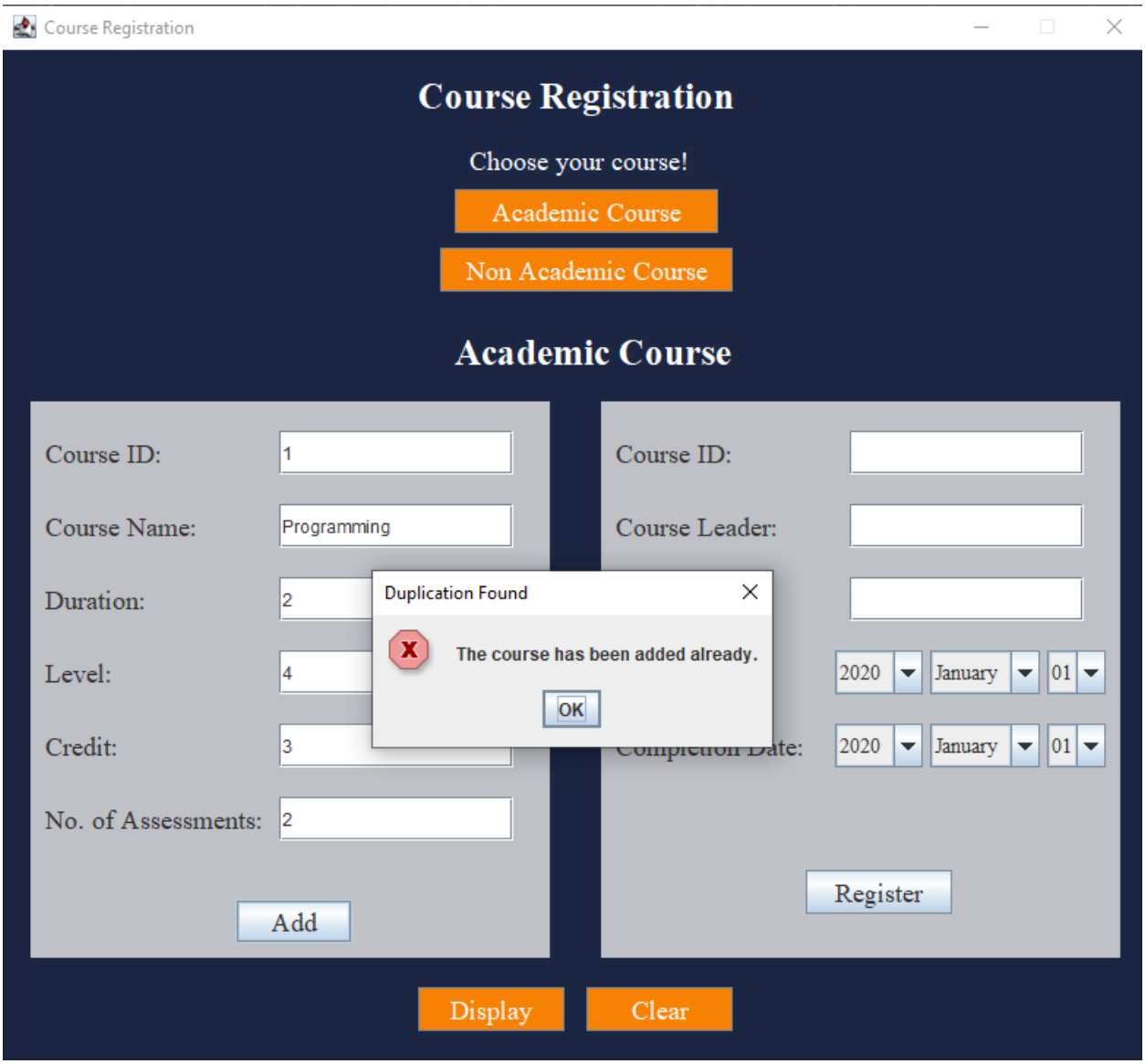

Figure 16 Adding a course with the same course ID

5.3.2 Adding a duplicate course ID for Non Academic Course

Test No.	3.2
Objective	To check if a course ID of non academic course can be added more than once.
Action	• Call courseGui() method through the main method. • Click on the button named Non Academic Course to display the fields required for the non-academic course • Fill the following fields from the addCoursePanel: ➢ Course ID ➢ Course Name ➢ Duration ➢ Prerequisite • Click the button named Add. • Once the dialog box appears alerting that the course has been added, add the same values again into the respective fields.
Expected Result	The course must not be added to the arraylist at the second attempt and a dialog box must appear to alert the user.
Output	A dialog box appeared with the message “The course has been added already.
Conclusion	The test was completed successfully.

Table 10 Test to check if a course ID of non academic course can be added more than once.

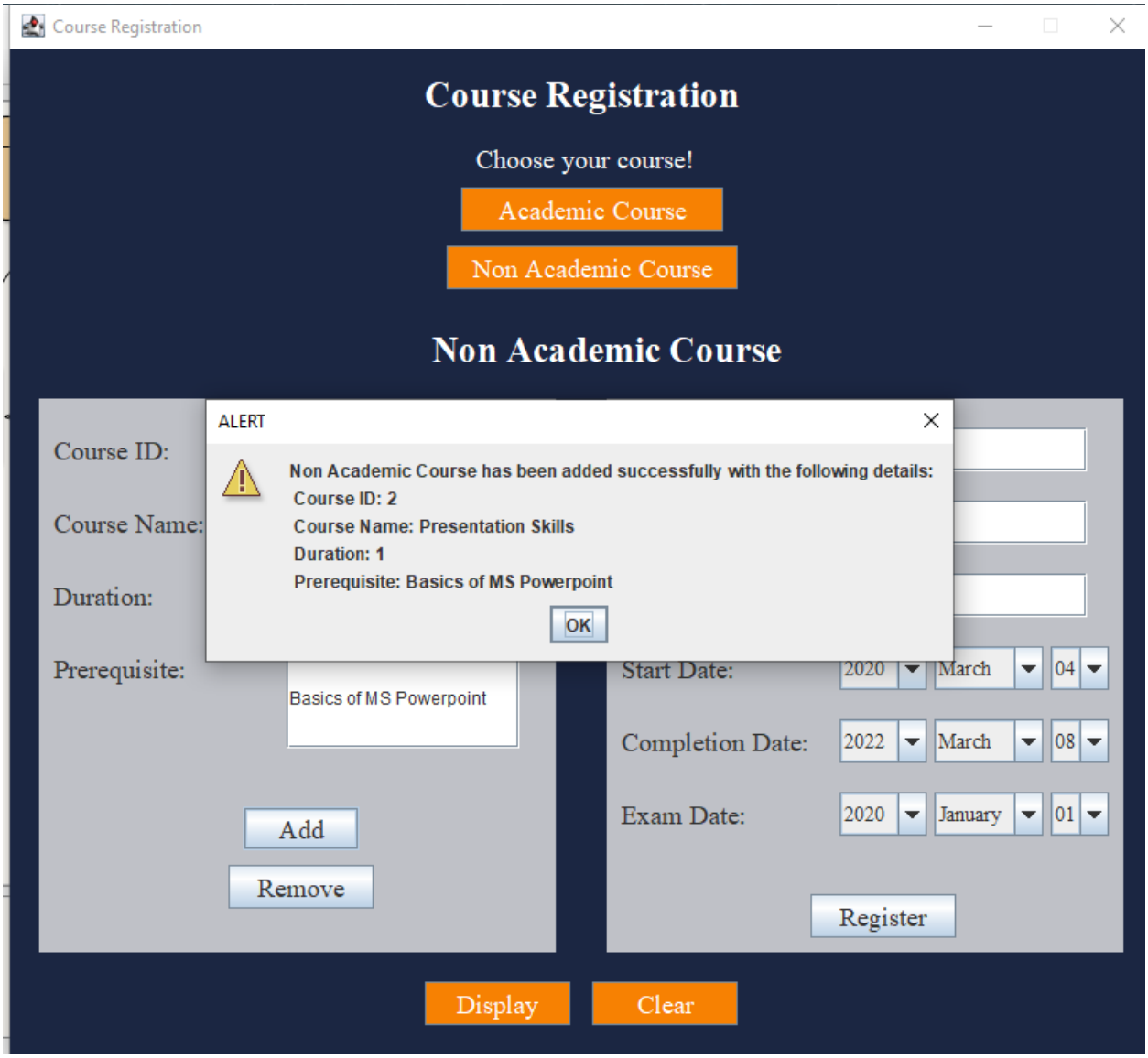

Figure 17 Adding a non academic course

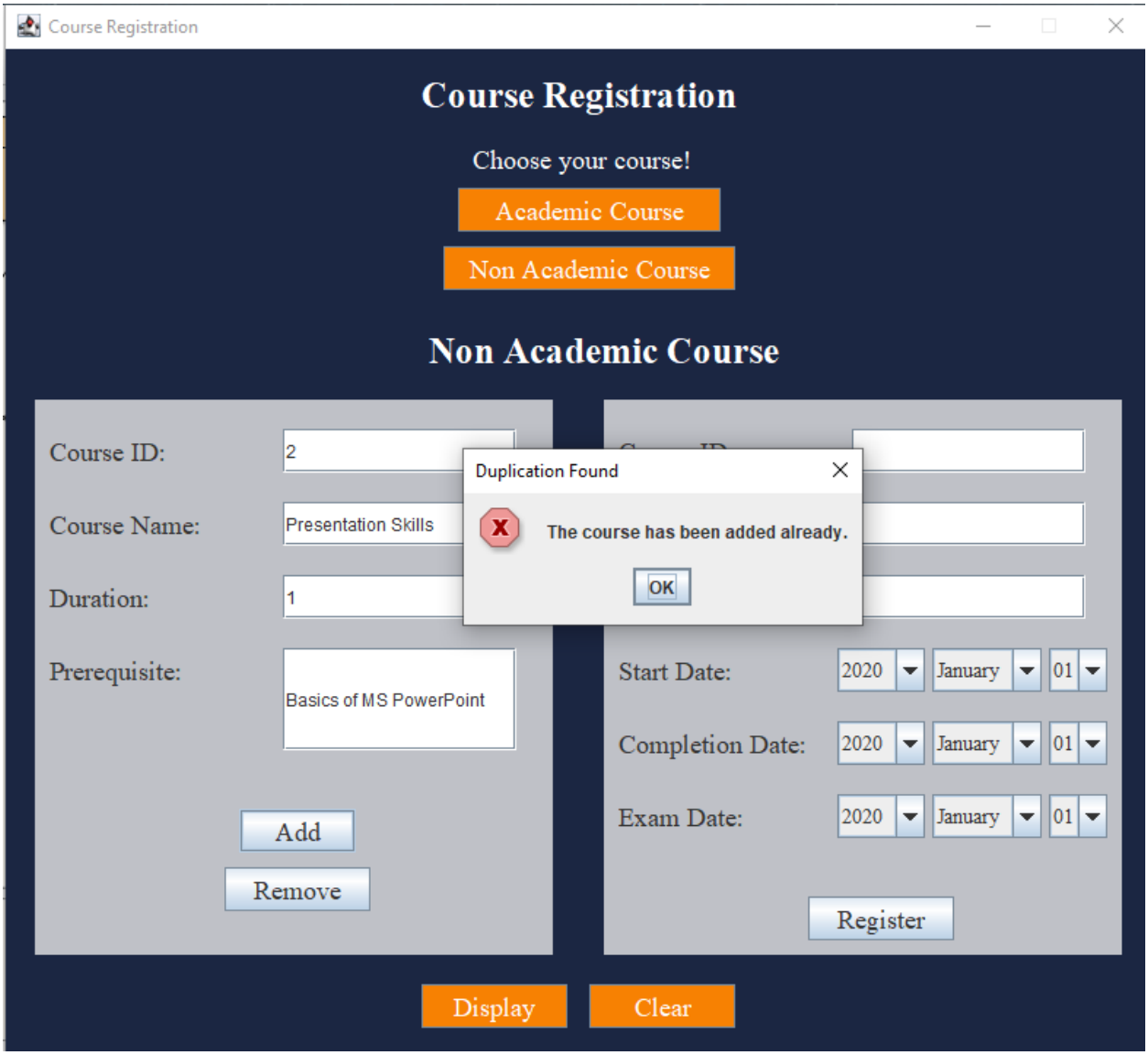

Figure 18 Adding a non academic course with the same courseID

5.3.3 Registering an academic course which has already been registered

Test No.	3.3
Objective	To check if an academic course can be registered more than once.
Action	• Call courseGui() method through the main method. • Ensure that the Academic Course is being displayed. • Add a course to the arraylist with the steps mentioned in test 2.1 • Fill the following fields from the registerPanel: ➢ Course ID ➢ Course Leader ➢ Lecturer ➢ Start Date ➢ Completion Date • Click the button named Register. • Once the dialog box appears alerting that the course has been registered, add the same values again into the respective fields.
Expected Result	The course must not be registered at the second attempt and a dialog box must appear to alert the user.
Output	A dialog box appeared with the message “The course has already been registered”
Conclusion	The test was completed successfully.

Table 11 Test to check if an academic course can be registered more than once.

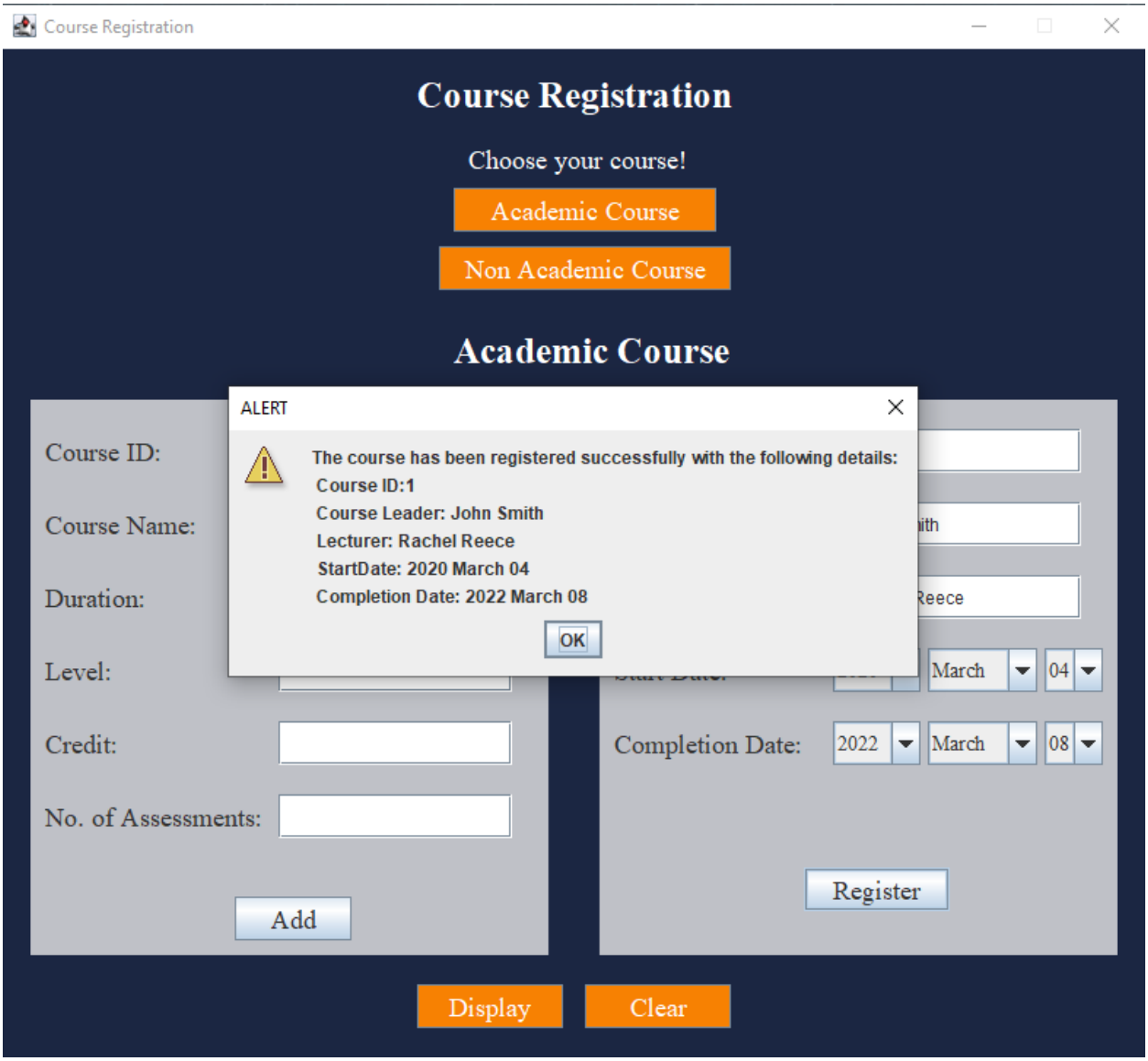

Figure 19 Registering an academic course

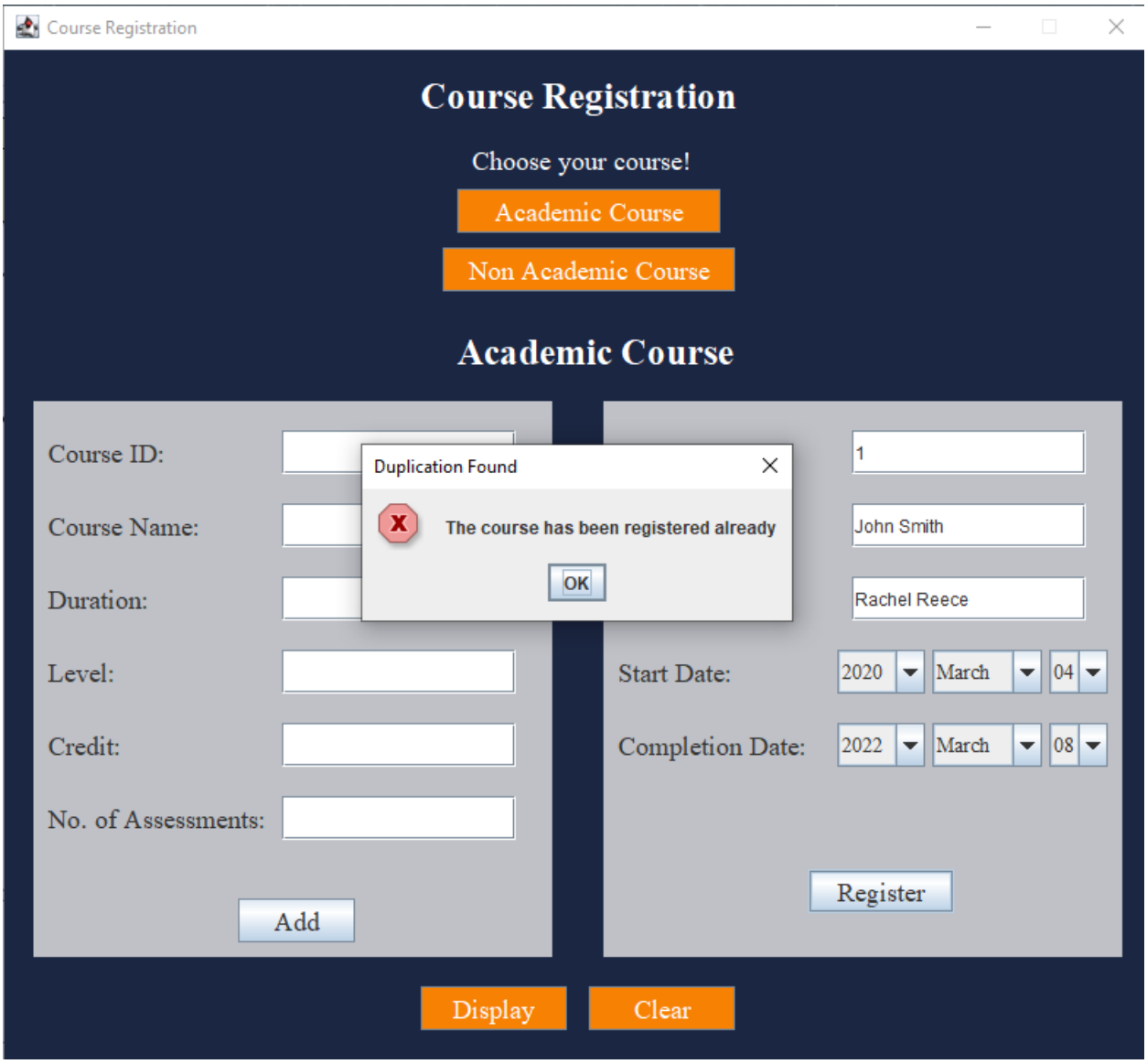

Figure 20 Registering an academic course which has been registered already

5.3.4 Registering a non academic course which has already been registered

Test No.	3.4
Objective	To check if a non academic course can be registered more than once.
Action	• Call courseGui() method through the main method. • Click on the button named Non Academic Course to display the fields required for the non-academic course • Add a course to the arraylist with the steps mentioned in test 2.2 • Fill the following fields from the registerPanel: ➢ Course ID ➢ Course Leader ➢ Instructor ➢ Start Date ➢ Completion Date ➢ Exam Date • Click the button named Register. • Once the dialog box appears alerting that the course has been registered, add the same values again into the respective fields.
Expected Result	The course must not be registered at the second attempt and a dialog box must appear to alert the user.
Output	A dialog box appeared with the message “The course has already been registered”
Conclusion	The test was completed successfully.

Table 12 Test to check if a non academic course can be registered more than once.

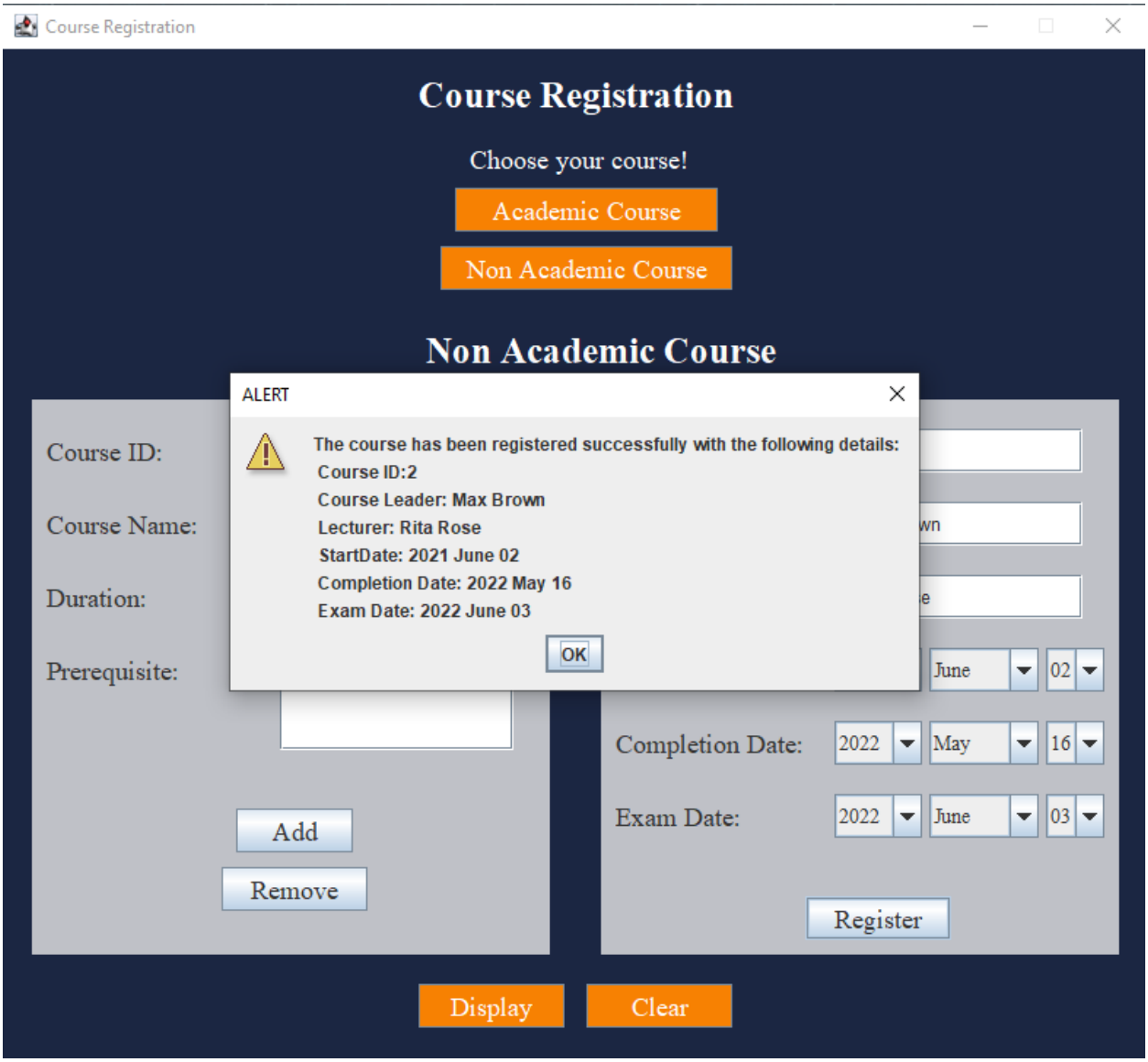

Figure 21 Registering a non academic course

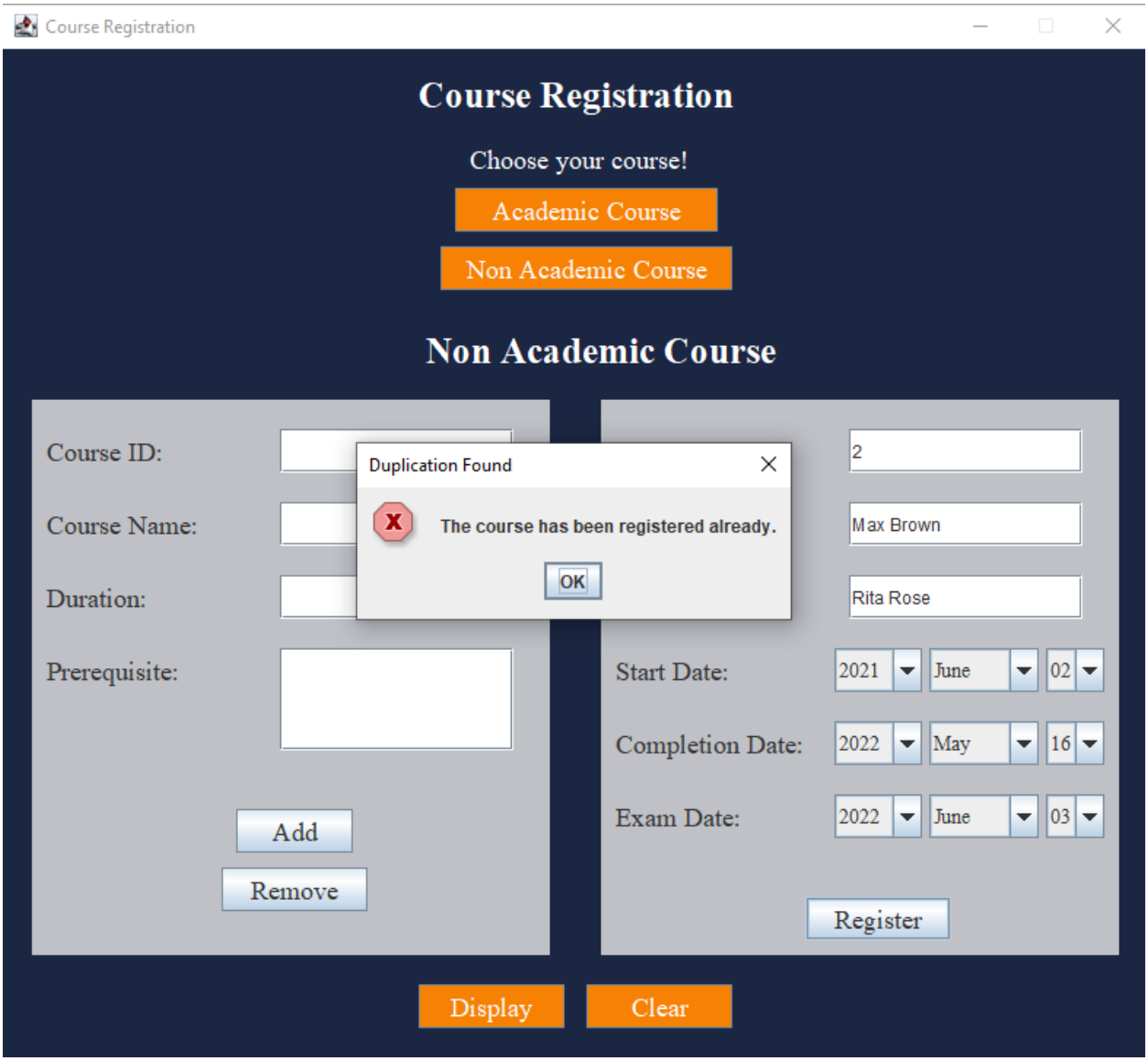

Figure 22 Registering a non academic course which has been registered already

5.3.5 Removing a non academic course which has been removed already

Test No.	3.5
Objective	To check if a non academic course can be removed more than once.
Action	• Call courseGui() method through the main method. • Click on the button named Non Academic Course to display the fields required for the non-academic course • Add a course to the arraylist with the steps mentioned in test 2.2 • Register the course with the steps mentioned in test 2.4 ➢ Fill the Course ID in the addCoursePanel • Click the button named Remove. • Once the dialog box appears alerting that the course has been removed, enter the same course ID into the text field.
Expected Result	The course will not be able to remove at the second attempt and a dialog box must appear to alert the user.
Output	A dialog box appeared with the message “The course has already been removed.”
Conclusion	The test was completed successfully.

Table 13 Test to check if a non academic course can be removed more than once.

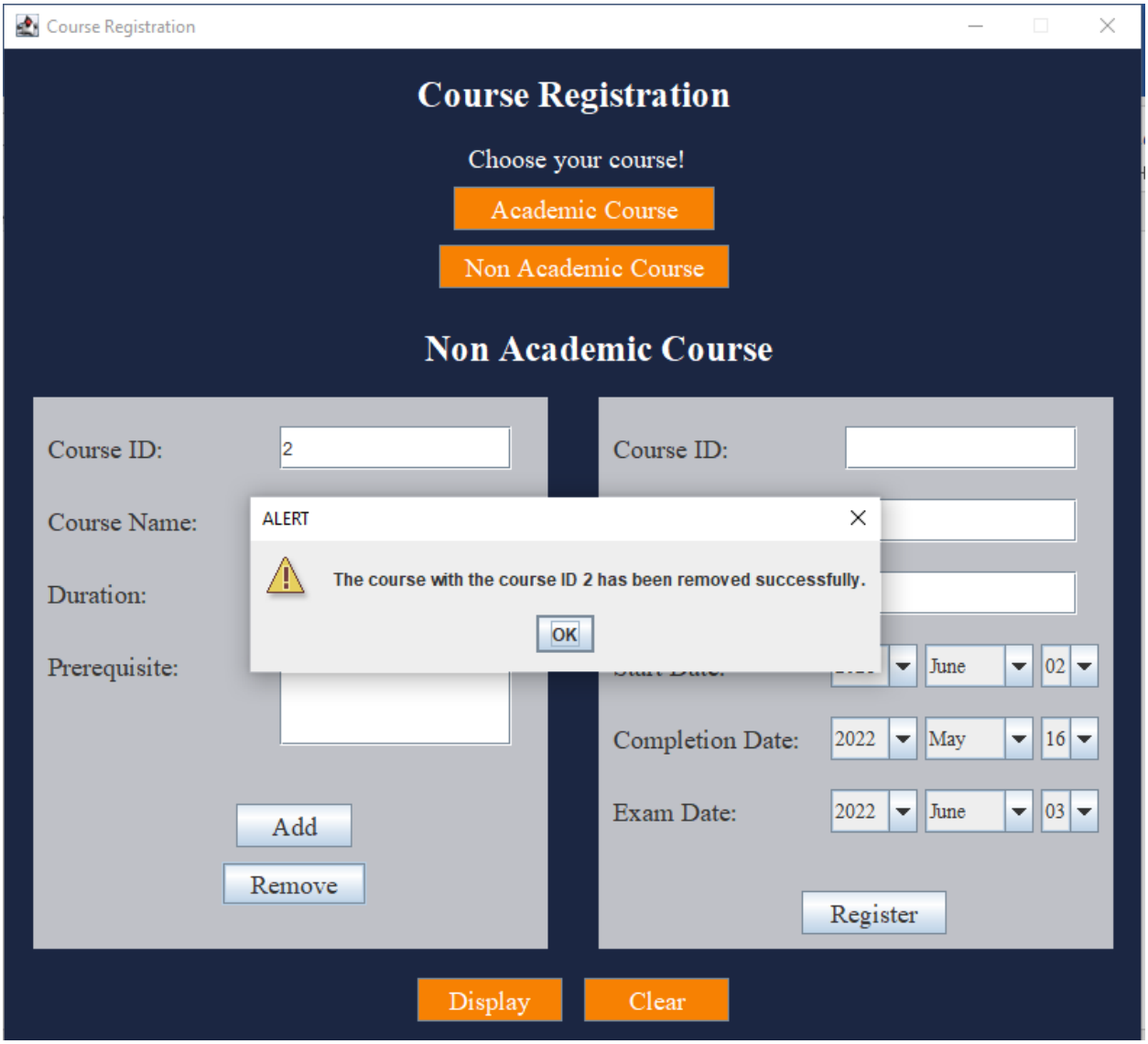

Figure 23 Removing a non academic course

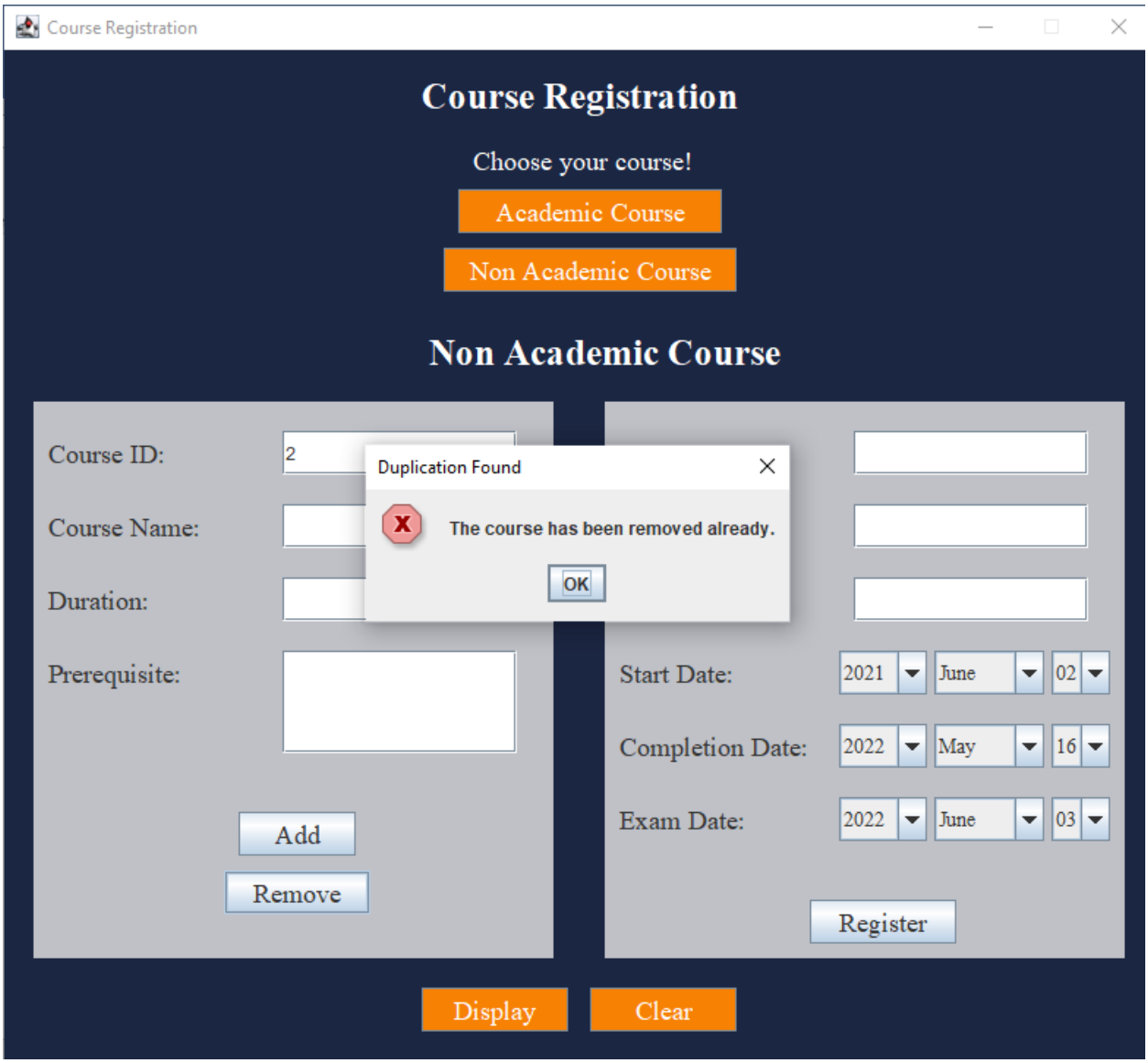

Figure 24 Removing a course which has been removed already

6. Errors

6.1 Syntax Error

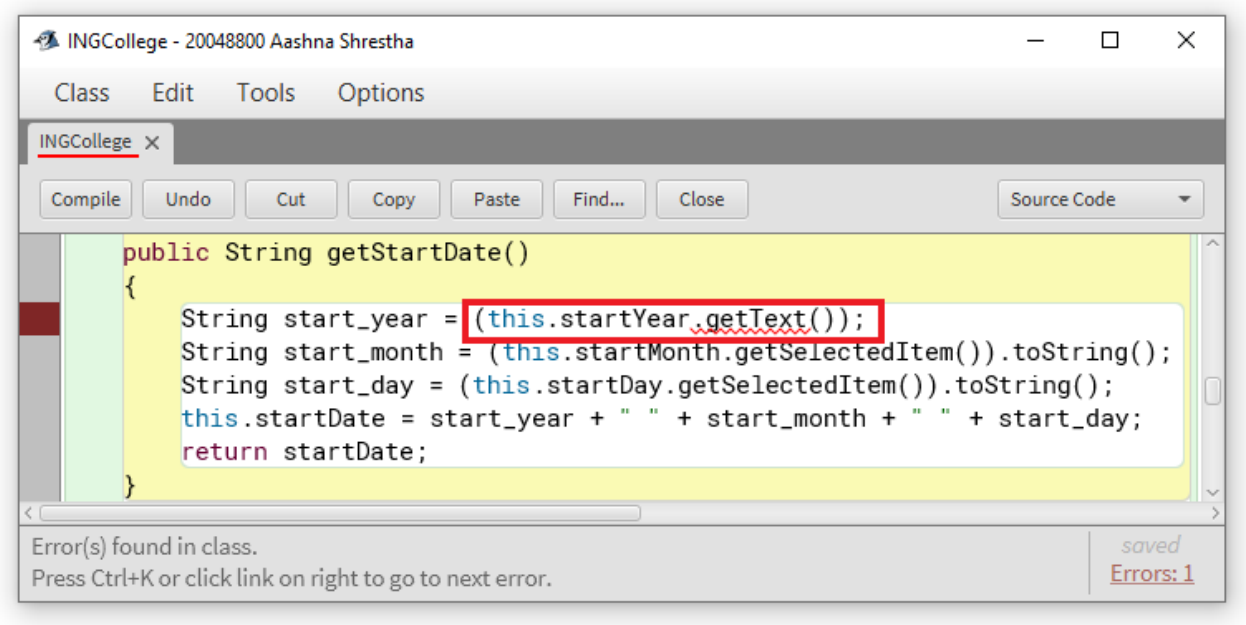

Figure 25 Syntax Error

Error: startYear is the reference variable for a JComboBox of year when a course starts. The year selected from a JComboBox cannot be called with the method getText().

Solution: The method getSelectedItem() should be called and the date should be converted into string with the method .toString().

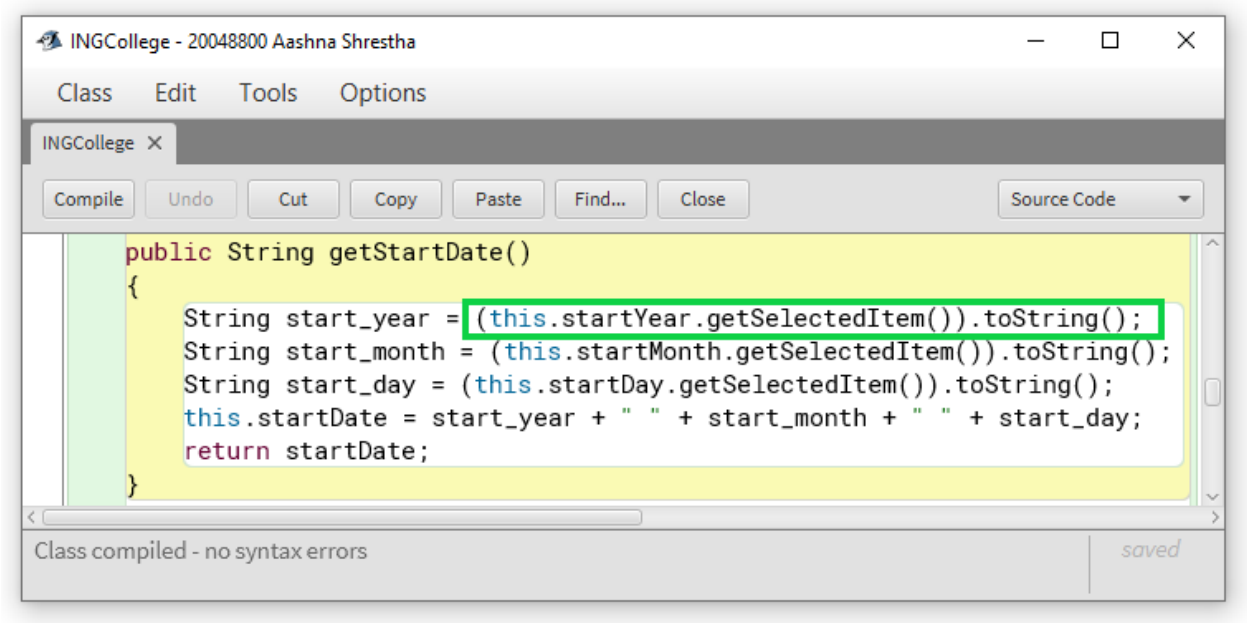

Figure 26 Correction of Syntax Error

16.2 Semantic Error

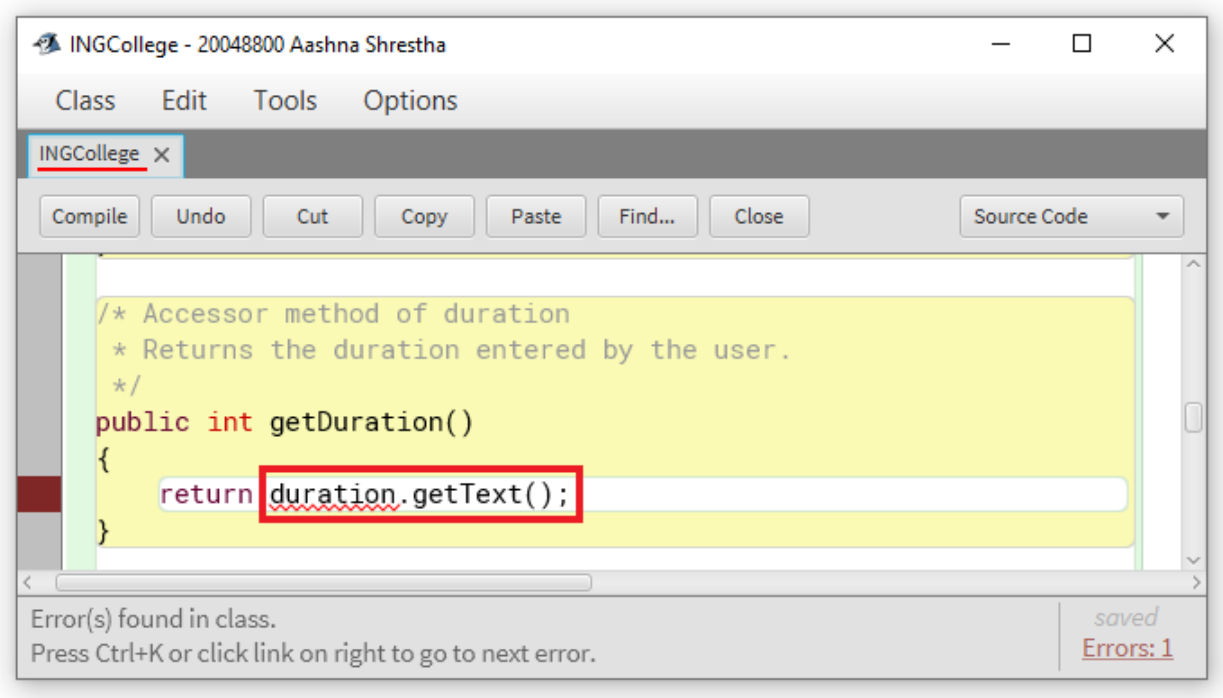

Figure 27 Semantic Error

Error: The syntax duration.getText() returns a string value entered by the user in a JTextField(); however the return type of the method getDuration() is int.

Solution: The duration entered by the user must be converted into integer before returning it in the method getDuration.

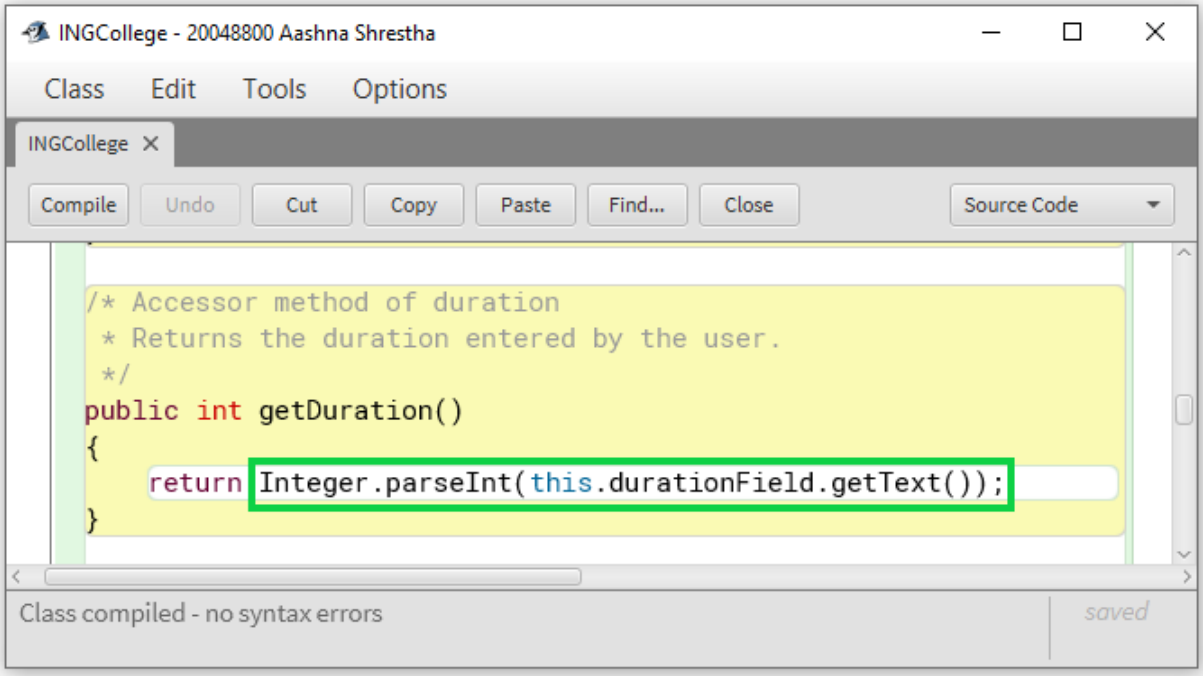

Figure 28 Correction of Semantic Error

6.2 Logic Error

```
else if(e.getSource() == registerAcademicButton){
    boolean courseFound = false;
    if(getRegisterId().isEmpty() || getCourseLeader().isEmpty() || getLecturer().isEmpty())
    {
        JOptionPane.showMessageDialog(frame, "Please fill all the fields");
    }
    else
    {
        for (Course course : courseList)
        {
            if (course.getcourseID().equals(getCourseId()) && course instanceof AcademicCourse){
                AcademicCourse academic_obj = (AcademicCourse) course;
                courseFound = true;
                if(academic_obj.getisRegistered() == true){
                    JOptionPane.showMessageDialog(frame, "The course has already been registered");
                }
                else{
                    academic_obj.register(getCourseLeader(), getLecturer(), getStartDate(), getCompletionDate());
                    JOptionPane.showMessageDialog(frame, "The course has been registered successfully");
                }
            }
        }
        if(!courseFound || courseList.size() == 0)
        {
            JOptionPane.showMessageDialog(frame, "The course has not been added yet");
        }
    }
```

Figure 29 Logic Error

Error: In order to register a course, the ArrayList containing the list of courses must be searched to check if the course ID entered by the user has been added. The method getCourseId() returns the ID entered by the user in the addCoursePanel instead of the registerPanel.

Solution: getRegisterId() must be compared to course.getcourseID().

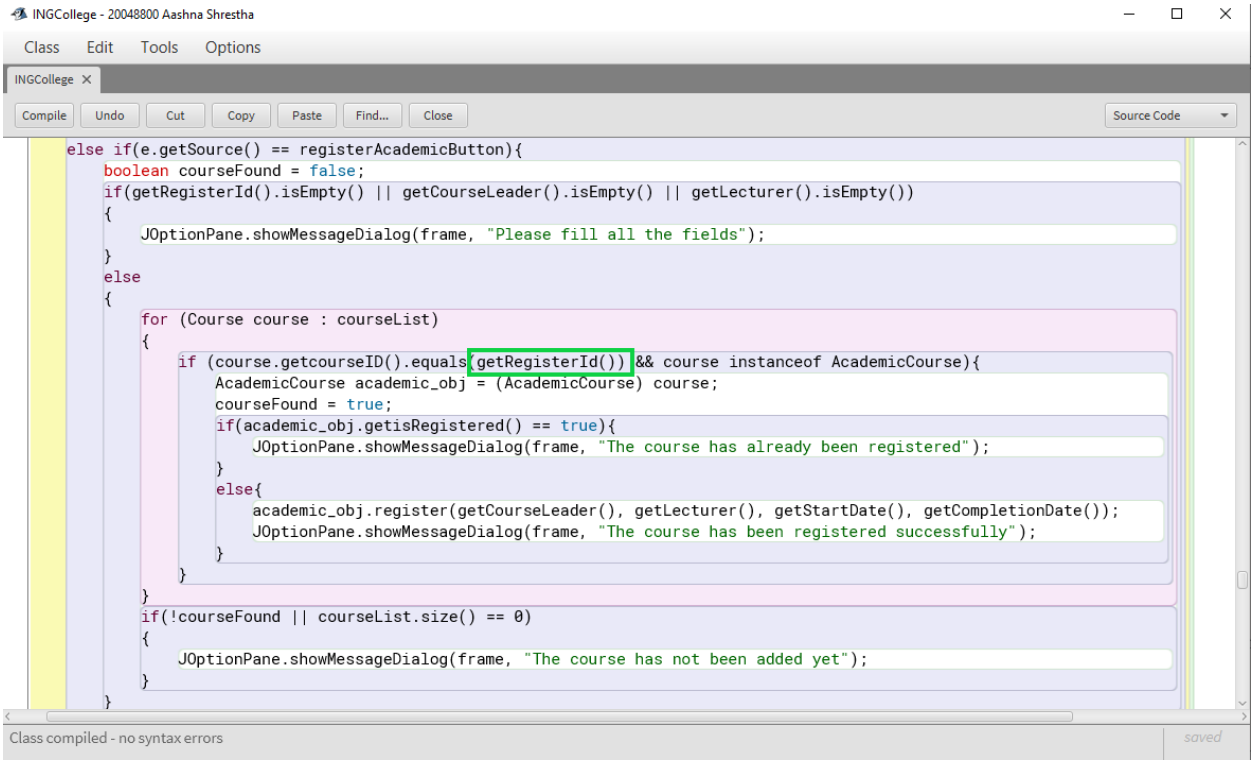

Figure 30 Correction of Logic Error

7. Conclusion

The report is based on a project to prepare a Graphical User Interface model. The GUI allows users to enter any academic or non-academic course details, add them to an array of courses, register the courses and remove them from the array. The program code for this project has been written in java. Methods to perform certain actions such as registering, removing or displaying a course were written in a previously compiled project. The methods have been called in the current program, when a user clicks the button for those specific tasks. The program also consists of codes to display dialog boxes when the tasks have been completed or if the tasks cannot be done.

Along with the program, a class diagram has been made. It consists a list of all the instance variables with their data types and methods with their return types. A pseudocode has been written for the complete program as well. It has been written in a simple language so that any user can understand how the code executes. The project also contains a method description table. The accessor modifiers of all the methods

have been mentioned in the table and the functions of each method have been explained.

After starting to work on the code, there were some obstructions due to frequent errors. The errors in the syntax could be fixed immediately as the program itself would not compile and direct the errors. However, some logical errors made the execution of the project complicated to some extent. For instance, the course could not be registered when the “Register” button was clicked even when the all the details were filled. After some failed attempts, it was found that the course ID from the panel to add courses was being compared to the ID from arraylist instead of the one from the panel to register courses. Brainstorming the ideas for some parts of the code was also a difficult yet interesting task. For example, toggling between the academic course panels and non-academic course panels seemed to be a tedious task. With some research and trials, a better way was figured - instead of creating separate panels for each part, the visibility of the components of each course could be changed as and when required.

Despite of the complications, the program has been tested and compiled successfully. The functions of various built-in methods were made clear while testing the program. The project taught that there can be various solutions to complete a certain task. It also taught to choose the most efficient method. Hence, the project has been an effective way to learn a real-life based problem and program it.

8. Bibliography

BlueJ, n.d. *BlueJ.* [Online]
Available at: https://www.bluej.org/
[Accessed 2 August 2021].
Geeks for Geeks, 2021. *NumberFormatException in Java with Examples.* [Online]
Available at: https://www.geeksforgeeks.org/numberformatexception-in-java-with-examples/
[Accessed 2 August 2021].
JavaTPoint, 2021. *javaTPoint.* [Online]
Available at: https://www.javatpoint.com/
[Accessed 3 August 2021].
Oracle, 2020. *Class NullPointerException.* [Online]
Available at:
https://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html
[Accessed 2 August 2021].

9. Appendix 1

9.1 INGCollege.java

```
/**
 * The class INGCollege contains a method courseGui() with JFrames, JPanels, JLabels, JTextFields, JButtons and JComboBox.
 * It contains accessor methods which return the values entered by the users in JTextFields.
 * It also contains actionPerformed method which executes certain course of actions such as adding, registering or removing a course
 * when particular buttons are clicked.
 *
 * @author (Aashna Shrestha)
 * @version (11.0.2)
 */
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;

public class INGCollege implements ActionListener
{
    //---Variable Declaration---//

    //JFrame, JPanel and title
    private JFrame frame;
    private JPanel mainPanel, addCoursePanel, registerPanel;
```

```
private JLabel title, academicTitle, nonAcademicTitle, courseOption;

//JButtons
private JButton academicButton, nonAcademicButton, addAcademicButton,
addNonAcademicButton, registerAcademicButton, registerNonAcademicButton,
removeButton,displayAcademicButton, displayNonAcademicButton, clearButton;

//JLabels in addCoursePanel
private JLabel courseIdLabel, courseNameLabel, durationLabel, levelLabel,
creditLabel,
assessmentLabel, prerequisiteLabel;

//JTextFields in addCoursePanel
private JTextField courseIdField, courseNameField, durationField, levelField,
creditField, assessmentField, prerequisiteField;

//JLabels in registerPanel
private JLabel idRegisterLabel, leaderLabel, lecturerLabel, instructorLabel,
startDateLabel, completionDateLabel, examDateLabel;

//JTextFields in registerPanel
private JTextField idRegisterField, leaderField, lecturerField, instructorField;

//JComboBox in registerPanel
private JComboBox startYear, startMonth, startDay, completionYear,
completionMonth, completionDay, examYear, examMonth, examDay;

//Variables to store date
private String start_year, start_month, start_day, startDate, completion_year,
completion_month, completion_day, completionDate,
exam_year, exam_month, exam_day, examDate;
```

```
//Static variabel for an object of INGCollege
static INGCollege ing;

//Creates an ArrayList
private ArrayList <Course> courseList = new ArrayList();

//Variable for objects of AcademicCourse class
private AcademicCourse academicObject;

//Variable for object of NonAcademicCourse class
private NonAcademicCourse nonAcademicObject;

/* Method to create a GUI
 * The method contains objects of JFrame, Jpanel, JLabel, JTextField, JComboBox
 * along with their bounds, font and background/foreground color
   */
public void courseGui()
{
    //Creating object for the frame
    frame = new JFrame("Course Registration");

    //Creating objects for colors
    Color mainBackground = new Color(27, 38, 66);
    Color innerPanel = new Color(191, 193, 199);
    Color buttonColor = new Color(247, 129, 2);

    //Creating object for the main panel
    mainPanel = new JPanel();
    mainPanel.setLayout(null);
    mainPanel.setBackground(mainBackground);
```

```
//Creating objects for fonts
Font titleFont = new Font("Serif", Font.BOLD, 25);
Font mainFont = new Font("Serif", Font.PLAIN, 18);
Font comboBoxFont = new Font("Serif", Font.PLAIN, 14);

//JLabel for the title of the frame
title = new JLabel("Course Registration");
title.setBounds(285, 5, 250, 50);
title.setFont(titleFont);
title.setForeground(Color.WHITE);
mainPanel.add(title);

//JLabel which asks user to choose a course
courseOption = new JLabel("Choose your course!");
courseOption.setFont(mainFont);
courseOption.setForeground(Color.WHITE);
courseOption.setBounds(320, 60, 300, 30);
mainPanel.add(courseOption);

//JButton to choose Academic Course
academicButton = new JButton("Academic Course");
academicButton.setBounds(310, 95, 180, 30);
academicButton.setFont(mainFont);
academicButton.setForeground(Color.WHITE);
academicButton.setBackground(buttonColor);
academicButton.addActionListener(this);
mainPanel.add(academicButton);

//JButton to choose Non Academic Course
nonAcademicButton = new JButton("Non Academic Course");
```

```
nonAcademicButton.setBounds(300, 135, 200, 30);
nonAcademicButton.setFont(mainFont);
nonAcademicButton.setForeground(Color.WHITE);
nonAcademicButton.setBackground(buttonColor);
nonAcademicButton.addActionListener(this);
mainPanel.add(nonAcademicButton);

//JLabel for the title of Academic Course
academicTitle = new JLabel("Academic Course");
academicTitle.setBounds(310, 180, 250, 50);
academicTitle.setFont(titleFont);
academicTitle.setForeground(Color.WHITE);
mainPanel.add(academicTitle);

//JLabel for the title of Non Academic Course
nonAcademicTitle = new JLabel("Non Academic Course");
nonAcademicTitle.setBounds(290, 180, 250, 50);
nonAcademicTitle.setFont(titleFont);
nonAcademicTitle.setForeground(Color.WHITE);
mainPanel.add(nonAcademicTitle);
nonAcademicTitle.setVisible(false);

//JPanel for adding course
addCoursePanel = new JPanel();
addCoursePanel.setBackground(innerPanel);
addCoursePanel.setBounds(20, 240, 355, 380);
addCoursePanel.setVisible(true);
addCoursePanel.setLayout(null);
mainPanel.add(addCoursePanel);

//JPanel for registering course
```

```
registerPanel = new JPanel();
registerPanel.setBackground(innerPanel);
registerPanel.setBounds(410, 240, 355, 380);
registerPanel.setVisible(true);
registerPanel.setLayout(null);
mainPanel.add(registerPanel);

//---Components of addCoursePanel---//

//JLabel for course ID in addCoursePanel
courseIdLabel = new JLabel("Course ID: ");
courseIdLabel.setBounds(10, 20, 100, 30);
courseIdLabel.setFont(mainFont);
addCoursePanel.add(courseIdLabel);

//JTextField for course ID in addCoursePanel
courseIdField = new JTextField();
courseIdField.setBounds(170, 20, 160, 30);
addCoursePanel.add(courseIdField);

//JLabel for course name
courseNameLabel = new JLabel("Course Name: ");
courseNameLabel.setBounds(10, 70, 110, 30);
courseNameLabel.setFont(mainFont);
addCoursePanel.add(courseNameLabel);

//JTextField for course name
courseNameField = new JTextField();
courseNameField.setBounds(170, 70, 160, 30);
addCoursePanel.add(courseNameField);
```

```
//JLabel for duration
durationLabel = new JLabel("Duration: ");
durationLabel.setBounds(10, 120, 100, 30);
durationLabel.setFont(mainFont);
addCoursePanel.add(durationLabel);

//JTextField for duration
durationField = new JTextField();
durationField.setBounds(170, 120, 160, 30);
addCoursePanel.add(durationField);

//JLabel for level
levelLabel = new JLabel("Level: ");
levelLabel.setBounds(10, 170, 100, 30);
levelLabel.setFont(mainFont);
addCoursePanel.add(levelLabel);

//JTextField for level
levelField = new JTextField();
levelField.setBounds(170, 170, 160, 30);
addCoursePanel.add(levelField);

//JLabel for credit
creditLabel = new JLabel("Credit: ");
creditLabel.setBounds(10, 220, 100, 30);
creditLabel.setFont(mainFont);
addCoursePanel.add(creditLabel);

//JTextField for credit
creditField = new JTextField();
creditField.setBounds(170, 220, 160, 30);
```

```
addCoursePanel.add(creditField);

//JLabel for number of assessments
assessmentLabel = new JLabel("No. of Assessments: ");
assessmentLabel.setBounds(10, 270, 160, 30);
assessmentLabel.setFont(mainFont);
addCoursePanel.add(assessmentLabel);

//JTextField for number of assesments
assessmentField = new JTextField();
assessmentField.setBounds(170, 270, 160, 30);
addCoursePanel.add(assessmentField);

//JLabel for prerequisite
prerequisiteLabel = new JLabel("Prerequisite: ");
prerequisiteLabel.setBounds(10, 170, 100, 30);
prerequisiteLabel.setFont(mainFont);
addCoursePanel.add(prerequisiteLabel);
prerequisiteLabel.setVisible(false);

//JTextField for prerequisite
prerequisiteField = new JTextField();
prerequisiteField.setBounds(170, 170, 160, 70);
addCoursePanel.add(prerequisiteField);
prerequisiteField.setVisible(false);

//JButton for adding an academic course
addAcademicButton = new JButton("Add");
addAcademicButton.setBounds(140, 340, 80, 30);
addAcademicButton.setFont(mainFont);
addAcademicButton.addActionListener(this);
```

```
addCoursePanel.add(addAcademicButton);

//JButton for adding a non academic course
addNonAcademicButton = new JButton("Add");
addNonAcademicButton.setBounds(140, 280, 80, 30);
addNonAcademicButton.setFont(mainFont);
addNonAcademicButton.addActionListener(this);
addCoursePanel.add(addNonAcademicButton);
addNonAcademicButton.setVisible(false);

//JButton to remove non academic course
removeButton = new JButton("Remove");
removeButton.setBounds(130, 320, 100, 30);
removeButton.setFont(mainFont);
removeButton.addActionListener(this);
addCoursePanel.add(removeButton);
removeButton.setVisible(false);

//---Components of registerPanel---//

//JLabel for course Id in registerPanel
idRegisterLabel = new JLabel("Course ID: ");
idRegisterLabel.setBounds(10, 20, 100, 30);
idRegisterLabel.setFont(mainFont);
registerPanel.add(idRegisterLabel);

//JTextField for course Id in registerPanel
idRegisterField = new JTextField();
idRegisterField.setBounds(170, 20, 160, 30);
registerPanel.add(idRegisterField);
```

```
//JLabel for course leader
leaderLabel = new JLabel("Course Leader: ");
leaderLabel.setBounds(10, 70, 120, 30);
leaderLabel.setFont(mainFont);
registerPanel.add(leaderLabel);

//JTextField for course leader
leaderField = new JTextField();
leaderField.setBounds(170, 70, 160, 30);
registerPanel.add(leaderField);

//JLabel for lecturer
lecturerLabel = new JLabel("Lecturer: ");
lecturerLabel.setBounds(10, 120, 100, 30);
lecturerLabel.setFont(mainFont);
registerPanel.add(lecturerLabel);

//JTextField for lecturer
lecturerField = new JTextField();
lecturerField.setBounds(170, 120, 160, 30);
registerPanel.add(lecturerField);

//JLabel for lecturer
instructorLabel = new JLabel("Instructor: ");
instructorLabel.setBounds(10, 120, 100, 30);
instructorLabel.setFont(mainFont);
registerPanel.add(instructorLabel);
instructorLabel.setVisible(false);

//JTextField for lecturer
instructorField = new JTextField();
```

```
instructorField.setBounds(170, 120, 160, 30);
registerPanel.add(instructorField);
instructorField.setVisible(false);

//JLabel for the date in which a course starts
startDateLabel = new JLabel("Start Date: ");
startDateLabel.setBounds(10, 170, 150, 30);
startDateLabel.setFont(mainFont);
registerPanel.add(startDateLabel);

//yearList is an array which stores the year
Integer yearList[] = new Integer[28];
int year = 2020;
for (int i = 0; i <= 26; i++){
    yearList[i] = year;
    year++;
}

//JComboBox for the year in which a course starts
startYear = new JComboBox(yearList);
startYear.setBounds(160, 170, 60, 30);
startYear.setFont(comboBoxFont);
registerPanel.add(startYear);

//month is an array which stores the month
String[] month = {"January", "February", "March", "April", "May", "June", "July",
"August", "September", "October", "November", "December"};

//JComboBox for the month in which a course starts
startMonth = new JComboBox(month);
startMonth.setBounds(225, 170, 75, 30);
```

```
startMonth.setFont(comboBoxFont);
registerPanel.add(startMonth);

//dayList is an array which stores the date
String dayList[] = new String[31];
int day = 1;
for (int i = 0; i <= 30; i++){
    if (day < 10){
        dayList[i] = "0" + day;
    }
    else{
        dayList[i] = String.valueOf(day);
    }
    day++;
}

//JComboBox for the date in which a course starts
startDay = new JComboBox(dayList);
startDay.setBounds(305, 170, 40, 30);
startDay.setFont(comboBoxFont);
registerPanel.add(startDay);

//JLabel for the date when a course completes
completionDateLabel = new JLabel("Completion Date: ");
completionDateLabel.setBounds(10, 220, 140, 30);
completionDateLabel.setFont(mainFont);
registerPanel.add(completionDateLabel);

//JComboBox for the year in which a course completes
completionYear = new JComboBox(yearList);
completionYear.setBounds(160, 220, 60, 30);
```

```
completionYear.setFont(comboBoxFont);
registerPanel.add(completionYear);

//JComboBox for the month in which a course completes
completionMonth = new JComboBox(month);
completionMonth.setBounds(225, 220, 75, 30);
completionMonth.setFont(comboBoxFont);
registerPanel.add(completionMonth);

//JComboBox for the date in which a course completes
completionDay = new JComboBox(dayList);
completionDay.setBounds(305, 220, 40, 30);
completionDay.setFont(comboBoxFont);
registerPanel.add(completionDay);

//JLabel for the date of exam
examDateLabel = new JLabel("Exam Date:");
examDateLabel.setBounds(10, 270, 160, 30);
examDateLabel.setFont(mainFont);
registerPanel.add(examDateLabel);
examDateLabel.setVisible(false);

//JComboBox for the year of exam
examYear = new JComboBox(yearList);
examYear.setBounds(160, 270, 60, 30);
examYear.setFont(comboBoxFont);
registerPanel.add(examYear);
examYear.setVisible(false);

//JComboBox for the month of exam
examMonth = new JComboBox(month);
```

```
examMonth.setBounds(225, 270, 75, 30);
examMonth.setFont(comboBoxFont);
registerPanel.add(examMonth);
examMonth.setVisible(false);

//JComboBox for the day of exam
examDay = new JComboBox(dayList);
examDay.setBounds(305, 270, 40, 30);
examDay.setFont(comboBoxFont);
registerPanel.add(examDay);
examDay.setVisible(false);

//JButton to register academic course
registerAcademicButton = new JButton("Register");
registerAcademicButton.setBounds(140, 320, 100, 30);
registerAcademicButton.setFont(mainFont);
registerAcademicButton.addActionListener(this);
registerPanel.add(registerAcademicButton);

//JButton to register non academic course
registerNonAcademicButton = new JButton("Register");
registerNonAcademicButton.setBounds(140, 340, 100, 30);
registerNonAcademicButton.setFont(mainFont);
registerNonAcademicButton.addActionListener(this);
registerPanel.add(registerNonAcademicButton);
registerNonAcademicButton.setVisible(false);

//JButton to display information of Academic Course
displayAcademicButton = new JButton("Display");
displayAcademicButton.setBounds(285, 640, 100, 30);
displayAcademicButton.setFont(mainFont);
```

```
        displayAcademicButton.setForeground(Color.WHITE);
        displayAcademicButton.setBackground(buttonColor);
        displayAcademicButton.addActionListener(this);
        mainPanel.add(displayAcademicButton);

        //JButton to display the information of Non Academic Course
        displayNonAcademicButton = new JButton("Display");
        displayNonAcademicButton.setBounds(285, 640, 100, 30);
        displayNonAcademicButton.setFont(mainFont);
        displayNonAcademicButton.setForeground(Color.WHITE);
        displayNonAcademicButton.setBackground(buttonColor);
        displayNonAcademicButton.addActionListener(this);
        mainPanel.add(displayNonAcademicButton);
        displayNonAcademicButton.setVisible(false);

        //JButton to clear all the text fields
        clearButton = new JButton("Clear");
        clearButton.setBounds(400, 640, 100, 30);
        clearButton.setFont(mainFont);
        clearButton.setForeground(Color.WHITE);
        clearButton.setBackground(buttonColor);
        clearButton.addActionListener(this);
        mainPanel.add(clearButton);

        frame.add(mainPanel);
        frame.setBounds(300, 0, 800, 730);
        frame.setVisible(true);
        frame.setResizable(false);
    }

    /* Accessor method of courseId.
```

```
 * Returns the course ID entered by the user.
 */
public String getCourseId()
{
    return this.courseIdField.getText();
}

/*Accessor method of courseName
 * Returns the course name entered by the user.
 */
public String getCourseName()
{
    return this.courseNameField.getText();
}

/* Accessor method of duration
 * Returns the duration entered by the user.
 */
public int getDuration()
{
    return Integer.parseInt(this.durationField.getText());
}

/* Accessor method of level
 * Returns the level entered by the user.
 */
public String getLevel()
{
    return this.levelField.getText();
}
```

```
/* Accessor method of credit
 * Returns the credit entered by the user.
 */
public String getCredit()
{
    return this.creditField.getText();
}

/* Accessor method of noOfAssessments
 * Returns the number of assessments entered by the user.
 */
public int getNoOfAssessments()
{
    return Integer.parseInt(this.assessmentField.getText());
}

/* Accessor method of prerequisite
 * Returns the prerequisite entered by the user.
 */
public String getPrerequisite()
{
    return this.prerequisiteField.getText();
}

/* Accessor method of Id
 * Returns the course ID entered by the user while registering a course.
 */
public String getRegisterId()
{
    return this.idRegisterField.getText();
}
```

```
/* Accessor method of course leader
 * Returns the course leader entered by the user.
 */
public String getCourseLeader()
{
    return this.leaderField.getText();
}

/* Accessor method of lecturer
 * Returns the name of the lecturer entered by the user.
 */
public String getLecturer()
{
    return this.lecturerField.getText();
}

/* Accessor method of instructor
 * Returns the name of the instructor entered by the user.
 */
public String getInstructor()
{
    return this.instructorField.getText();
}

/* Accessor method of starting date
 * Concatenates the year, month and date entered by the user.
 * Returns the date on which the course starts.
 */
public String getStartDate()
{
```

```
        String start_year = (this.startYear.getSelectedItem()).toString();
        String start_month = (this.startMonth.getSelectedItem()).toString();
        String start_day = (this.startDay.getSelectedItem()).toString();
        this.startDate = start_year + " " + start_month + " " + start_day;
        return startDate;
    }

    /* Accessor method of completion date
     * Concatenates the year, month and date entered by the user.
     * Returns the date on which the course completes.
     */
    public String getCompletionDate()
    {
        String completion_year = (this.completionYear.getSelectedItem()).toString();
        String completion_month = (this.completionMonth.getSelectedItem()).toString();
        String completion_day = (this.completionDay.getSelectedItem()).toString();
       this.completionDate = completion_year + " " + completion_month + " " +
completion_day;
        return completionDate;
    }

    /* Accessor method of exam date
     * Concatenates the year, month and date entered by the user.
     * Returns the date of exam.
     */
    public String getExamDate()
    {
        String exam_year = (this.examYear.getSelectedItem()).toString();
        String exam_month = (this.examMonth.getSelectedItem()).toString();
        String exam_day = (this.examDay.getSelectedItem()).toString();
        this.examDate = exam_year + " " + exam_month + " " + exam_day;
```

```
        return examDate;
    }

    /* ActionPerformed method checks which button has been clicked by the user
     * Certain course of action will execute according to the required button
     */
    public void actionPerformed(ActionEvent e)
    {
        /*Actions to be performed when nonAcademicButton is clicked
        * Visibility of JLabel, JField, JButton are changed to display the fields needed for
Non Academic Course
         */
        if (e.getSource() == nonAcademicButton){
            //Changes to be made in addCoursePanel
            levelLabel.setVisible(false);
            creditLabel.setVisible(false);
            assessmentLabel.setVisible(false);
            levelField.setVisible(false);
            creditField.setVisible(false);
            lecturerLabel.setVisible(false);
            lecturerField.setVisible(false);
            assessmentField.setVisible(false);
            prerequisiteLabel.setVisible(true);
            prerequisiteField.setVisible(true);
            addAcademicButton.setVisible(false);
            addNonAcademicButton.setVisible(true);
            removeButton.setVisible(true);

            //Changes to be made in registerPanel
            examDateLabel.setVisible(true);
            examYear.setVisible(true);
```

```
        examMonth.setVisible(true);
        examDay.setVisible(true);
        registerNonAcademicButton.setVisible(true);
        registerAcademicButton.setVisible(false);
        instructorLabel.setVisible(true);
        instructorField.setVisible(true);

        //Changes to be made in mainPanel
        nonAcademicTitle.setVisible(true);
        academicTitle.setVisible(false);
        displayAcademicButton.setVisible(false);
        displayNonAcademicButton.setVisible(true);
    }

    /* Actions to be performed when academicButton is clicked
     * Visibility of JLabel, JField, JButton are changed to display only the required fields
     */
    else if(e.getSource() == academicButton){
        //Changes to be made in addCoursePanel
        levelLabel.setVisible(true);
        creditLabel.setVisible(true);
        assessmentLabel.setVisible(true);
        levelField.setVisible(true);
        creditField.setVisible(true);
        assessmentField.setVisible(true);
        lecturerLabel.setVisible(true);
        lecturerField.setVisible(true);
        prerequisiteLabel.setVisible(false);
        prerequisiteField.setVisible(false);
        addNonAcademicButton.setVisible(false);
        addAcademicButton.setVisible(true);
```

```
            registerNonAcademicButton.setVisible(false);
            registerAcademicButton.setVisible(true);
            removeButton.setVisible(false);

            //Changes to be made in registerPanel
            examDateLabel.setVisible(false);
            examYear.setVisible(false);
            examMonth.setVisible(false);
            examDay.setVisible(false);
            instructorLabel.setVisible(false);
            instructorField.setVisible(false);

            //Changes to be made in mainPanel
            nonAcademicTitle.setVisible(false);
            academicTitle.setVisible(true);
            displayAcademicButton.setVisible(true);
            displayNonAcademicButton.setVisible(false);
        }

        /* Actions to be performed when addAcademicButton is clicked.
         * Checks if all the fields have been filled.
         * An object of AcademicCourse class is created.
         * The course is added to the ArrayList: courseList.
         */
        else if(e.getSource() == addAcademicButton){
            boolean courseFound = false;
            for(Course course : courseList)
            {
              if (course.getcourseID().equals(getCourseId()) && course instanceof
AcademicCourse)
               {
```

```
                JOptionPane.showMessageDialog(frame, "The course has been added
already.", "Duplication Found", JOptionPane.ERROR_MESSAGE);
                courseFound = true;
            }
        }
        if (!courseFound || courseList.size() == 0)
        {
            try{
                if(getCourseId().isEmpty() || getCourseName().isEmpty() ||
getLevel().isEmpty() || getCredit().isEmpty())
                {
                JOptionPane.showMessageDialog(frame, "Please fill all the fields.",
"WARNING", JOptionPane.ERROR_MESSAGE);
                }
                else
                    {
                    academicObject = new AcademicCourse(getCourseId(),
getCourseName(), getDuration(), getLevel(), getCredit(),
                    getNoOfAssessments());
                    courseList.add(academicObject);
                    JOptionPane.showMessageDialog(frame, "Academic Course has been
added successfully with the following details: \n Course ID: "
                    + getCourseId() +"\n Course Name: " + getCourseName() + "\n Duration:
" + getDuration() + "\n Level: " + getLevel() +
                    "\n Credit: " + getCredit() + "\n Number of Assessments: " +
getNoOfAssessments(), "ALERT", JOptionPane.WARNING_MESSAGE);
                    }
            }
            catch(NumberFormatException ae)
            {
```

```
                JOptionPane.showMessageDialog(frame, "Enter a numerical value in duration and number of assessments", "ERROR", JOptionPane.ERROR_MESSAGE);
            }
            catch(NullPointerException ex)
            {
                JOptionPane.showMessageDialog(frame, "Please fill all the fields.", "ERROR", JOptionPane.ERROR_MESSAGE);
            }
        }
    }

    /* Actions to be performed when registerAcademicButton is clicked.
     * Checks if the entered course ID is present in the ArrayList.
     * If found, it checks if the course has been registered.
     * If the course has not been registered, the register method from the AcademicCourse class will be called.
     */

    else if(e.getSource() == registerAcademicButton){
        boolean courseFound = false;
        if(getRegisterId().isEmpty() || getCourseLeader().isEmpty() || getLecturer().isEmpty())
        {
            JOptionPane.showMessageDialog(frame, "Please fill all the fields", "WARNING", JOptionPane.ERROR_MESSAGE);
        }
        else
        {
            for (Course course : courseList)
            {
```

```
            if (course.getcourseID().equals(getRegisterId()) && course instanceof AcademicCourse){
                AcademicCourse academic_obj = (AcademicCourse) course;
                courseFound = true;
                if(academic_obj.getisRegistered() == true){
                 JOptionPane.showMessageDialog(frame, "The course has been registered already", "Duplication Found", JOptionPane.ERROR_MESSAGE);
                }
                else{
                 academic_obj.register(getCourseLeader(), getLecturer(), getStartDate(), getCompletionDate());
                 JOptionPane.showMessageDialog(frame, "The course has been registered successfully with the following details: \n Course ID:"
                 + getRegisterId() + "\n Course Leader: " + getCourseLeader() + "\n Lecturer: " + getLecturer() + "\n StartDate: "
                 + getStartDate() + "\n Completion Date: " + getCompletionDate(), "ALERT", JOptionPane.WARNING_MESSAGE);
                }
            }
        }
        if(!courseFound || courseList.size() == 0)
        {
          JOptionPane.showMessageDialog(frame, "The course has not been added yet", "WARNING", JOptionPane.ERROR_MESSAGE);
        }
      }
    }

    /* Actions to be performed when displayAcademicButton is clicked.
     * The courseList is searched for the details of all the academic courses.
     * The display method is called from the AcademicCourse class to print the details.
```

```
 */
else if(e.getSource() == displayAcademicButton){
    boolean courseFound = false;
    for (Course course : courseList)
    {
        if(course instanceof AcademicCourse)
        {
            AcademicCourse academic_obj = (AcademicCourse) course;
            academic_obj.display();
            courseFound = true;
        }
    }
    if(!courseFound || courseList.size() == 0)
    {
      JOptionPane.showMessageDialog(frame, "Academic Course has not been
added yet.", "WARNING", JOptionPane.ERROR_MESSAGE);
    }
}

/* Actions to be performed when addNonAcademicButton is clicked.
 * Checks if all the fields have been filled.
 * An object of NonAcademicCourse class is created.
 * The course is added to the ArrayList: courseList.
 */
else if(e.getSource() == addNonAcademicButton){
    boolean courseFound = false;
    for(Course course : courseList)
    {
      if (course.getcourseID().equals(getCourseId()) && course instanceof
NonAcademicCourse)
        {
```

```
                JOptionPane.showMessageDialog(frame, "The course has been added
already.", "Duplication Found", JOptionPane.ERROR_MESSAGE);
                courseFound = true;
            }
        }
        if (!courseFound || courseList.size() == 0)
        {
            try
            {
                if(getCourseId().isEmpty() || getCourseName().isEmpty() ||
getPrerequisite().isEmpty()){
                    JOptionPane.showMessageDialog(frame, "Please fill all the fields.",
"WARNING", JOptionPane.ERROR_MESSAGE);
                }
                else
                {
                    nonAcademicObject = new NonAcademicCourse(getCourseId(),
getCourseName(), getDuration(), getPrerequisite());
                        courseList.add(nonAcademicObject);
                    JOptionPane.showMessageDialog(frame, "Non Academic Course has
been added successfully with the following details: \n Course ID: "
                    + getCourseId() +"\n Course Name: " + getCourseName() + "\n Duration:
" + getDuration() + "\n Prerequisite: "
                        + getPrerequisite(), "ALERT", JOptionPane.WARNING_MESSAGE);
                }
            }
            catch(NumberFormatException ae)
            {
                JOptionPane.showMessageDialog(frame, "Please enter a numerical value
in duration", "WARNING", JOptionPane.ERROR_MESSAGE);
            }
```

```
            catch(NullPointerException ex)
            {
              JOptionPane.showMessageDialog(frame, "Please fill all the fields.", "WARNING", JOptionPane.ERROR_MESSAGE);
            }
        }
    }

    /* Actions to be performed when registerNonAcademicButton is clicked.
     * Checks if the entered course ID is present in the ArrayList.
     * If found, it checks if the course has been registered.
    * If the course has not been registered, the register method from the NonAcademicCourse class will be called.
     */
    else if(e.getSource() == registerNonAcademicButton){
        boolean courseFound = false;
       if(getRegisterId().isEmpty() || getCourseLeader().isEmpty() || getInstructor().isEmpty())
        {
          JOptionPane.showMessageDialog(frame, "Please fill all the fields", "WARNING", JOptionPane.ERROR_MESSAGE);
        }
        else
        {
            for(Course course : courseList)
            {
              if (course.getcourseID().equals(getRegisterId()) && course instanceof NonAcademicCourse){
                  NonAcademicCourse nonAcademic_obj = (NonAcademicCourse) course;
                    courseFound = true;
```

```
                    if(nonAcademic_obj.getisRegistered() == true){
                     JOptionPane.showMessageDialog(frame, "The course has been registered already.", "Duplication Found", JOptionPane.ERROR_MESSAGE);
                    }
                    else{
                     nonAcademic_obj.register(getCourseLeader(), getInstructor(), getStartDate(),getCompletionDate(),getExamDate());
                     JOptionPane.showMessageDialog(frame, "The course has been registered successfully with the following details: \n Course ID:"
                     + getRegisterId() + "\n Course Leader: " + getCourseLeader() + "\n Lecturer: " + getInstructor() + "\n StartDate: "
                     + getStartDate() + "\n Completion Date: " + getCompletionDate() + "\n Exam Date: " + getExamDate(),
                       "ALERT", JOptionPane.WARNING_MESSAGE);
                       break;
                    }
                 }
              }
              if (!courseFound || courseList.size() == 0)
              {
                JOptionPane.showMessageDialog(frame, "The course has not been added yet", "WARNING", JOptionPane.ERROR_MESSAGE);
              }
           }

        }

        /* Actions to be performed when displayNonAcademicButton is clicked.
         * The courseList if searched for the details of all the non academic courses.
        * The display method is called from the NonAcademicCourse class to print the details.
```

```
 */
else if(e.getSource() == displayNonAcademicButton)
{
    boolean courseFound = false;
    for (Course course: courseList)
    {
        if(course instanceof NonAcademicCourse)
        {
            NonAcademicCourse nonAcademic_obj = (NonAcademicCourse) course;
            nonAcademic_obj.display();
            courseFound = true;
        }
    }
    if(!courseFound || courseList.size() == 0)
    {
      JOptionPane.showMessageDialog(frame, "Non Academic Course has not
been added yet.", "WARNING", JOptionPane.ERROR_MESSAGE);
    }
}

/* Actions to be performed when removeButton is clicked.
 * Checks if the entered course ID is present in the arraylist.
 * If found, the remove method from NonAcademicCourse class is called.
 */
else if(e.getSource() == removeButton)
{
    if (getCourseId().isEmpty())
    {
      JOptionPane.showMessageDialog(frame, "Please enter the Course ID",
"WARNING", JOptionPane.ERROR_MESSAGE);
    }
```

```
        else
        {
            boolean courseFound = false;
            for(Course course : courseList)
            {
              if(course.getcourseID().equals(getCourseId())  &&  course  instanceof
NonAcademicCourse)
                {
                    courseFound = true;
                  NonAcademicCourse  nonAcademic_obj  =  (NonAcademicCourse)
course;
                    if (nonAcademic_obj.getisRemoved() == true)
                    {
                     JOptionPane.showMessageDialog(frame,  "The  course  has  been
removed already.", "Duplication Found",
                        JOptionPane.ERROR_MESSAGE);
                    }
                    else
                    {
                        nonAcademic_obj.remove();
                     JOptionPane.showMessageDialog(frame, "The course with the course
ID " + getCourseId() + " has been removed successfully."
                        , "ALERT", JOptionPane.WARNING_MESSAGE);
                        break;
                    }

                }
            }
            if(!courseFound || courseList.size() == 0)
            {
```

```
                JOptionPane.showMessageDialog(frame, "The course has not been added
yet", "WARNING", JOptionPane.ERROR_MESSAGE);
            }
        }
    }

    /* Actions to be performed when clearButton is clicked
     * The text of all the JTextFields are set to and empty string("") to clear all the fields
     */
    else if(e.getSource() == clearButton){
        courseIdField.setText("");
        courseNameField.setText("");
        durationField.setText("");
        levelField.setText("");
        creditField.setText("");
        assessmentField.setText("");
        prerequisiteField.setText("");
        idRegisterField.setText("");
        leaderField.setText("");
        lecturerField.setText("");
        instructorField.setText("");
    }
}

/* Main method of the program
 * Calls the courseGui() method.
 */
public static void main(String[] args){
    ing = new INGCollege();
    ing.courseGui();
}
```

```
}
```

10. Appendix 2

10.1 Course.java

```
/**
```

```
 * The Course class consists of four attributes i.e., courseID, courseName,
courseLeader and duration.
 * The class contains a constructor where the variables are initialized, accessor and
mutator methods,
 * and a display method to output the details of a course.
 *
 * @author (Aashna Shrestha)
 * @version (11.0.2)
 */

public class Course
{
    //Declares the instance variables
    private String courseID;
    private String courseName;
    private String courseLeader;
    private int duration;

    /* Constructor with 3 parameters: courseID, courseName, duration
     * Initialize the attributes */
    Course(String courseID, String courseName, int duration)
    {
        this.courseID = courseID;
        this.courseName = courseName;
        this.duration = duration;
        this.courseLeader = "";
    }
    //Accessor method for courseID
    public String getcourseID()
    {
        return this.courseID;
```

```
    }
    //Accessor method for courseName
    public String getcourseName()
    {
        return this.courseName;
    }
    //Accessor method for duration
    public int getduration()
    {
        return this.duration;
    }
    //Accessor method for courseLeader
    public String getcourseLeader()
    {
        return this.courseLeader;
    }
    //Mutator method for courseLeader
    public void setcourseLeader(String course_leader)
    {
        this.courseLeader = course_leader;
    }
    //Displays the course details and name of the course leader if the leader has been
assigned
    public void display()
    {
        System.out.println("Course ID: " + getcourseID());
        System.out.println("Course Name: " + getcourseName());
        System.out.println("Duration: " + getduration());
        if (courseLeader != "")
        {
        System.out.println("Course Leader: " + getcourseLeader());
```

```
        }
        else
        {
            System.out.println("The course leader has not been assigned");
        }
    }
}
```

10.2 Academic Course

```
/**
 * The AcademicCourse class consists of six attributes i.e.,
 * lecturerName, level, credit, startingDate, completionDate, noOfAssessments and
isRegistered
 * The class inherits Course class.
 * The class contains a constructor where the variables are initialized, accessor and
mutator methods, a method to register the course
 * and a method to output the details of a course.
 *
 * @author (Aashna Shrestha)
 * @version (11.0.2)
 */

public class AcademicCourse extends Course
{
    //Declares the instance variables
    private String lecturerName;
    private String level;
    private String credit;
    private String startingDate;
    private String completionDate;
    private int noOfAssessments;
```

```
    private boolean isRegistered;

    /* Constructor with six parameters:
     * courseID, courseName, duration, level, credit, noOfAssessments
     */
    AcademicCourse(String courseID, String courseName, int duration,String level,
String credit, int noOfAssessments)
    {
        // Calls the super class, Course
        super(courseID, courseName, duration);

        //Initialize the instance variables
        this.lecturerName ="";
        this.startingDate = "";
        this.completionDate = "";
        this.isRegistered = false;
    }
    //Accessor method of lecturerName
    public String getlecturerName()
    {
        return this.lecturerName;
    }
    //Accessor method of level
    public String getlevel()
    {
        return this.level;
    }
    //Accessor method of credit
    public String getcredit()
    {
        return this.credit;
```

```
    }
    //Accessor method of startingDate
    public String getstartingDate()
    {
        return this.startingDate;
    }
    //Accessor method of completionDate
    public String getcompletionDate()
    {
        return this.completionDate;
    }
    //Accessor method of noOfAssessments
    public int getnoOfAssessments()
    {
        return this.noOfAssessments;
    }
    //Accessor method of isRegistered
    public boolean getisRegistered()
    {
        return this.isRegistered;
    }
    //Mutator method of lecturerName
    public void setlecturerName(String lecturer_name)
    {
        this.lecturerName = lecturer_name;
    }
    //Mutator method of noOfAssessments
    public void setnoOfAssessments(int noOfAssessments)
    {
        this.noOfAssessments = noOfAssessments;
    }
```

```
    /* Checks if an Academic Course has been registered
     * If the course has not been registered, initializes the attributes and registers the course
     */
    public void register(String course_leader,String lecturer_name,String starting_date, String completion_date)
    {
        if (isRegistered == true){
            System.out.println("You have registered for the course.");
            System.out.println("Instructor's Name: " + getlecturerName());
            System.out.println("Starting Date: " + getstartingDate());
            System.out.println("Completion Date: " + getcompletionDate());
        }
        else{
            //Calls the method setcourseLeader() from the super class with its parameter
            super.setcourseLeader(course_leader);

            this.lecturerName = lecturer_name;
            this.startingDate = starting_date;
            this.completionDate = completion_date;
            this.isRegistered = true;

            System.out.println("You have registered for the course.");
            System.out.println("Lecturer: " + getlecturerName());
            System.out.println("Starting Date: " + getstartingDate());
            System.out.println("Completion Date: " + getcompletionDate());
        }
    }
    //Displays the course details
    public void display()
    {
```

```
        //Calls the display() method from the super class
        super.display();
        if (isRegistered == true)
        {
        System.out.println("Lecturer: " + getlecturerName());
        System.out.println("level:" + getlevel());
        System.out.println("credit: " + getcredit());
        System.out.println("Starting Date: " + getstartingDate());
        System.out.println("Completion Date: " + getcompletionDate());
        System.out.println("Number of Assessments: " + getnoOfAssessments());
        }
    }
}
```

10.3 NonAcademicCourse.java

```
/**
 * The NonAcademicCourse class has eight attributes i.e., instructorName, startDate, completionDate, examDate, prerequisite,
 * duration, isRegistered, isRemoved.
 * The class inherits Course class.
 * The class contains a constructor, accessor and mutator methods, methods to register course, remove course,
 * and display the course details.
 *
 * @author (Aashna Shrestha)
 * @version (11.0.2)
 */

public class NonAcademicCourse extends Course
{
    //Declares the instance variables
    private String instructorName;
```

```
    private String startDate;
    private String completionDate;
    private String examDate;
    private String prerequisite;
    private int duration;
    private boolean isRegistered;
    private boolean isRemoved;

    /* Constructor with four parameters i.e.,
     * courseID, courseName, duration, prerequisite
       */
    NonAcademicCourse(String courseID, String courseName, int duration, String
prerequisite)
    {
        super(courseID, courseName, duration);
        this.prerequisite = prerequisite;
        this.startDate = "";
        this.completionDate = "";
        this.examDate = "";
        this.isRegistered = false;
        this.isRemoved = false;
    }
    //Accessor method of instructor method
    public String getinstructorName()
    {
        return this.instructorName;
    }
    //Accessor method of duration
    public int getduration()
    {
        return this.duration;
```

```
    }
    //Accessor method of startDate
    public String getstartDate()
    {
        return this.startDate;
    }
    //Accessor method of completionDate
    public String getcompletionDate()
    {
        return this.completionDate;
    }
    //Accessor method of examDate
    public String getexamDate()
    {
        return this.examDate;
    }
    //Accessor method of prerequisite
    public String getprerequisite()
    {
        return this.prerequisite;
    }
    //Accessor method of isRegistered
    public boolean getisRegistered()
    {
        return this.isRegistered;
    }
    //Accessor method of isRemoved
    public boolean getisRemoved()
    {
        return this.isRemoved;
    }
```

```
    /*Mutator method of instructorName
     *Initializes the instructorName if the course has not been registered
     */
    public void setinstructorName(String instructor_name)
    {
        if (isRegistered == false){
            this.instructorName = instructor_name;
        }
        else{
          System.out.println("Update failed. Changing the instructor name is not
possible.");
        }
    }
    /* Checks if the course has been registered
     * If the course has not been registered, initializes the instructorName and
registers the course
      */
   public void register(String courseLeader, String instructor_name, String
startDate,String completionDate,String examDate)
    {
        if (isRegistered == false){
            // Calls the method setinstructorName with its parameter.
            setinstructorName(instructor_name);
            this.isRegistered = true;
        }
        else{
          System.out.println("The course has already been registered. Instructor name
can not be changed.");
        }
    }
    //Removes the course
```

```
    public void remove()
    {
        if(isRemoved == true){
            System.out.println("The course has been removed.");
        }
        else{
            //Calls the setcourseLeader() method from the super class with its parameter
            super.setcourseLeader("");

            //Initailize the instance variables to remove the course
            this.instructorName = "";
            this.startDate = "";
            this.completionDate = "";
            this.examDate = "";
            this.isRegistered = false;
            this.isRemoved = true;
        }
    }
    //Displays the details of the course
    public void display()
    {
        //Calls the display() method from the super class.
        super.display();
        if (isRegistered == true){
            System.out.println("Instructor Name: " + getinstructorName());
            System.out.println("Start Date: " + getstartDate());
            System.out.println("Completion Date: " + getcompletionDate());
            System.out.println("Exam Date: " + getexamDate());
        }
        else{
            System.out.println("The course has not been registered.");
```

```
        }
      }
    }
```