LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CS4001NI Programming

COURSEWORK-2

Assessment Weightage & Type

30% Individual Coursework

Semester and Year

Spring 2021

Student Name: Subriti Aryal

Group: C13

London Met ID: 20049062

College ID: np01cp4s210044

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

Introduction

About the Coursework

This project relates to the development of a Graphical User Interface (GUI) for a system that stores details of Course including both of Academic and Non-Academic course type. These classes are created during the first part of the coursework where Course class is the parent class and two of its child classes are Academic and Non-Academic courses.

The main motive of the project is to provide an interface for easy adding and registration of courses while also providing options to display or remove the code with ease. The main GUI is built using various of components present within java.awt and javax.swing packages. For the storage of data input via the GUI, it uses an ArrayList of Course type. The buttons are validated with some actions to perform. The objects of classes, Academic_Course and NonAcademic_Course are also created, and various methods of these classes are called in for encouraging proper functioning as well as for production of desired outputs in the GUI.
The main method holds the creation of an object of our class ING_College and calls its method for displaying the GUI.

The block of codes is tested and inspected thoroughly. The errors encountered during the writing of the codes are also corrected for ensuring the production of desired outputs. The try and catch blocks are also used for catching any Number Format Exception that is thrown in converting the string to an integer, here particularly for the case of Duration and NumberOfAssessments. Appropriate dialog box and error messages are also used here to notify users about the passing or failing of the actions they have performed.

Tools Used

Since it is a real-world based scenario which uses the concept of Object-Oriented Programming. All the coding part for this project is done in BlueJ. It is a Java integrated development environment which has a simple interface and hence, is very user friendly for beginners to start with.

The MS Word was used for writing of method descriptions which are the complete descriptions to all the methods used. Additionally, pseudocodes were also written for the blocks of code written in java. The full report for this project was built using MS Word.

The descriptive class diagram providing full information regarding the classes and their attributes and methods are also given below. It was created with the help of a free diagram building software, Draw.io. It is very convenient for creating flowcharts and diagrams.

Class Diagram

Pseudocode

START

CREATE a class ING_College implements ActionListener

DECLARE instance variables

frame as JFrame

main_panel, add_panel, register_panel, add1_panel, register1_panel as JPanel

label, id_label, name_label, duration_label, level_label, credit_label, assessment_label, assessment1_label, lecturer_label, leader_label, start_label, end_label, id_label1, name_label1, duration_label1, pre_label, lecturer_label1, leader_label1, start_label1, end_label1, exam_label as JLabel

id_field, name_field, duration_field, level_field, credit_field, assessment_field, lecturer_field, leader_field, id_field1, name_field1, duration_field1, pre_field, lecturer_field1, leader_field1 as JTextField

year_combo, month_combo, day_combo, years_combo, months_combo, days_combo, startyear_combo, startmonth_combo, startday_combo, endyear_combo, endmonth_combo, endday_combo, examyear_combo, exammonth_combo, examday_combo as JComboBox

add_button, register_button, clear_button, nonacademic_button, display_button, add1_button, register1_button, clear1_button, remove_button, academic_button as JButton

Academic_Course object_academic

NonAcademic_Course object_nonacademic

ArrayList<Course> alist= new ArrayList<Course> ()

CREATE GUI ()

DO

INITIALIZE the JFrame to frame with its title “Course Registration”

INITIALIZE the JPanel to main_panel

SET the Layout of main_panel to null

INITIALIZE the Color of main_panel to main_color

SET background of main_panel to main_color

INITIALIZE the Border of main_panel to border

SET border of main_panel to border

INITIALIZE the JPanel to add_panel

INITIALIZE the Color of add_panel to add_color

SET background of add_panel to add_color

SET the bounds of add_panel to x-axis 15, y-axis 70, width 305 and height300

SET the Layout of add_panel to null

ADD the add_panel to the main_panel

INITIALIZE the JPanel to register_panel

INITIALIZE the Color of register_panel to register_color

SET background of register_panel to register_color

SET the bounds of register_panel to x-axis 340, y-axis 110, width 305 and height 205

SET the Layout of register_panel to null

ADD the register_panel to the main_panel

INITIALIZE the JPanel to add1_panel

SET background of add1_panel to add_color

SET the bounds of add1_panel to x-axis as 15, y-axis as 70, width as 305 and height as 210

SET the Layout of add1_panel to null

ADD the add1_panel to the main_panel

INITIALIZE the JPanel to register1_panel

SET background of register1_panel to register_color

SET the bounds of register1_panel to x-axis 340, y-axis 110, width 305 and height 250

SET the Layout of register1_panel to null

ADD the register1_panel to the main_panel

INITIALIZE JLabel to label with “Academic Course”

SET the bounds of label to x-axis as 250, y-axis as 20, width as 400 and height as 25

SET the foreground of label with Color.blue

INITIALISE the Font to ff ("Times New Roman",Font.BOLD,23)

SET Font of label to ff

ADD the label to the main_panel

INITIALIZE JLabel to id_label with “Course ID”

SET the bounds of id_label to x-axis as 0, y-axis as 10, width as 400 and height as 25

ADD the id_label to the add_panel

INITIALIZE JTextField to id_field

SET the bounds of id_field to x-axis as 100, y-axis as 5, width as 130 and height as 30

ADD the id_field to the add_panel

INITIALIZE JLabel to name_label with “Course Name”

SET the bounds of name_label to x-axis as 0, y-axis as 50, width as 120 and height as 20

ADD the name_label to the add_panel

INITIALIZE JTextField to name_field

SET the bounds of name_field to x-axis as 100, y-axis as 45, width as 200 and height as 30

ADD the name_field to the add_panel

INITIALIZE JLabel to duration_label with “Duration”

SET the bounds of duration_label to x-axis as 0, y-axis as 90, width as 120 and height as 20

ADD the duration_label to the add_panel

INITIALIZE JTextField to duration_field

SET the bounds of duration_field to x-axis as100, y-axis as 85, width as 200 and height as 30

ADD the duration_field to the add_panel

INITIALIZE JLabel to level_label with “Level”

SET the bounds of level_label to x-axis as 0, y-axis as 130, width as 120 and height as 20

ADD the level_label to the add_panel

INITIALIZE JTextField to level_field

SET the bounds of level_field to x-axis as 100, y-axis as 125, width as 200 and height as 30

ADD the level_field to the add_panel

INITIALIZE JLabel to credit_label with “Credit”

SET the bounds of credit_label to x-axis as 0, y-axis as 170, width as 120 and height as 20

ADD the credit_label to the add_panel

INITIALIZE JTextField to credit_field

SET the bounds of credit_field to x-axis as 100, y-axis as 165, width as 200 and height as 30

ADD the credit_field to the add_panel

INITIALIZE JLabel to assessment_label with “No. of”

SET the bounds of assessment_label to x-axis 0, y-axis as 205, width as 120 and height as 20

ADD the assessment_label to the add_panel

INITIALIZE JLabel to assessment1_label with “Assessments”

SET the bounds of assessment1_label to x-axis0, y-axis as 215, width as 120 and height as 20

ADD the assessment1_label to the add_panel

INITIALIZE JTextField to assessment_field

SET the bounds of assessment_field to x-axis100, y-axis 205, width as 200 and height as 30

ADD the assessment_field to the add_panel

INITIALIZE JButton to add_button with “Add Course”

SET the bounds of add_button to x-axis as 100, y-axis as 260, width as 150 and height as 30

SET addActionListener(this) to add_button

ADD the add_button to the add_panel

INITIALIZE JButton to remove_button with “Remove Course”

SET the bounds of remove_button to x-axis120, y-axis as 350, width as 150 and height as 30

SET addActionListener(this) to remove_button

ADD the remove_button to the main_panel

INITIALIZE JButton to display_button with “Display”

SET the bounds of display_button to x-axis 80, y-axis as 400, width as 100 and height as 30

SET addActionListener(this) to display_button

ADD the display_button to the main_panel

INITIALIZE JButton to clear_button with “Clear”

SET the bounds of clear_button to x-axis as 200, y-axis as 400, width as 100 and height as 30

SET addActionListener(this) to clear_button

ADD the clear_button to the main_panel

INITIALIZE JButton to clear1_button with “Clear”

SET the bounds of clear1_button to x-axis 200, y-axis as 400, width as 100 and height as 30

SET addActionListener(this) to clear1_button

ADD the clear1_button to the main_panel

INITIALIZE JLabel to lecturer_label with “Lecturer Name”

SET the bounds of lecturer_label to x-axis as 0, y-axis as 10, width as 200 and height as 20

ADD the lecturer_label to the register_panel

INITIALIZE JTextField to lecturer_field

SET the bounds of lecturer_field to x-axis as 100, y-axis as 5, width as 200 and height as 30

ADD the lecturer_field to the register_panel

INITIALIZE JLabel to leader_label with “Course Leader”

SET the bounds of leader_label to x-axis as 0, y-axis as 50, width as 200 and height as 20

ADD the leader_label to the register_panel

INITIALIZE JTextField to leader_field

SET the bounds of leader_field to x-axis as 100, y-axis as 45, width as 200 and height as 30

ADD the leader_field to the register_panel

INITIALIZE JLabel to start_label with “Start Date”

SET the bounds of start_label to x-axis as 0, y-axis as 90, width as 200 and height as 20

ADD the start_label to the register_panel

INITIALIZE arraylist months with String type

INITIALIZE arraylist years with String type

INITIALIZE arraylist days with String type

INITIALIZE JComboBox to year_combo with years

SET the bounds of year_combo to x-axis as 100, y-axis as 90, width as 60 and height as 20

INITIALIZE JComboBox to month_combo with months

SET the bounds of month_combo to x-axis as 162, y-axis as 90, width as 80 and height as 20

INITIALIZE JComboBox to day_combo with days

SET the bounds of day_combo to x-axis as 244, y-axis as 90, width as 40 and height as 20

ADD the year_combo to the register_panel

ADD the month_combo to the register_panel

ADD the day_combo to the register_panel

INITIALIZE JLabel to end_label with “End Date”

SET the bounds of end_label to x-axis as 0, y-axis as 130, width as 200 and height as 20

ADD the end_label to the register_panel

INITIALIZE JComboBox to years_combo with years

SET the bounds of years_combo to x-axis as 100, y-axis as 130, width as 60 and height as 20

INITIALIZE JComboBox to months_combo with months

SET the bounds of months_combo to x-axis 162, y-axis as 130, width as 80 and height as 20

INITIALIZE JComboBox to days_combo with days

SET the bounds of days_combo to x-axis as 244, y-axis as 130, width as 40 and height as 20

ADD the years_combo to the register_panel

ADD the months_combo to the register_panel

ADD the days_combo to the register_panel

INITIALIZE JButton to register_button with “Register Course”

SET the bounds of register_button to x-axis 90, y-axis as 170, width as 150 and height as 30

SET addActionListener(this) to register_button

ADD the register_button to the register_panel

INITIALIZE JButton to nonacademic_button with “Register for Non-Academic Course”

SET the bounds of nonacademic_button to x-axis 400, y-axis 400, width 250 and height 30

SET addActionListener(this) to nonacademic_button

ADD the nonacademic_button to the main_panel

INITIALIZE JLabel to id_label1 with “Course ID”

SET the bounds of id_label1 to x-axis as 0, y-axis as 10, width as 400 and height as 25

ADD the id_label1 to the add1_panel

INITIALIZE JTextField to id_field1

SET the bounds of id_field1 to x-axis as 100, y-axis as 5, width as 130 and height as 30

ADD the id_field1 to the add1_panel

INITIALIZE JLabel to name_label1 with “Course Name”

SET the bounds of name_label1 to x-axis as 0, y-axis as 50, width as 120 and height as 20

ADD the name_label1 to the add1_panel

INITIALIZE JTextField to name_field1

SET the bounds of name_field1 to x-axis as 100, y-axis as 45, width as 200 and height as 30

ADD the name_field1 to the add1_panel

INITIALIZE JLabel to duration_label1 with “Duration”

SET the bounds of duration_label1 to x-axis as 0, y-axis as 90, width as 120 and height as 20

ADD the duration_label1 to the add1_panel

INITIALIZE JTextField to duration_field1

SET the bounds of duration_field1 to x-axis 100, y-axis as 85, width as 200 and height as 30

ADD the duration_field1 to the add1_panel

INITIALIZE JLabel to pre_label with “Prerequisite”

SET the bounds of pre_label to x-axis as 0, y-axis as 130, width as 120 and height as 20

ADD the pre_label to the add1_panel

INITIALIZE JTextField to pre_field

SET the bounds of pre_field to x-axis as 100, y-axis as 125, width as 200 and height as 30

ADD the pre_field to the add1_panel

INITIALIZE JButton to add1_button with “Add Course”

SET the bounds of add1_button to x-axis as 100, y-axis as 180, width as 150 and height as 30

SET addActionListener(this) to add1_button

ADD the add1_button to the add1_panel

INITIALIZE JLabel to lecturer_label1 with “Instructor Name”

SET the bounds of lecturer_label1 to x-axis as 0, y-axis as 10, width as 200 and height as 20

ADD the lecturer_label1 to the register1_panel

INITIALIZE JTextField to lecturer_field1

SET the bounds of lecturer_field1 to x-axis as 100, y-axis as 5, width as 200 and height as 30

ADD the lecturer_field1 to the register1_panel

INITIALIZE JLabel to leader_label1 with “Course Leader”

SET the bounds of leader_label1 to x-axis as 0, y-axis as 50, width as 200 and height as 20

ADD the leader_label1 to the register1_panel

INITIALIZE JTextField to leader_field1

SET the bounds of leader_field1 to x-axis as 100, y-axis as 45, width as 200 and height as 30

ADD the leader_field1 to the register1_panel

INITIALIZE JLabel to start_label1 with “Start Date”

SET the bounds of start_label1 to x-axis as 0, y-axis as 90, width as 200 and height as 20

ADD the start_label1 to the register1_panel

INITIALIZE JComboBox to startyear_combo with years

SET the bounds of startyear_combo to x-axis100, y-axis as 90, width as 60 and height as 20

INITIALIZE JComboBox to startmonth_combo with months

SET the bounds of startmonth_combo to x-axis 162, y-axis 90, width as 80 and height as 20

INITIALIZE JComboBox to startday_combo with days

SET the bounds of startday_combo to x-axis 244, y-axis as 90, width as 40 and height as 20

ADD the startyear_combo to the register1_panel

ADD the startmonth_combo to the register1_panel

ADD the startday_combo to the register1_panel

INITIALIZE JLabel to end_label1 with “End Date”

SET the bounds of end_label1 to x-axis as 0, y-axis as 130, width as 200 and height as 20

ADD the end_label1 to the register1_panel

INITIALIZE JComboBox to endyear_combo with years

SET the bounds of endyear_combo to x-axis 100, y-axis as 130, width as 60 and height as 20

INITIALIZE JComboBox to endmonth_combo with months

SET the bounds of endmonth_combo to x-axis 162, y-axis 130, width as 80 and height as 20

INITIALIZE JComboBox to endday_combo with days

SET the bounds of endday_combo to x-axis 244, y-axis as 130, width as 40 and height as 20

ADD the endyear_combo to the register1_panel

ADD the endmonth_combo to the register1_panel

ADD the endday_combo to the register1_panel

INITIALIZE JLabel to exam_label with “Exam Date”

SET the bounds of exam_label to x-axis as 0, y-axis as 170, width as 200 and height as 20

ADD the exam_label to the register1_panel

INITIALIZE JComboBox to examyear_combo with years

SET the bounds of examyear_combo to x-axis 100, y-axis 170, width as 60 and height as 20

INITIALIZE JComboBox to exammonth_combo with months

SET the bounds of exammonth_combo to x-axis 162, y-axis170, width as 80 and height as 20

INITIALIZE JComboBox to examday_combo with days

SET the bounds of examday_combo to x-axis 244, y-axis 170, width as 40 and height as 20

ADD the examyear_combo to the register1_panel

ADD the exammonth_combo to the register1_panel

ADD the examday_combo to the register1_panel

INITIALIZE JButton to register1_button with “Register Course”

SET the bounds of register1_button to x-axis 90, y-axis as 220, width as 150 and height as 30

SET addActionListener(this) to register1_button

ADD the register1_button to the register1_panel

INITIALIZE JButton to academic_button with “Register for Academic Course”

SET the bounds of academic_button to x-axis 400, y-axis 400, width 250 and height 30

SET addActionListener(this) to academic_button

ADD the academic_button to the main_panel

ADD main_panel to the frame

SET Visible of add_panel to true

SET Visible of register_panel to true

SET Visible of add1_panel to false

SET Visible of register1_panel to false

SET Visible of remove_button to false

SET Visible of academic_button to true

SET Visible of nonacademic_button to false

SET Visible of clear_button to true

SET Visible of clear1_button to false

SET the Bounds of frame to x-axis as 300, y-axis as 100, width as 700 and height as 500

SET the Resizable of frame to false

SET Visible of frame to true

END DO

CREATE main (String [] args)

DO

INITIALISE ING_College to ing

CALL ing.GUI ( )

END DO

CREATE getCourseID ( )

DO

INITIALIZE this. id_field.getText()

END DO

CREATE getCoursename ( )

DO

INITIALIZE this.name_field.getText()

END DO

CREATE getDuration ( )

DO

INITIALIZE Integer.parseInt(this.duration_field.getText())

END DO

```
CREATE getLevel ()
DO
        INITIALIZE this.level_field.getText()
END DO

CREATE getCredit ( )
DO
        INITIALIZE this.credit_field.getText()
END DO

CREATE getNumberOfAssessments()
DO
        INITIALIZE Integer.parseInt(this.assessment_field.getText())
END DO

CREATE getCourse_leader ( )
DO
        INITIALIZE this.leader_field.getText();
END DO

CREATE getLecturer_name ( )
DO
        INITIALIZE this.lecturer_field.getText()
END DO
```

```
CREATE getStartingDate()
DO
        INITIALIZE
                String year=(year_combo.getSelectedItem()).toString()
                 String month= (month_combo.getSelectedItem()).toString()
                 String day=(day_combo.getSelectedItem()).toString()
                 INITIALIZE (year+""+month+""+day)
END DO
CREATE getCompletionDate()
DO
        INITIALIZE
                String years=(years_combo.getSelectedItem()).toString()
                String months= (months_combo.getSelectedItem()).toString()
                 String days=(days_combo.getSelectedItem()).toString()
                  INITIALIZE (years+""+months+""+days)
END DO

CREATE getCourseID1 ( )
DO
        INITIALIZE this. id_field1.getText()
END DO

CREATE getCoursename1 ( )
DO
        INITIALIZE this.name_field1.getText()
END DO
```

```
CREATE getDuration1 ( )
DO
        INITIALIZE Integer.parseInt(this.duration_field1.getText())
END DO

CREATE getPrerequisite()
DO
        INITIALIZE this.pre_field.getText()
END DO

CREATE getCourse_leader1 ( )
DO
        INITIALIZE this.leader_field1.getText()
END DO

CREATE getLecturer_name1 ( )
DO
        INITIALIZE this.lecturer_field1.getText()
END DO

CREATE getStartingDate1()
DO
        INITIALIZE
                String startyear=(startyear_combo.getSelectedItem()).toString()
                String startmonth= (startmonth_combo.getSelectedItem()).
                toString ()
                String startday=(startday_combo.getSelectedItem()).toString()
```

```
                INITIALIZE (startyear+""+startmonth+""+startday)
END DO

CREATE getCompletionDate1()
DO
        INITIALIZE
                String endyear=(endyear_combo.getSelectedItem()).toString()
                String endmonth= (endmonth_combo.getSelectedItem()).
                toString()
                String endday=(endday_combo.getSelectedItem()).toString()
                INITIALIZE (endyear+""+endmonth+""+endday)
END DO

CREATE getExamDate()
DO
        INITIALIZE
                String examyear= (examyear_combo.getSelectedItem())
                .toString()
                String exammonth= (exammonth_combo.getSelectedItem()).
                toString()
                String examday=(examay_combo.getSelectedItem()).toString()
                INITIALIZE (examyear+""+exammonth+""+examday)
END DO
```

```
CREATE actionPerformed(ActionEvent e)
DO
    IF e.getSource() == nonacademic_button
    DO
        SET Visible of add1_panel to true
        SET Visible of register1_panel to true
        SET Visible of add_panel to false
        SET Visible of register_panel to false

        SET Text of label to "Non-Academic Course")
        SET Visible of remove_button to true
        SET Visible of academic_button to true
        SET Visible of clear1_button to true
        SET Visible of clear_button to false
        SET Visible of nonacademic_button to false
    END DO
    ELSE IF e.getSource() == academic_button
    DO
        SET Visible of add1_panel to false
        SET Visible of register1_panel to false
        SET Visible of add_panel to true
        SET Visible of register_panel to true
        SET Text of label to "Academic Course")
        SET Visible of remove_button to false
```

```
        SET Visible of nonacademic_button to true
        SET Visible of clear1_button to false
        SET Visible of clear_button to true
        SET Visible of academic_button to false
    END DO


    ELSE IF e.getSource() == add_button
    DO
        INITIALIZE boolean academic_add=false
        INITIALIZE ct as 0
        FOR (Course course: alist)
        DO
            IF (course.getCourseID().equals(getCourseID()))
            DO
                INITIALIZE academic_add to true
                break
            END DO
        END DO
        IF (getCourseID().isEmpty() || getCourseName().isEmpty() ||
        getLevel().isEmpty() || getCredit().isEmpty())
        DO
            OUTPUT JOptionPane.showMessageDialog(frame,
            "Please fill in all the fields correctly","Error",
            JOptionPane.WARNING_MESSAGE)
            INITIALIZE ct to 1
```

```
END DO
IF (ct == 0)
DO
    TRY
    DO
        INITIALIZE getDuration()
        TRY
        DO
            INITIALIZE getNumberOfAssessments()
        END DO
        CATCH (NumberFormatException ex)
        DO
            OUTPUT
            JOptionPane.showMessageDialog(frame,"E
            nter a number in the text field of No. of
            assessments", "Alert",
            JOptionPane.WARNING_MESSAGE)
            INITIALIZE ct to 1
        END DO
    END DO
    CATCH (NumberFormatException ex)
    DO
        OUTPUT
        JOptionPane.showMessageDialog(frame,"Enter a
        number in the text field of
        duration","Alert",JOptionPane.WARNING_MESSA
        GE)
```

```
                    INITIALIZE ct to 1
            END DO
        END DO
        IF (academic_add==true)
        DO
            OUTPUT JOptionPane.showMessageDialog(frame,"The
            Course having Course ID : " + getCourseID() + " is
            already added", "Error",
            JOptionPane.ERROR_MESSAGE)
        END DO
        ELSE IF (ct == 0)
        DO
            INITIALIZE object_academic= new Academic_Course
            (getCourseID(),getCourseName(),getDuration(),getLevel(
            ),getCredit(),getNumberOfAssessments())
            ADD object_academic to alist
            OUPUT
            JOptionPane.showMessageDialog(frame,"Successfully
            added to ArrayList ! \n\n Course ID: " +getCourseID() +
            "\n Course Name: " + getCourseName() + "\n Duration: "
            + getDuration() + "\n Level: " +getLevel() + "\n Credit: " +
            getCredit() + "\n Number of Assessments: " +
            getNumberOfAssessments())
        END DO
    END DO
```

```
ELSE IF e.getSource() == add1_button
DO
    INITIALIZE boolean nonacademic_add=false
    INITIALIZE ct as 0
    FOR (Course course: alist)
    DO
        IF (course.getCourseID().equals(getCourseID1()))
        DO
            INITIALIZE nonacademic_add to true
            break
        END DO
    END DO
    IF (getCourseID1().isEmpty() || getCourseName1().isEmpty() ||
    getPrerequisite().isEmpty())
    DO
        OUTPUT JOptionPane.showMessageDialog(frame,
        "Please fill in all the fields correctly","Error",
        JOptionPane.WARNING_MESSAGE);
        INITIALIZE ct to 1
    END DO
    IF (ct == 0)
    DO
        TRY
        DO
            INITIALIZE getDuration1()
```

```
            END DO
            CATCH (NumberFormatException ex)
            DO
                OUTPUT JOptionPane.showMessageDialog(frame,"Enter a number in the text field of duration","Alert",JOptionPane.WARNING_MESSAGE)
                INITIALIZE ct to 1
            END DO
        END DO

        IF (nonacademic_add==true)
        DO
            OUTPUT JOptionPane.showMessageDialog(frame,"The Course having Course ID : " + getCourseID1() + " is already added", "Error", JOptionPane.ERROR_MESSAGE)
        END DO

        ELSE IF (ct == 0)
        DO
            INITIALIZE object_nonacademic= new NonAcademic_Course (getCourseID1(),getCourseName1(),getDuration1(),getPrerequisite())
            ADD object_nonacademic to alist
```

```
                    OUPUT
                    JOptionPane.showMessageDialog(frame,"Successfully
                    added to ArrayList ! \n\n Course ID: " +getCourseID1() +
                    "\n Course Name: " + getCourseName1() + "\n Duration: "
                    + getDuration1() + "\n Prerequisite: " +getPrerequisite())

            END DO

        END DO


        ELSE IF e.getSource() == register_button

        DO

            INITIALIZE counter as false

            INITIALIZE c as 0

            IF (getCourseID().isEmpty() || getCourse_leader().isEmpty() ||
            getLecturer_name().isEmpty() || getStartingDate().isEmpty() ||
            getCompletionDate().isEmpty())

            DO

                OUTPUT JOptionPane.showMessageDialog(frame,
                "Please fill in all the fields correctly","Error",
                JOptionPane.WARNING_MESSAGE);

                INITIALIZE counter to true

            END DO

            IF counter==false

            DO

                FOR Course course: alist

                DO

                    IF course instanceof Academic_Course

                    DO
```

```
INITIALIZE c to 1


IF
course.getCourseID().equals(getCourseID()
DO
    INITIALIZE object_academic =
    (Academic_Course) course
    INITIALIZE c to 2
    IF object_academic .getisRegistered
    ()==true
    DO
        OUTPUT
        JOptionPane.showMessageDi
        alog(frame, "The Course is
        already registered ! \n\n
        Lecturer Name: "+
        getLecturer_name() + "\n Start
        Date: " +getStartingDate() +
        "\n End Date: " +
        getCompletionDate())
        break
    END DO
    ELSE IF
    object_academic.getisRegistered()
    ==false
    DO
        CALL
        object_academic.setRegister(
        getCourse_leader(),
```

```
                    getLecturer_name(),
                    getStartingDate(),
                    getCompletionDate())

                    OUTPUT
                    JOptionPane.showMessageDi
                    alog(frame, "Successfully
                    registered the course! \n\n
                    Course ID: " + getCourseID()
                    + "\n Lecturer Name: "+
                    getLecturer_name() + "\n Start
                    Date: " +getStartingDate() +
                    "\n End Date: " +
                    getCompletionDate())

                    break

                END DO

            END DO

        END DO

    END DO

    IF c==1

    DO

        OUTPUT
        JOptionPane.showMessageDialog(frame, "Please
        fill in the CourseID correctly","Error",
        JOptionPane.ERROR_MESSAGE)

    END DO

    ELSE IF c==0

    DO
```

```
                    OUPUT JOptionPane.showMessageDialog(frame,
                    "No Academic Course has been added
                    yet.","Error", JOptionPane.ERROR_MESSAGE)
            END DO
        END DO
    END DO
    ELSE IF e.getSource() == register1_button
    DO
        INITIALIZE counter as false
        INITIALIZE c as 0
        IF getCourseID1().isEmpty() || getCourse_leader1().isEmpty() ||
        getLecturer_name1() .isEmpty()
        DO
            OUTPUT JOptionPane.showMessageDialog(frame,
            "Please fill in all the fields correctly","Error",
            JOptionPane.WARNING_MESSAGE)
            INITIALIZE counter to true
        END DO
        IF counter==false
        DO
            FOR Course course: alist
            DO
                IF course instanceof NonAcademic_Course
                DO
                    INITIALIZE c to 1
```

```
IF
course.getCourseID1().equals(getCourseID
()
DO
    INITIALIZE object_nonacademic=
    (NonAcademic_Course) course
    INITIALIZE c to  2
    IF
    object_nonacademic.getisRegistered
    ()==true
    DO
        OUTPUT
        JOptionPane.showMessageDi
        alog (frame, "The Course is
        already registered !")
        break
    END DO
    ELSE IF
    object_nonacademic.getisRegistered
    ()==false
    DO
        CALL
        object_nonacademic.setRegis
        ter (getCourse_leader1(),
        getLecturer_name1(),
        getStartingDate1(),
        getCompletionDate1(),
        getExamdate())
```

OUTPUT
JOptionPane.showMessageDialog(frame, "Successfully registered the course! \n\n Course ID: " + getCourseID1() + "\n Instructor Name: " + getLecturer_name1() + "\n Course Leader: " + getCourse_leader1() + "\n Start Date: " +getStartingDate1() + "\n End Date: " + getCompletionDate1() + "\n Exam Date: " + getExamDate())

break

END DO

END DO

END DO

END DO

IF c==1

DO

OUTPUT
JOptionPane.showMessageDialog(frame, "Please fill in the CourseID correctly","Error", JOptionPane.ERROR_MESSAGE)

END DO

ELSE IF c==0

DO

OUPUT JOptionPane.showMessageDialog(frame, "No Non-Academic Course has been added yet.","Error", JOptionPane.ERROR_MESSAGE)

END DO

END DO

END DO

```
ELSE IF e.getSource()==display_button
DO
    IF getCourseID().isEmpty() && getCourseID1().isEmpty()
    DO
        SET Text of id_field to (“1”)
        SET Text of id_field1 to (“1”)
        OUTPUT JOptionPane.showMessageDialog(frame,"No
        Course to display ! ","Error",
        JOptionPane.ERROR_MESSAGE)
    END DO
    INITIALISE String getLabel= label.getText()
    FOR Course course: alist
    DO
        IF getLabel.equals("Academic Course")
        DO
            IF course instanceof Academic_Course
            DO
                INITIALIZE object_academic=
                (Academic_Course) course
                CALL object_academic.display()
                PRINT " "
            END DO
        END DO
```

```
                IF getLabel.equals("Non-Academic Course")
                DO
                    IF course instanceof NonAcademic_Course
                    DO
                        INITIALIZE object_nonacademic=
                        (NonAcademic_Course) course
                        CALL object_nonacademic.display()
                        PRINT " "
                    END DO
                END DO
            END DO
        END DO

        ELSE IF e.getSource()== clear_button
        DO
            SET text of id_field to ("")
            SET text of name_field to ("")
            SET text of duration_field to ("")
            SET text of level_field to ("")
            SET text of credit_field to ("")
            SET text of assessment_field to ("")
            SET text of leader_field to ("")
            SET text of lecturer_field to ("")
            SET SelectedIndex of year_combo to (0)
            SET SelectedIndex of month_combo to (0)
```

```
            SET SelectedIndex of day_combo to (0)
            SET SelectedIndex of years_combo to (0)
            SET SelectedIndex of months_combo to (0)
            SET SelectedIndex of days_combo to (0)
        END DO
        ELSE IF e.getSource()== clear1_button
        DO
            SET text of id_field1 to ("")
            SET text of name_field1 to ("")
            SET text of duration_field1 to ("")
            SET text of pre_field to ("")
            SET text of leader_field1 to ("")
            SET text of lecturer_field1 to ("")
            SET SelectedIndex of startyear_combo to (0)
            SET SelectedIndex of startmonth_combo to (0)
            SET SelectedIndex of startday_combo to (0)
            SET SelectedIndex of endyear_combo to (0)
            SET SelectedIndex of endmonth_combo to (0)
            SET SelectedIndex of endday_combo to (0)
            SET SelectedIndex of examyear_combo to (0)
            SET SelectedIndex of exammonth_combo to (0)
            SET SelectedIndex of examday_combo to (0)
        END DO
```

```
ELSE IF e.getSource()== remove_button
DO
    FOR Course course: alist
    DO
        IF course.getCourseID().equals (getCourseID1())
        DO
            IF course instanceof NonAcademic_Course
            DO
                INITIALIZE object_nonacademic=
                (NonAcademic_Course) course
                IF
                object_nonacademic.getisRemoved()==true
                DO
                    OUTPUT
                    JOptionPane.showMessageDialog(fr
                    ame,"The Course is already removed
                    ! ")
                    break
                END DO
                ELSE IF
                object_nonacademic.getisRemoved()==fals
                e
                DO
                    CALL
                    object_nonacademic.Remove()
                    OUPUT
                    JOptionPane.showMessageDialog(fr
```

```
                                        ame, "Successfully removed the
                                        course!")
                                        break
                                END DO
                        END DO
                END DO
                ELSE
                DO
                        OUTPUT
                        JOptionPane.showMessageDialog(frame, "Please
                        input valid CourseID","Error",
                        JOptionPane.ERROR_MESSAGE)
                END DO
        END DO
        IF getCourseID1().isEmpty()
        DO
                OUTPUT JOptionPane.showMessageDialog(frame,
                "Please input the CourseID for the course you would like
                to remove","Error", JOptionPane.ERROR_MESSAGE)
        END DO
    END DO
  END DO
END
```

Method Description

ING_College.java

void GUI ()	void GUI () is the name of the method. It is used to build the complete GUI for the registration of courses. This method has no return value since it is void.
void main (String [] args)	void main (String [] args) is the main method of our class “ING_College” It is used to create an object of ING_College and call its method GUI ().
String getCourseID ()	getCourseID () is the name of the method. It is used to initialize the value of CourseID. It returns a value with a String datatype.
String getCourseName ()	getCourseName () is the name of the method. It is used to initialize the value of Coursename. It returns a value with a String datatype.
int getDuration ()	getDuration () is the name of the method. It is used to initialize the value of Duration. It returns a value with an int datatype.
String getLevel ()	getLevel () is the name of the method. It is used to initialize the value of Level. It returns a value with a String datatype.

String getCredit ()	getCredit () is the name of the method. It is used to initialize the value of Credit. It returns a value with a String datatype.
int getNumberOfAssessments ()	getNumberOfAssessments () is the name of the method. It is used to initialize the value of NumberOfAssessments. It returns a value with an int datatype.
String getCourse_leader ()	getCourse_leader () is the name of the method. It is used to initialize the value of Course_leader. It returns a value with a String datatype.
String getLecturer_name ()	getLecturer_name () is the name of the method. It is used to initialize the value of Lecturer_name. It returns a value with a String datatype.
String getStartingDate ()	getStartingDate () is the name of the method. It is used to initialize the value of StartingDate. It returns a value with a String datatype.
String getCompletionDate ()	getCompletionDate () is the name of the method. It is used to initialize the value of CompletionDate. It returns a value with a String datatype.

String getCourseID1 ()	getCourseID1 () is the name of the method. It is used to initialize the value of CourseID1. It returns a value with a String datatype.
String getCourseName1()	getCourseName1 () is the name of the method. It is used to initialize the value of Coursename1. It returns a value with a String datatype.
int getDuration1()	getDuration1 () is the name of the method. It is used to initialize the value of Duration1. It returns a value with an int datatype.
String getPrerequisite ()	getPrerequisite () is the name of the method. It is used to initialize the value of Prerequisite. It returns a value with a String datatype.
String getCourse_leader1 ()	getCourse_leader1 () is the name of the method. It is used to initialize the value of Course_leader1. It returns a value with a String datatype.
String getLecturer_name1()	getLecturer_name1() is the name of the method. It is used to initialize the value of Lecturer_name1. It returns a value with a String datatype.

String getStartingDate1()	getStartingDate1() is the name of the method. It is used to initialize the value of StartingDate1. It returns a value with a String datatype.
String getCompletionDate1()	getCompletionDate1() is the name of the method. It is used to initialize the value of CompletionDate1. It returns a value with a String datatype.
String getExamDate ()	getExamDate () is the name of the method. It is used to initialize the value of ExamDate. It returns a value with a String datatype.
void actionPerformed (ActionEvent e)	void actionPerformed () is the name of the method. (ActionEvent e) is the parameter passed to the method. It is used to add functionality to the buttons and specify the actions it needs to perform, on being pressed.
• if(e.getSource() == nonacademic_button)	On the press of this button, the existing panel holding JLabels, JTextFields and JButtons which took input of the details regarding Academic Course is replaced by another panel for allowing the input of NonAcademic Courses. The main heading for the frame is also changed using setText() method and the visibility of the buttons are also adjusted accordingly.

• else if(e.getSource() == academic_button)	On the press of this button, the existing panel holding JLabels, JTextFields and JButtons which took input of the details regarding NonAcademic Course is again replaced by the first panel for allowing the input of Academic Courses. The main heading for the frame is also changed using setText() method and the visibility of the buttons are also adjusted accordingly.
• else if(e.getSource() == add_button)	An object of type Academic_Course "object_academic" is created which takes all the getter method of CourseID, CourseName, Duration, Level, Credit and NumberOfAssessments as its parameter and later gets added to the ArrayList of Course class.
• else if(e.getSource() == add1_button)	An object of type NonAcademic_Course "object_nonacademic" is created which takes all the getter method of CourseID, CourseName, Duration and Prerequisite as its parameter and later gets added to the ArrayList of Course class.
• else if(e.getSource() == register_button)	The CourseID entered in the gui is compared to the CourseID in the ArrayList and if it is found valid, the object of Course class is casted as Academic_Course type and the method to register is called from the Academic_Course class.

• else if(e.getSource() == register1_button)	The CourseID entered in the gui is compared to the CourseID in the ArrayList and if it is found valid, the object of Course class is casted as NonAcademic_Course type and the method to register is called from the NonAcademic_Course class.
• else if(e.getSource()==display_button)	The for loop is used to iterate over the ArrayList and if it finds the instance of Academic_Course, it calls the display method from the Academic_Course class to display the course details. And, if it finds the instance of NonAcademic_Course, it calls the display method from NonAcademic_Course class to display the corresponding course details.
• else if(e.getSource()== clear_button) • else if(e.getSource()== clear1_button)	On the press of this button, all the JTextFields are made to reset its stored value to an empty string. It is done using the setText(“ “) method.
• else if(e.getSource()== remove_button)	The CourseID entered in the gui is compared to the CourseID in the ArrayList and if it is found valid, the object of Course class is casted as NonAcademic_Course type and the method to remove is called from the NonAcademic_Course class.

Table 1: Method Description for class ING_College

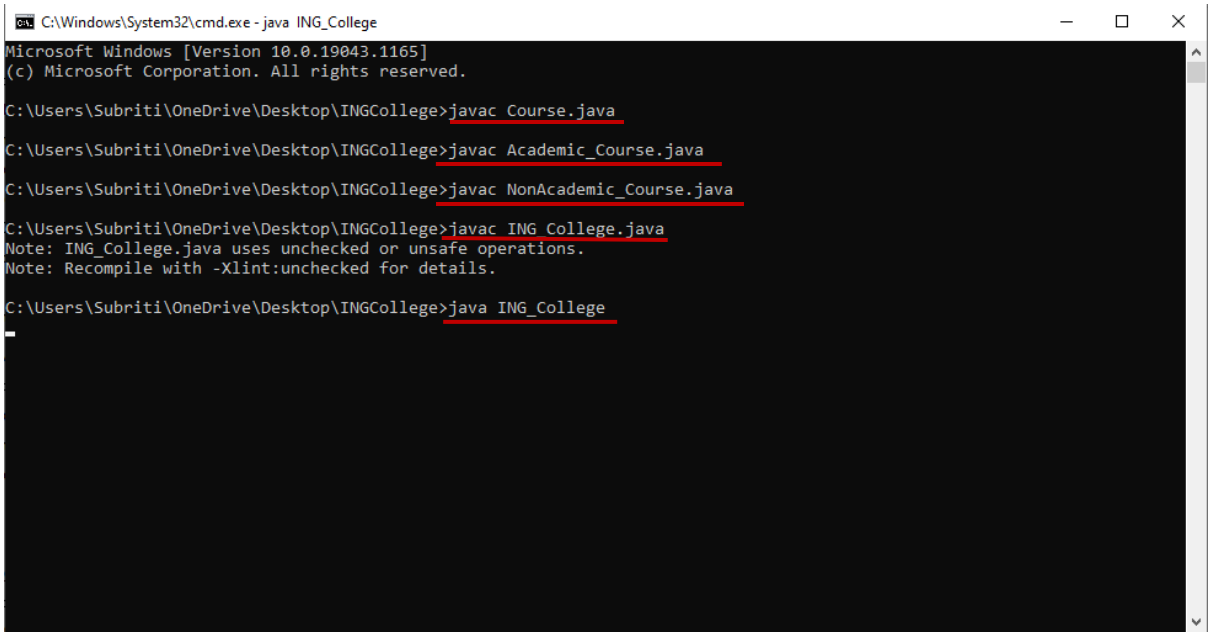
Testing

Test 1: Test that the program can be compiled and run using the command prompt

Test Number :	1
Objective :	To test that the program can be compiled and run using the command prompt.
Action :	Following actions were performed for this objective ○ Command prompt was opened from the same folder where the code was located ○ Command entered in the command prompt for the compilation of Course class was “javac .java “ ○ Command entered in the command prompt for the compilation of Academic_Course was “javac ING_College.java “ ○ Command entered in the command prompt for the compilation of NonAcademic_Course was “javac ING_College.java “ ○ Command entered in the command prompt for running the program was “ java ING_College “
Expected Result :	The program would be compiled and run via the command prompt.
Actual Result :	The program compiled and ran successfully, displaying the GUI.
Conclusion :	The test is successful.

Table 2: Testing of program using command prompt

Output of Test 1:



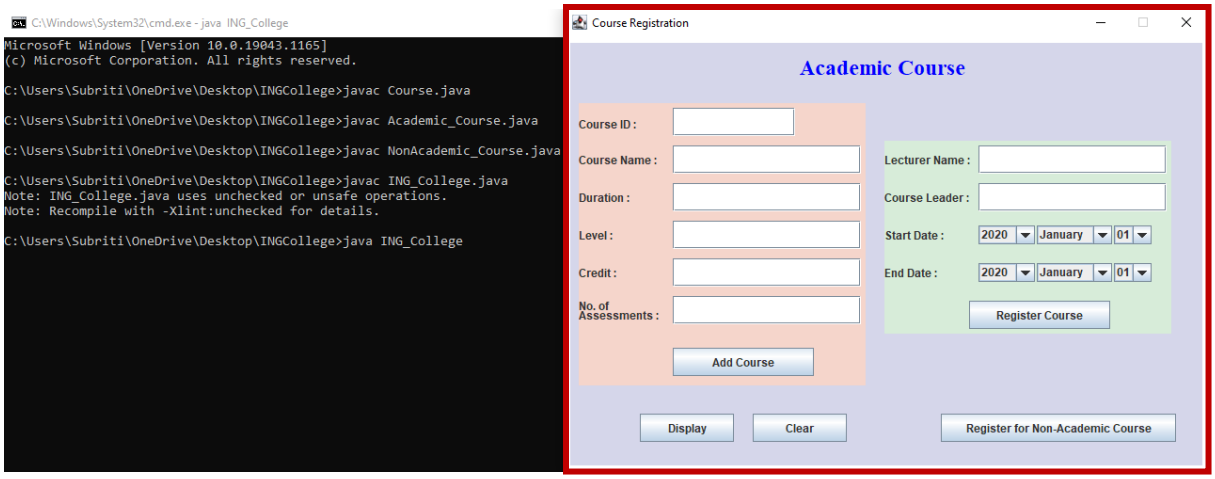


Figure 1: Testing of program from the command prompt

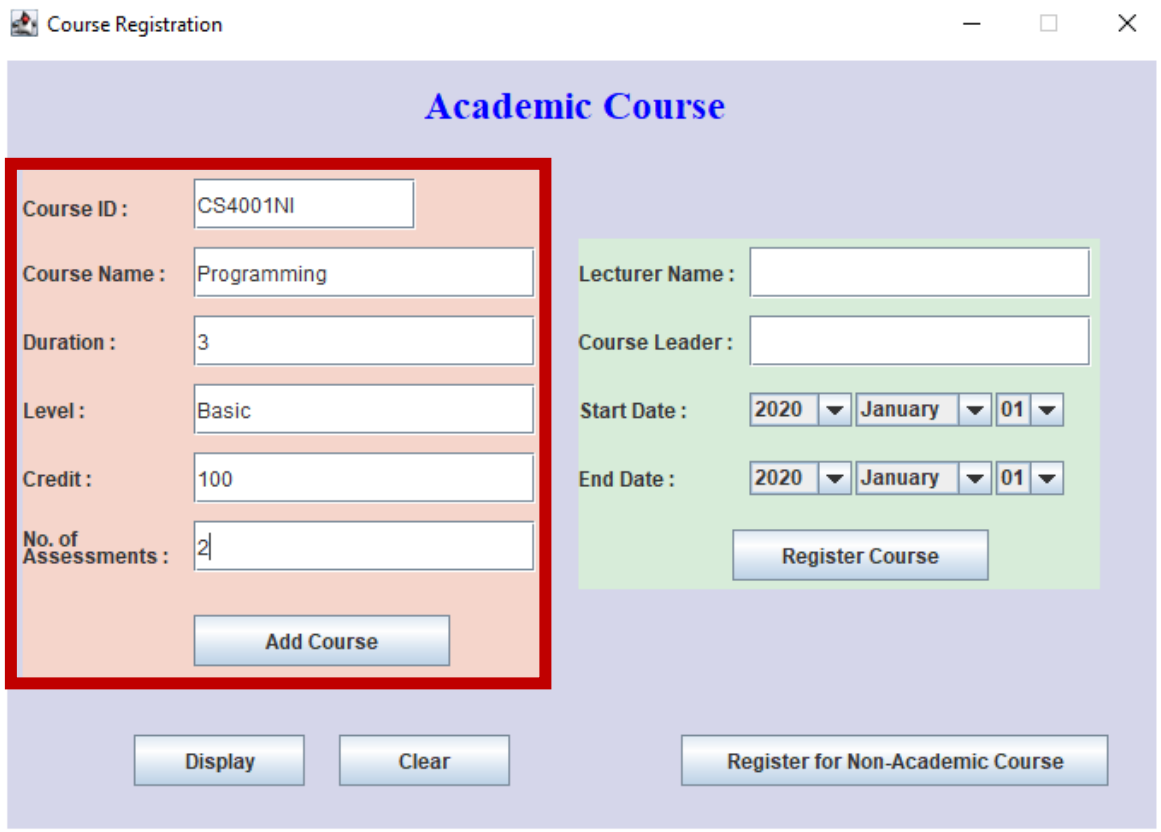
Test 2: Evidence should be shown of:

a. Add course for Academic course

Test Number :	2(a)
Objective :	To Add course for Academic course
Action :	Following actions were performed for this objective ○ Command prompt was opened from the same folder where the code was located ○ Command entered in the command prompt for the compilation of Course class was “javac .java “ ○ Command entered in the command prompt for the compilation of Academic_Course was “javac ING_College.java “ ○ Command entered in the command prompt for the compilation of NonAcademic_Course was “javac ING_College.java “ ○ Command entered in the command prompt for running the program was “ java ING_College “ ○ All the text fields were filled with some values for the course to be added. The values passed were Course ID: CS4001NI Course Name: Programming Duration: 3 Level: Basic Credit: 100 Number of Assessments: 2
Expected Result :	The course with all the filled in details would be added.
Actual Result :	The course was successfully added to the arraylist.
Conclusion :	The test is successful.

Table 3: Adding of Academic Course

Output of Test 2(a) :



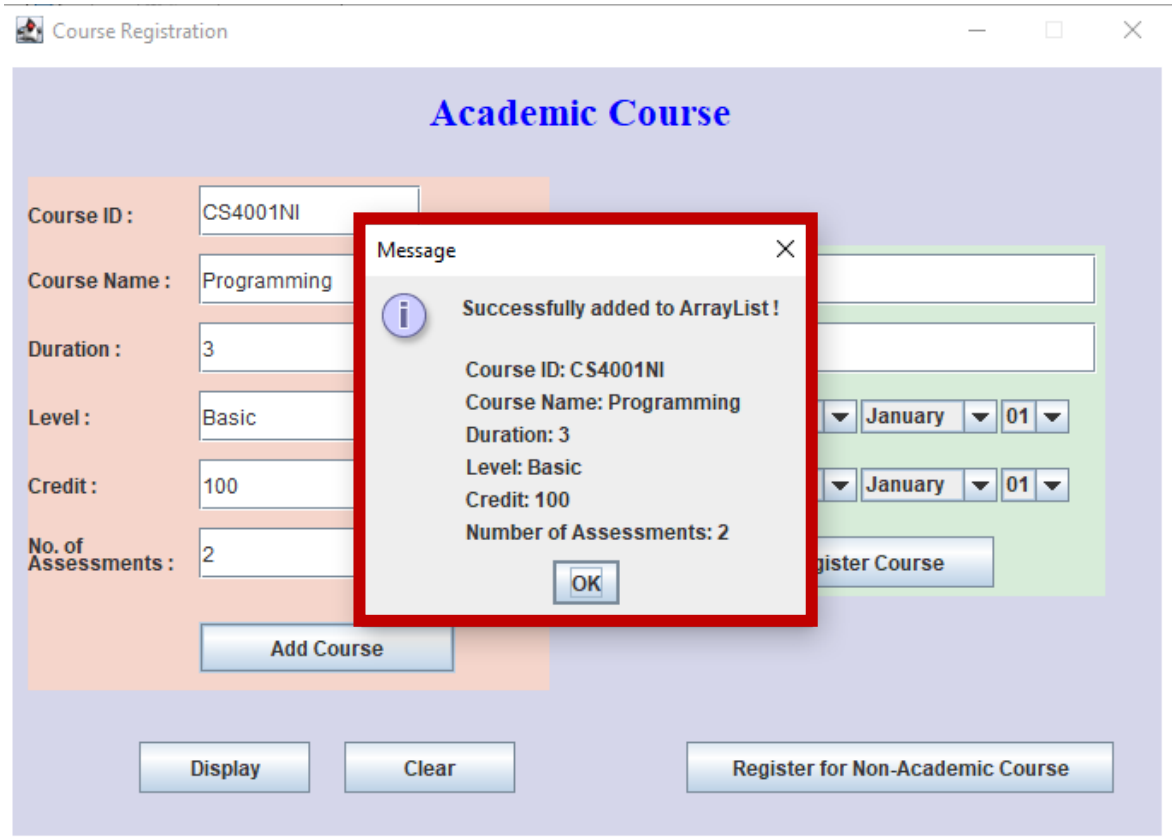


Figure 2 : Adding of an Academic Course

b. Add course for Non-academic course

Test Number :	2(b)
Objective :	To Add course for Non-Academic course
Action :	Following actions were performed for this objective ○ Command prompt was opened from the same folder where the code was located ○ Command entered in the command prompt for the compilation of Course class was “javac .java “ ○ Command entered in the command prompt for the compilation of Academic_Course was “javac ING_College.java “ ○ Command entered in the command prompt for the compilation of NonAcademic_Course was “javac ING_College.java “ ○ Command entered in the command prompt for running the program was “ java ING_College “ ○ All the text fields were filled with some values for the course to be added. The values passed were Course ID: CC4057NI Course Name: Database Duration: 1 Prerequisite: +2pass
Expected Result :	The course with all the filled in details would be added.
Actual Result :	The course was successfully added to the arraylist.
Conclusion :	The test is successful.

Table 4: Adding of Non-Academic Course

Output of Test 2 (b) :

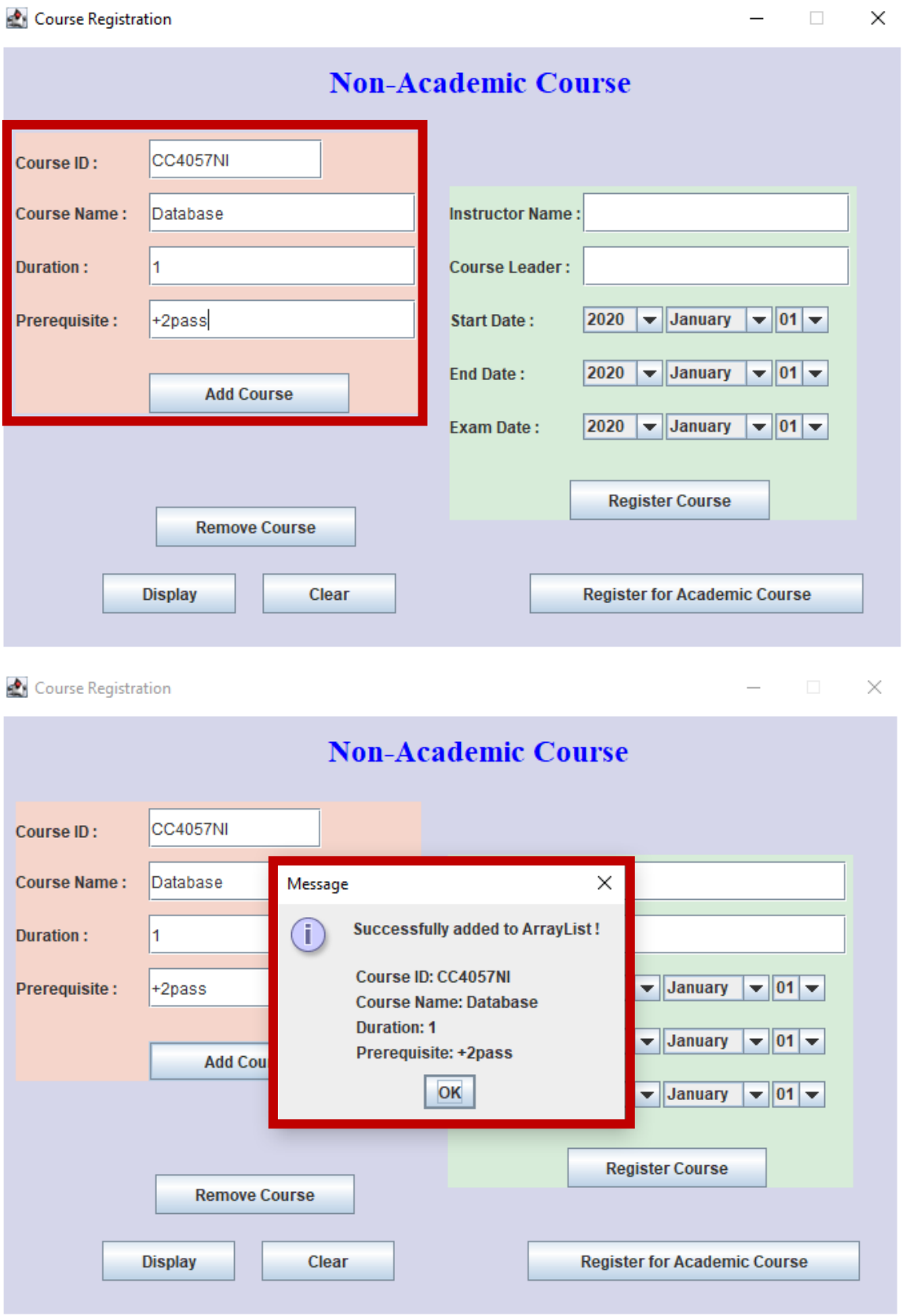


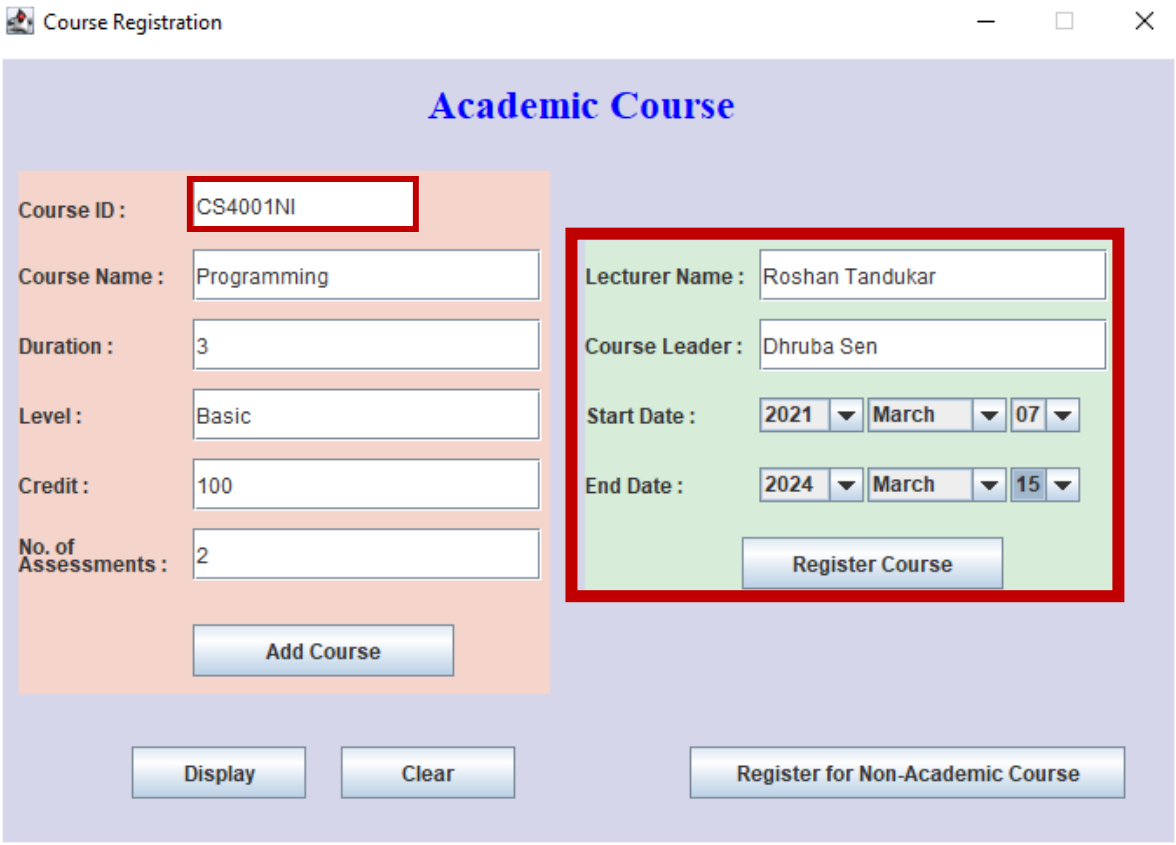
Figure 3 : Adding of a Non-Academic Course

c. Register Academic course

Test Number :	2(c)
Objective :	To Register course for Academic course
Action :	Following actions were performed for this objective ○ Command prompt was opened from the same folder where the code was located ○ Command entered in the command prompt for the compilation of Course class was “javac .java “ ○ Command entered in the command prompt for the compilation of Academic_Course was “javac ING_College.java “ ○ Command entered in the command prompt for the compilation of NonAcademic_Course was “javac ING_College.java “ ○ Command entered in the command prompt for running the program was “ java ING_College “ ○ All the text fields were filled with some values for the course to be registered. The values passed were CourseID: CS4001NI Lecturer Name: Roshan Tandukar Course Leader: Dhruba Sen Start Date: 2021 March 07 End Date: 2024 March 15
Expected Result :	The course with all the filled in details would be registered.
Actual Result :	The course was successfully registered.
Conclusion :	The test is successful.

Table 5: Registration of Academic Course

Output of Test 2 (c) :



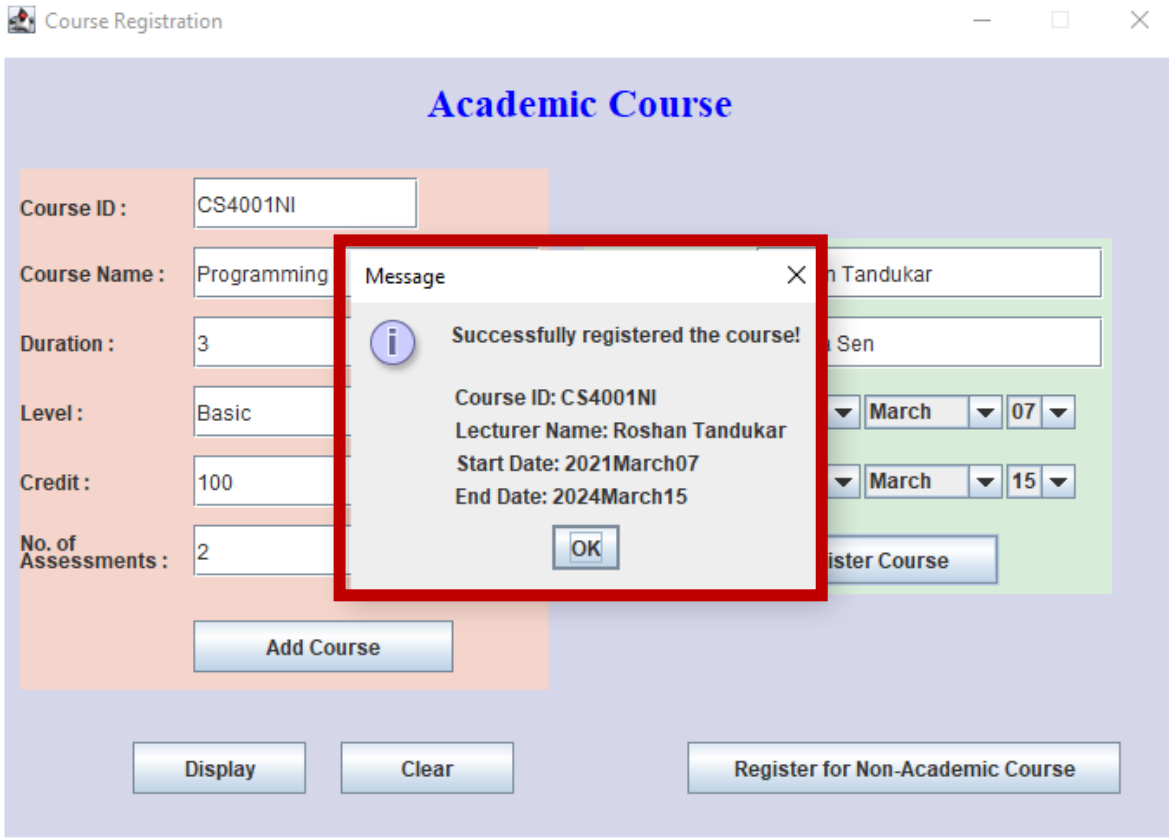


Figure 4 : Registering of an Academic Course

d. Register Non-Academic course

Test Number :	2(d)
Objective :	To Register course for Non-Academic course
Action :	Following actions were performed for this objective ○ Command prompt was opened from the same folder where the code was located ○ Command entered in the command prompt for the compilation of Course class was “javac .java “ ○ Command entered in the command prompt for the compilation of Academic_Course was “javac ING_College.java “ ○ Command entered in the command prompt for the compilation of NonAcademic_Course was “javac ING_College.java “ ○ Command entered in the command prompt for running the program was “ java ING_College “ ○ All the text fields were filled with some values for the course to be registered. The values passed were CourseID: CC4057NI Lecturer Name: Bibek Raj Joshi Course Leader: Sukrit Shakya Start Date: 2021 March 07 End Date: 2022 March 01 Exam Date: 2022 February 25
Expected Result :	The course with all the filled in details would be registered.
Actual Result :	The course was successfully registered.
Conclusion :	The test is successful.

Table 6: Registration of Non-Academic Course

Output of Test 2 (d) :

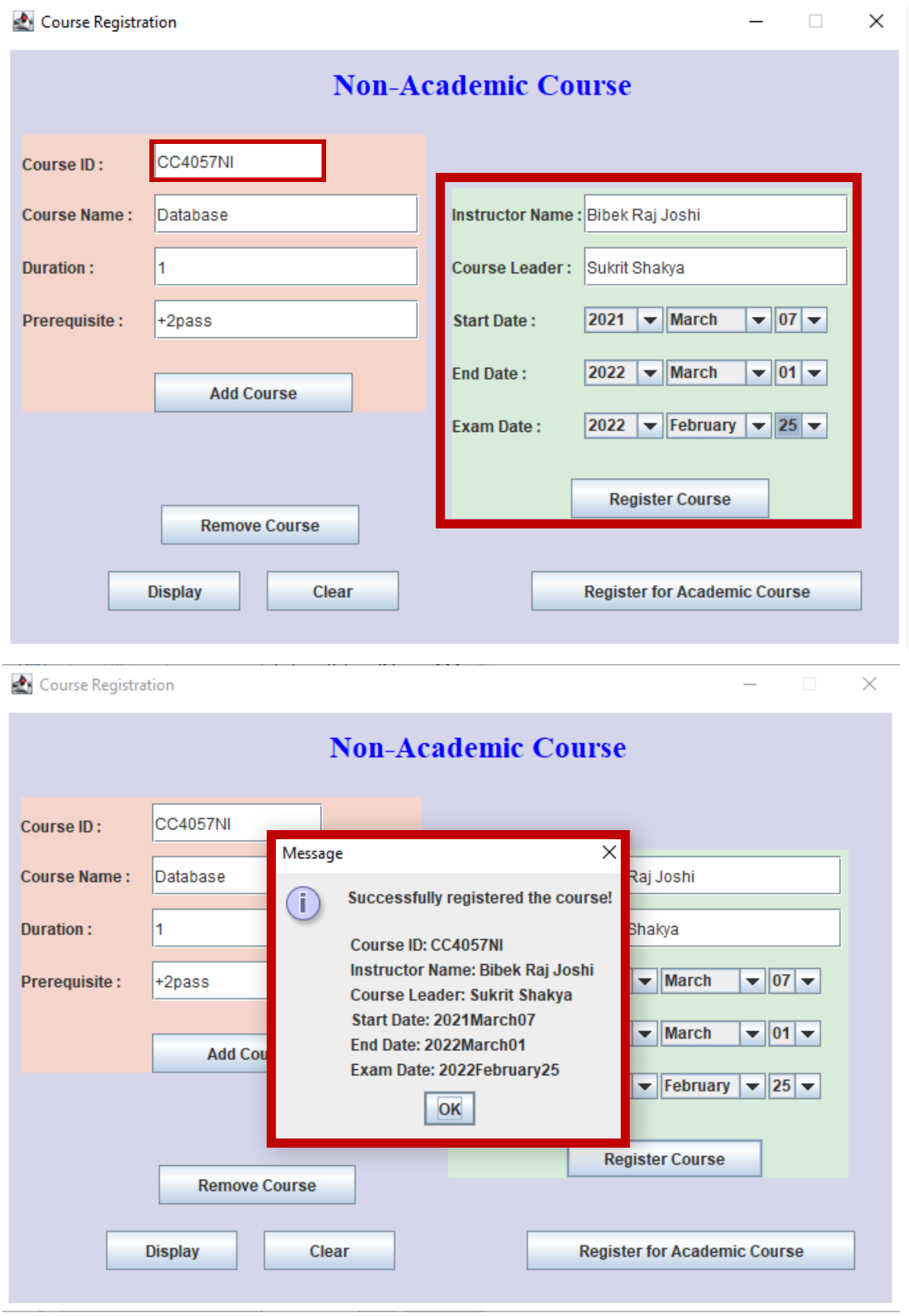


Figure 5 : Registering of Non-Academic Course

e. Remove non-academic course

Test Number :	2(e)
Objective :	To Remove a Non-Academic course
Action :	Following actions were performed for this objective ○ Command prompt was opened from the same folder where the code was located ○ Command entered in the command prompt for the compilation of Course class was “javac .java “ ○ Command entered in the command prompt for the compilation of Academic_Course was “javac ING_College.java “ ○ Command entered in the command prompt for the compilation of NonAcademic_Course was “javac ING_College.java “ ○ Command entered in the command prompt for running the program was “ java ING_College “ ○ After the course had been added and registered, the button which said “Remove” was clicked.
Expected Result :	The course would be removed on the press of the Remove button
Actual Result :	The course was successfully removed.
Conclusion :	The test is successful.

Table 7: Removal of Non-Academic Course

Output of Test 2 (e) :

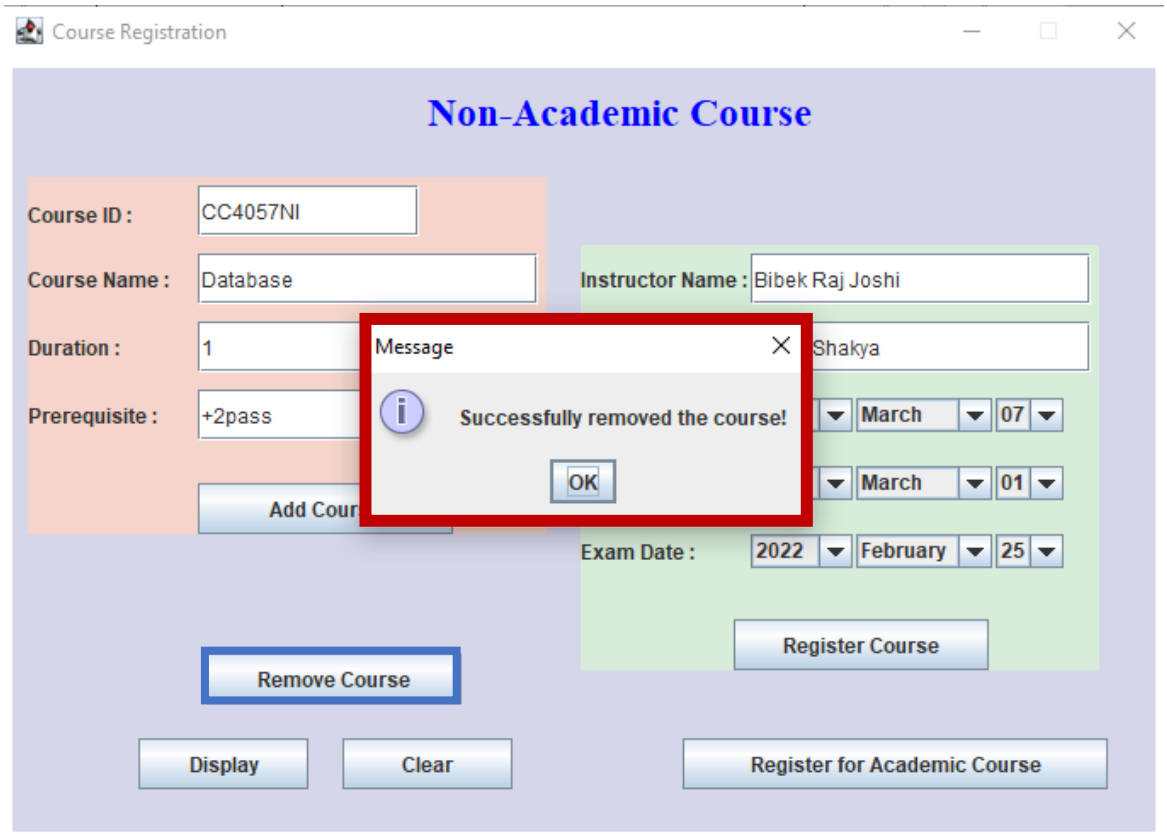


Figure 6: Removing of Non-Academic Course

Test 3: Test that appropriate dialog boxes appear when:

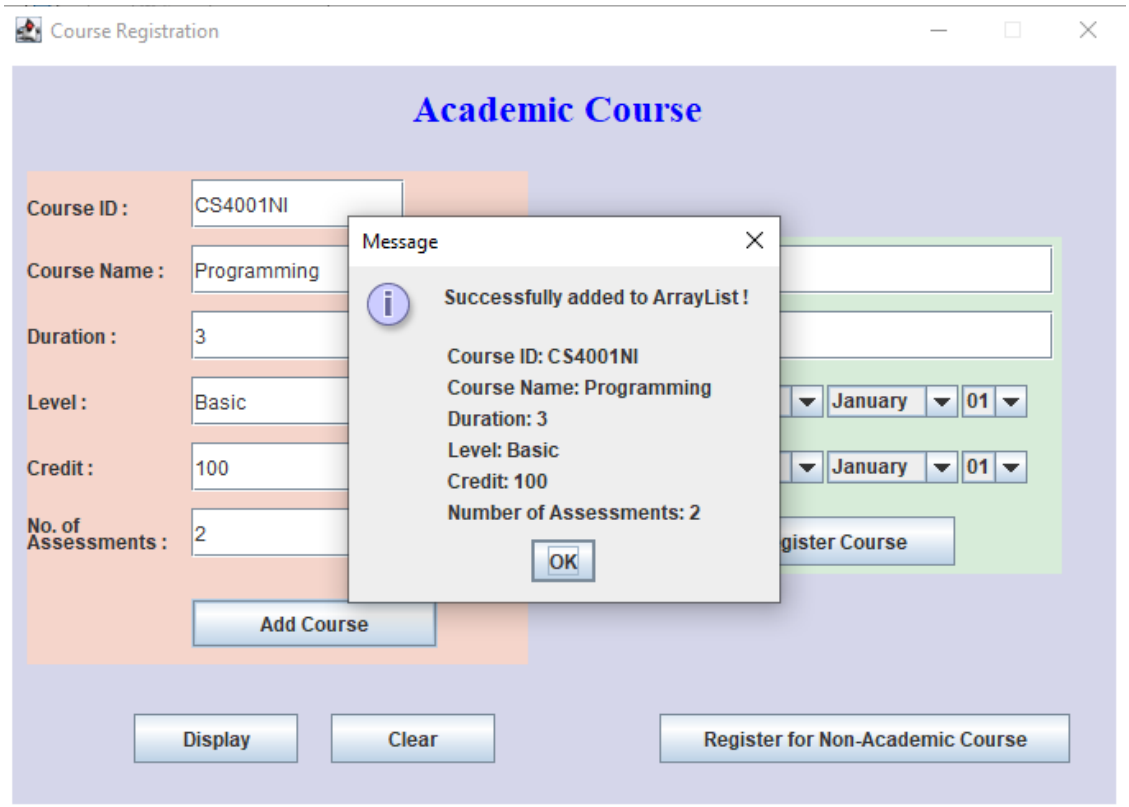
a. Trying to add duplicate CourseID

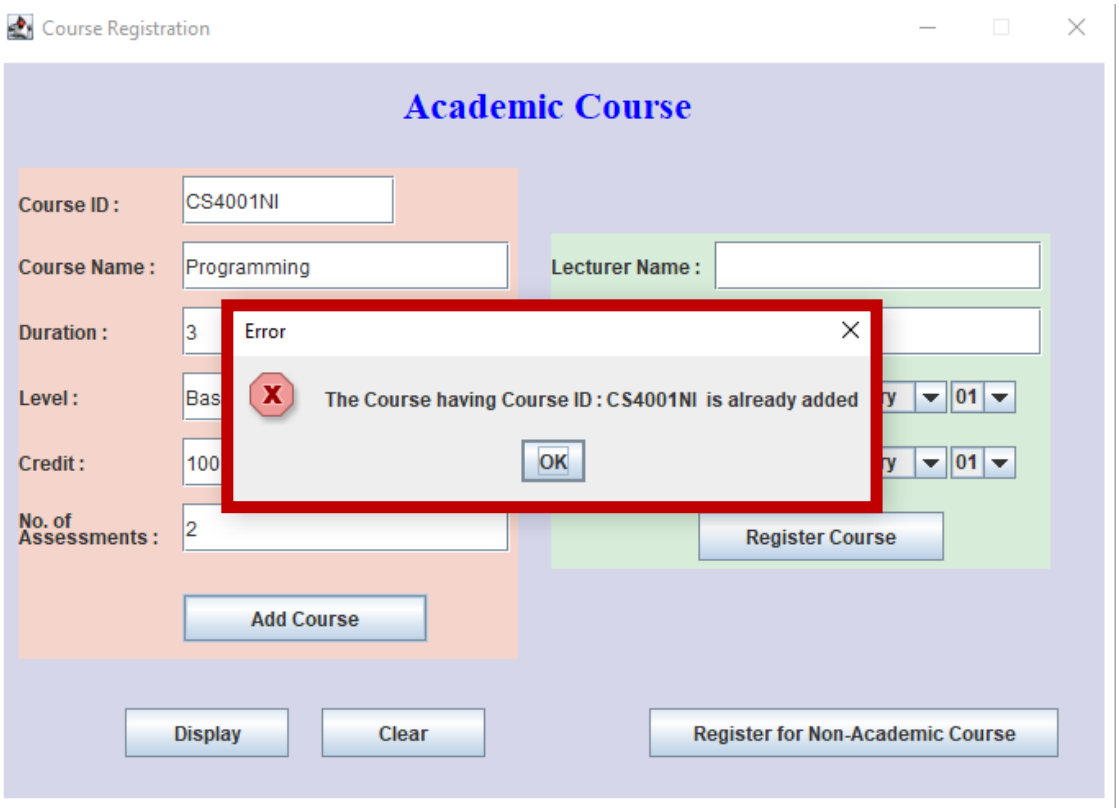
Test Number :	3(a)
Objective :	To test the appearance of dialog boxes while entering a duplicate Course ID
Action :	Following actions were performed for this objective ○ Command prompt was opened from the same folder where the code was located ○ Command entered in the command prompt for the compilation of Course class was “javac .java “

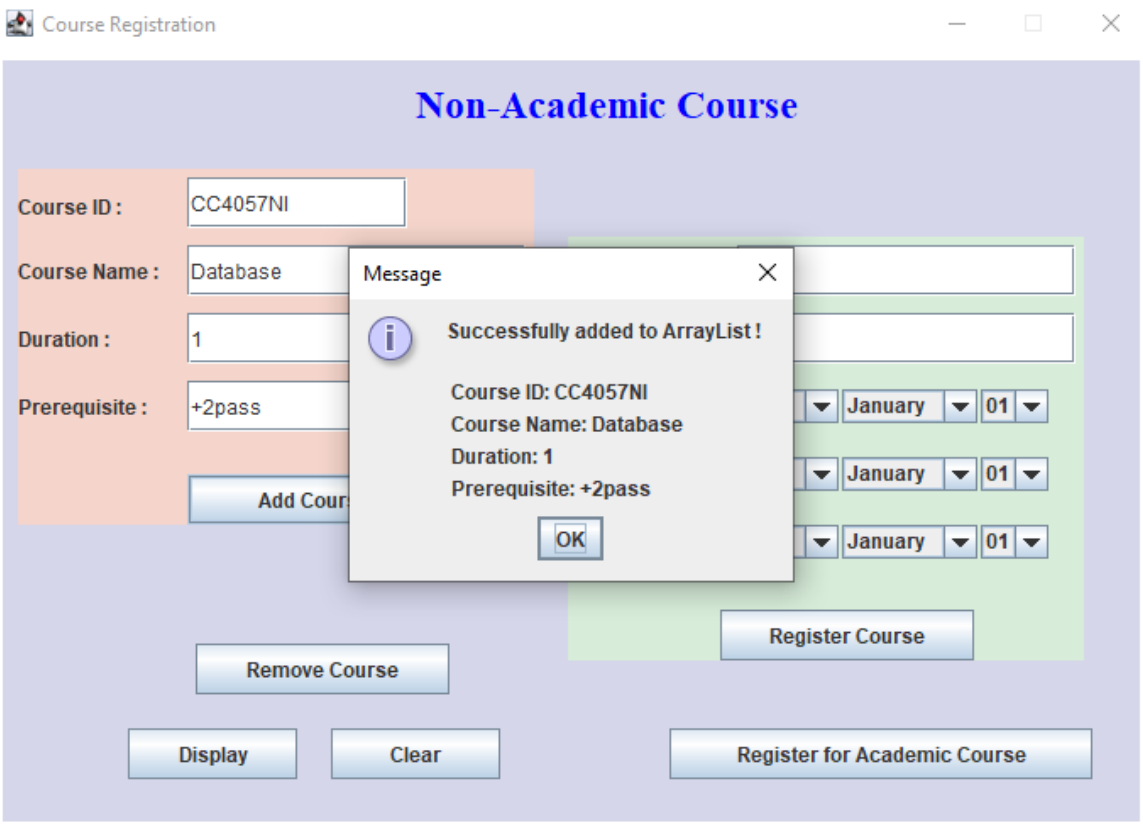
	○ Command entered in the command prompt for the compilation of Academic_Course was "javac ING_College.java " ○ Command entered in the command prompt for the compilation of NonAcademic_Course was "javac ING_College.java " ○ Command entered in the command prompt for running the program was " java ING_College " ○ All the text fields were filled with some values for the course to be added. The values passed were <u>For Academic Course</u> Course ID: CS4001NI Course Name: Programming Duration: 3 Level: Basic Credit: 100 Number of Assessments: 2 <u>For Non-Academic Course</u> CourseID: CC4057NI Course Name: Database Duration: 1 Prerequisite: +2pass
Expected Result :	The program would generate a dialog box saying the course has already been added.
Actual Result :	The program generated a dialog box saying the course with the currently filled coursed is already added.
Conclusion :	The test is successful.

Table 8: Entry of Duplicate Course ID

Output of Test 3 (a):







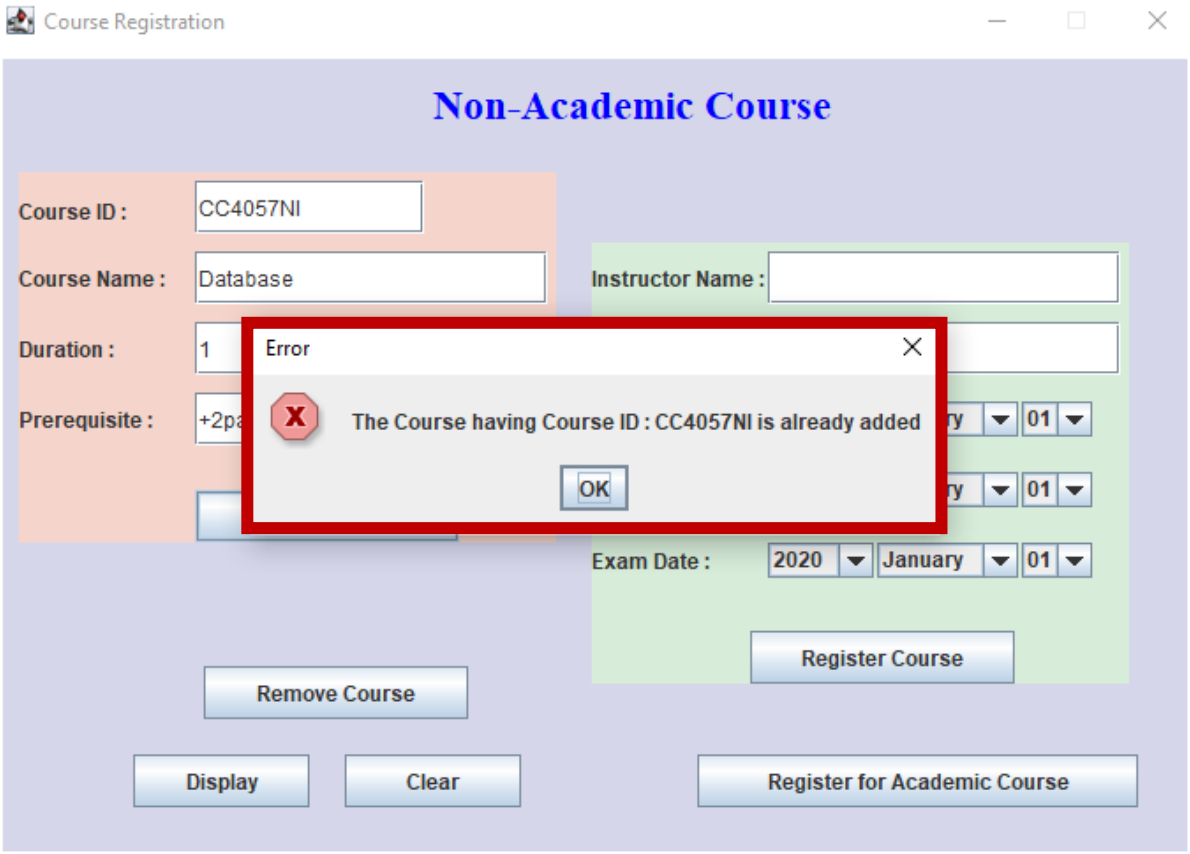


Figure 7: Adding of Course with duplicate Course ID

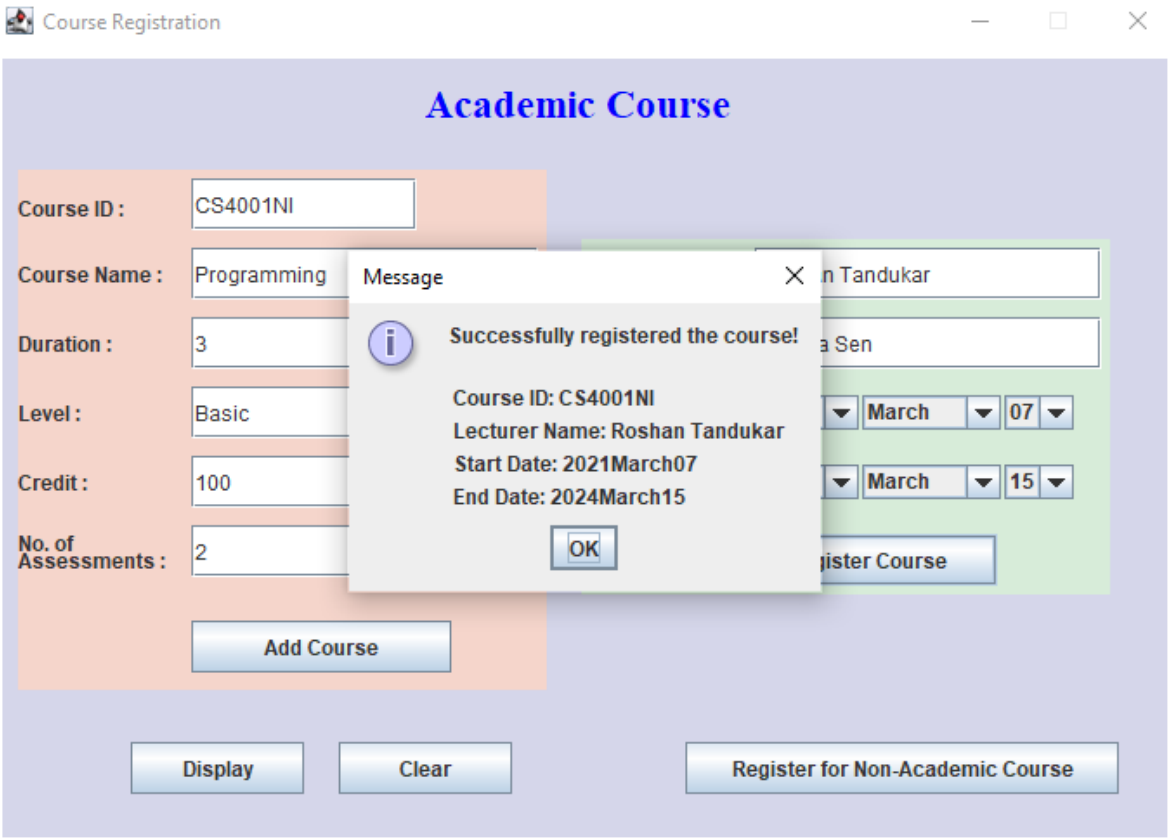
b. Trying to register already registered course

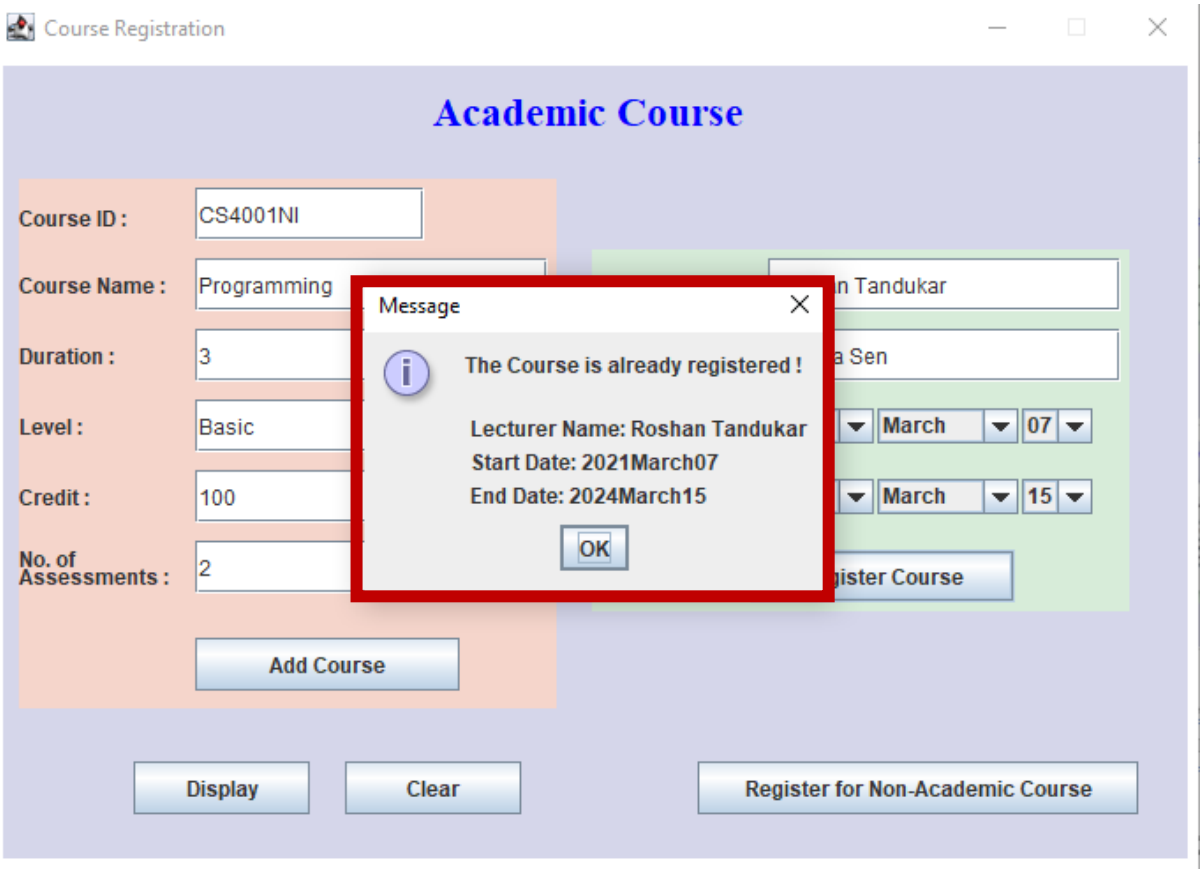
Test Number :	3(b)
Objective :	To test the appearance of dialog boxes while trying to register an already registered course.
Action :	Following actions were performed for this objective ○ Command prompt was opened from the same folder where the code was located ○ Command entered in the command prompt for the compilation of Course class was “javac .java “ ○ Command entered in the command prompt for the compilation of Academic_Course was “javac ING_College.java “ ○ Command entered in the command prompt for the compilation of NonAcademic_Course was “javac ING_College.java “ ○ Command entered in the command prompt for running the program was “ java ING_College “ ○ All the text fields were filled with some values for the course to be registered. The values passed were <u>For Academic Course</u> CourseID: CS4001NI Lecturer Name: Roshan Tandukar Course Leader: Dhruba Sen Start Date: 2021 March 07 End Date: 2024 March 15

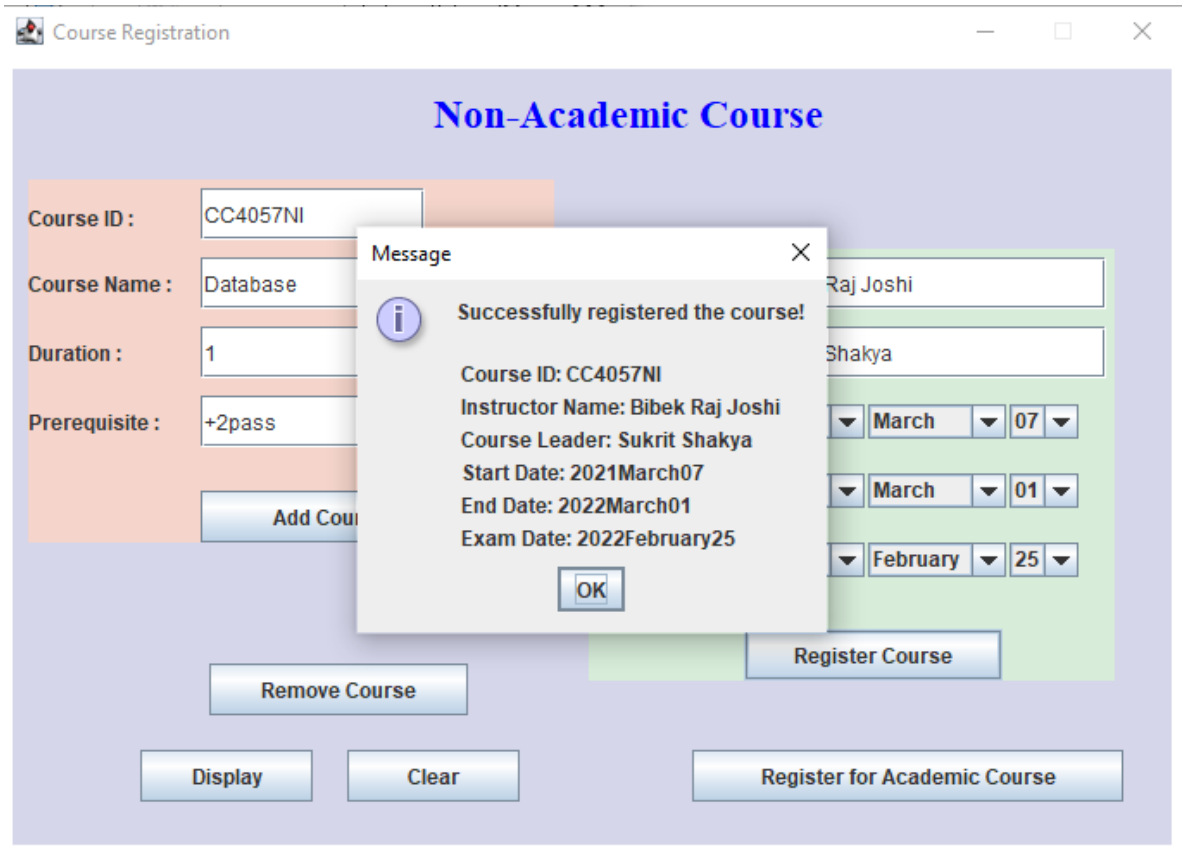
	For Non-Academic Course CourseID: CC4057NI Lecturer Name: Bibek Raj Joshi Course Leader: Sukrit Shakya Start Date: 2021 March 07 End Date: 2022 March 01 Exam Date: 2022 February 25
Expected Result :	The program would generate a dialog box saying the course has already been registered.
Actual Result :	The program generated a dialog box saying the course is already registered and showed additional details about the course.
Conclusion :	The test is successful.

Table 9: Registration of already Registered Course

Output of Test 3 (b) :







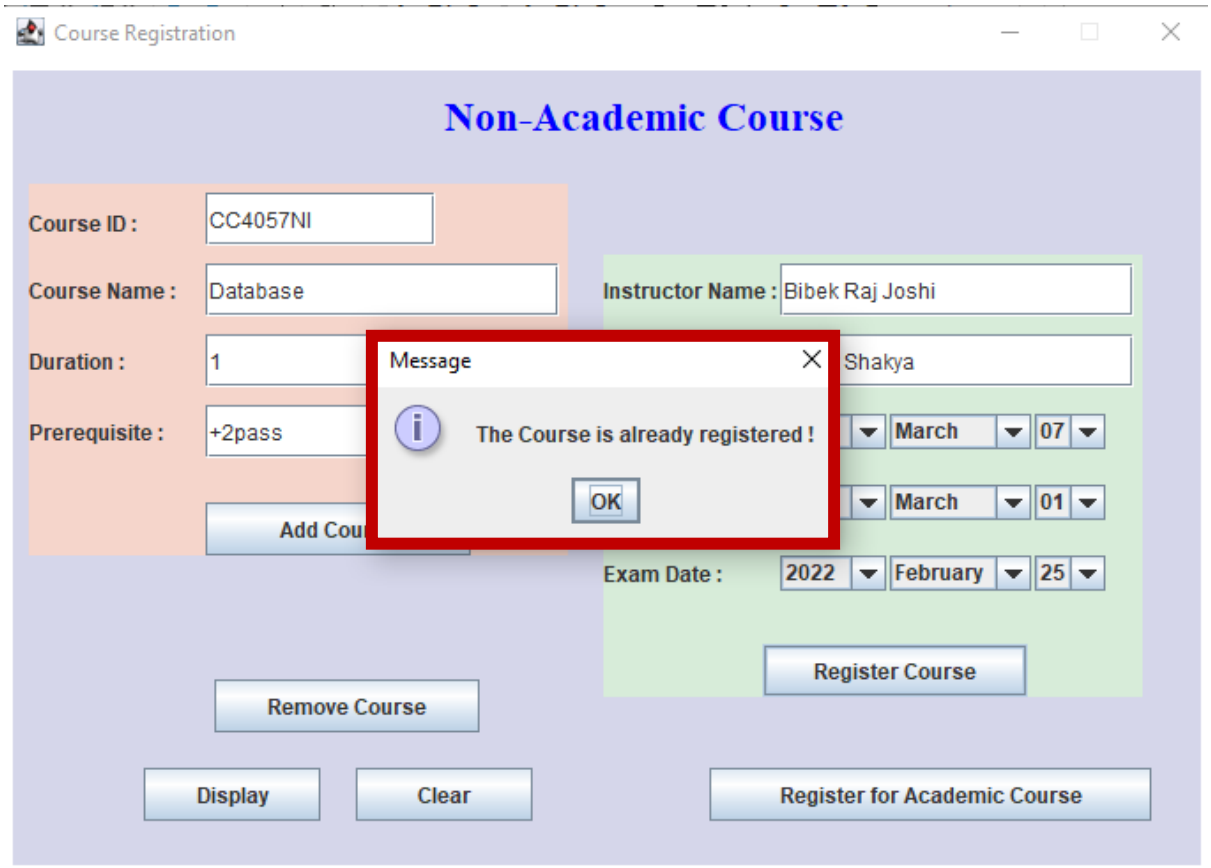


Figure 8: Registering of an already Registered course

c. Trying to remove the non-academic course which is already removed.

Test Number :	3(c)
Objective :	To test the appearance of dialog boxes while trying to remove an already removed course.
Action :	Following actions were performed for this objective ○ Command prompt was opened from the same folder where the code was located ○ Command entered in the command prompt for the compilation of Course class was “javac .java “ ○ Command entered in the command prompt for the compilation of Academic_Course was “javac ING_College.java “ ○ Command entered in the command prompt for the compilation of NonAcademic_Course was “javac ING_College.java “ ○ Command entered in the command prompt for running the program was “ java ING_College “ ○ After the removal of course, the button saying Remove was clicked on again.
Expected Result :	The program would generate a dialog box saying the course has already been removed.
Actual Result :	The program generated a dialog box saying the course is already removed.
Conclusion :	The test is successful.

Table 10: Removal of already removed course

Output of Test 3 (c) :

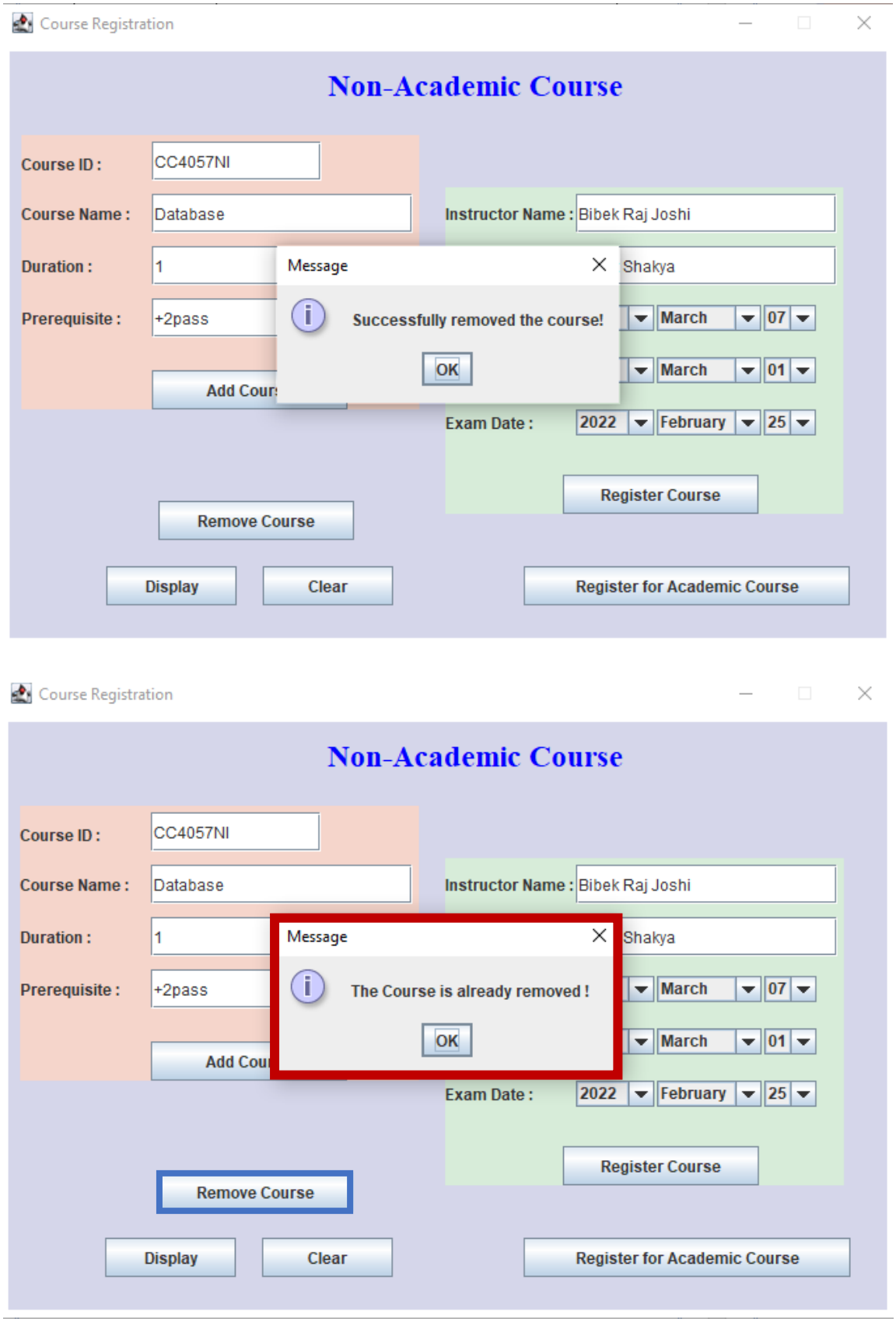


Figure 9: Removal of already removed course

Errors

An error prevents the program from execution and manufacturing the proper output. It may additionally terminate the execution of the program suddenly or even crash the system.

Types of errors:

1. **Syntax errors:** These errors are produced when the syntax of the language is not followed. It is indicated by the compiler.

Syntax Error Detection:

The return type of the method was not specified.

```
//Getters method for CourseID's field
public getCourseID()
{
    return this.id_field.getText();
}
//Getters method for CourseName's field
public String getCourseName()
{
    return this.name_field.getText();
}
```

invalid method declaration; return type required

Figure 10: Syntax Error Detection

Syntax Error Correction:

The error was resolved by appropriately defining the return type for the method. By doing this, the program compiled smoothly.

```
//Getters method for CourseID's field
public String getCourseID()
{
    return this.id_field.getText();
}
//Getters method for CourseName's field
public String getCourseName()
{
    return this.name_field.getText();
}
```

Figure 11: Syntax Error Correction

2. **Semantic errors:** These errors are produced due to improper use of program statements. It is indicated by the compiler.

Semantic Error Detection:

The instance variable was declared with the String datatype but is called with int data type in the getter method.

```
//Getters method for Credit's field
public int getCredit()
{
    return this.credit_field.getText();
}
//Getters method for NumberOfAssessme
public int getNumberOfAssessments()
{
    return Integer.parseInt(this.assessment_field.getText());
}
```

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

Figure 12: Semantic Error Detection

Semantic Error Correction:

The error was resolved by appropriately defining the data type for the instance variable to the String datatype.
By doing this, the program compiled smoothly.

```
//Getters method for Credit's field
public String getCredit()
{
    return this.credit_field.getText();
}
//Getters method for NumberOfAssessments's field
public int getNumberOfAssessments()
{
    return Integer.parseInt(this.assessment_field.getText());
}
```

Figure 13: Semantic Error Correction

3. **Logical errors:** These errors are produced because of the mistakes within the logic of the program. The program is compiled and executed, however; it doesn't generate the requested result.

Logical Error:

The error was found while specifying the messages within the dialog boxes. Instead of showing a message of successful registration of a course at the first call, it showed a different dialog box saying the course is already registered.

It shows clear conflict in the logic of the displayed message.

```
if (getCourseID1().equals(course.getCourseID()))
{
    object_nonacademic= (NonAcademic_Course)course;
    c = 2;
    if(object_nonacademic.getisRegistered()==true)
    {
        JOptionPane.showMessageDialog(frame,"Successfully registered the course! \n\n Course ID: " + getCourseID1() + "\n I
        break;
    }
    else if(object_nonacademic.getisRegistered()==false)
    {
        object_nonacademic.setRegister(getCourse_leader1(), getLecturer_name1() , getStartingDate1(), getCompletionDate1(),
        JOptionPane.showMessageDialog(frame,"The Course is already registered ! ");
        break;
    }
}
```

Figure 14: Logical Error

Logical Error Detection:

Since the program compiled smoothly, the actual detection was done during the runtime i.e., during the registration of Course from the GUI.

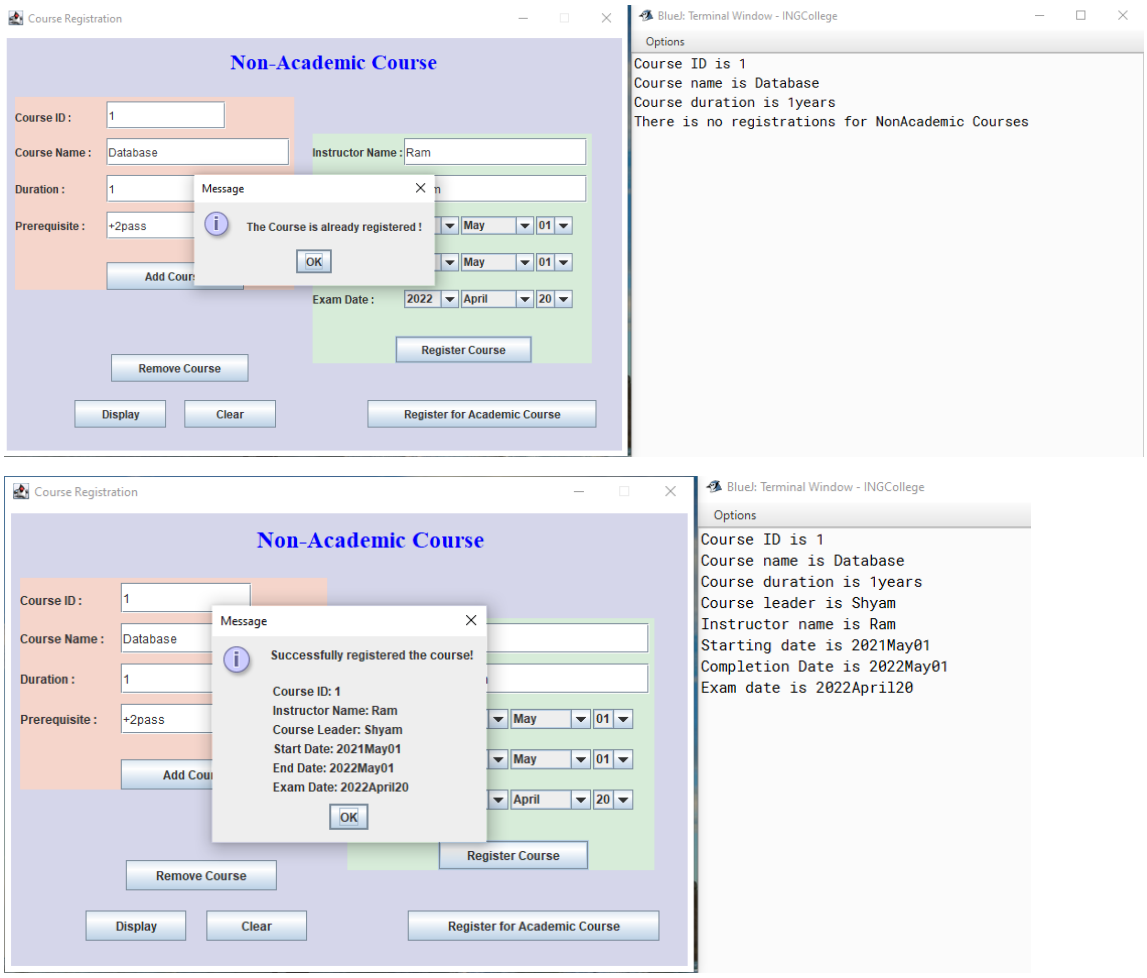


Figure 15: Logical Error Detection

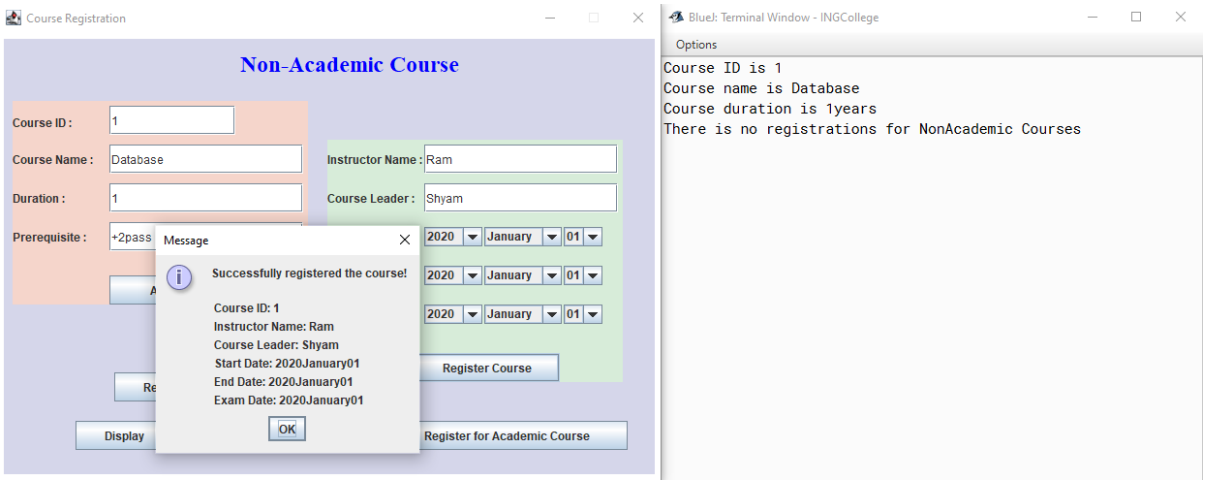
When the user tried registering a course for the first time, the message said that the “Course is already registered” instead of saying “Successfully registered”.

And when the course had already been registered, it displayed the message saying “Successfully registered” instead of saying that the “Course is already registered”

Logical Error Correction:

This conflict in the logic of the code surely put the user in a dilemma and hence the error in the logic of the code was detected and hence corrected.

```
if(course instanceof NonAcademic_Course)
{
    c = 1;
    if (getCourseID1().equals(course.getCourseID()))
    {
        object_nonacademic= (NonAcademic_Course)course;
        c = 2;
        if(object_nonacademic.getisRegistered()==true)
        {
            JOptionPane.showMessageDialog(frame,"The Course is already registered ! ");
            break;
        }
        else if(object_nonacademic.getisRegistered()==false)
        {
            object_nonacademic.setRegister(getCourse_leader1(), getLecturer_name1() , getStartingDate1(), getCompletionDate1()
            JOptionPane.showMessageDialog(frame,"Successfully registered the course! \n\n Course ID: " + getCourseID1() + "\n
            break;
        }
    }
}
```



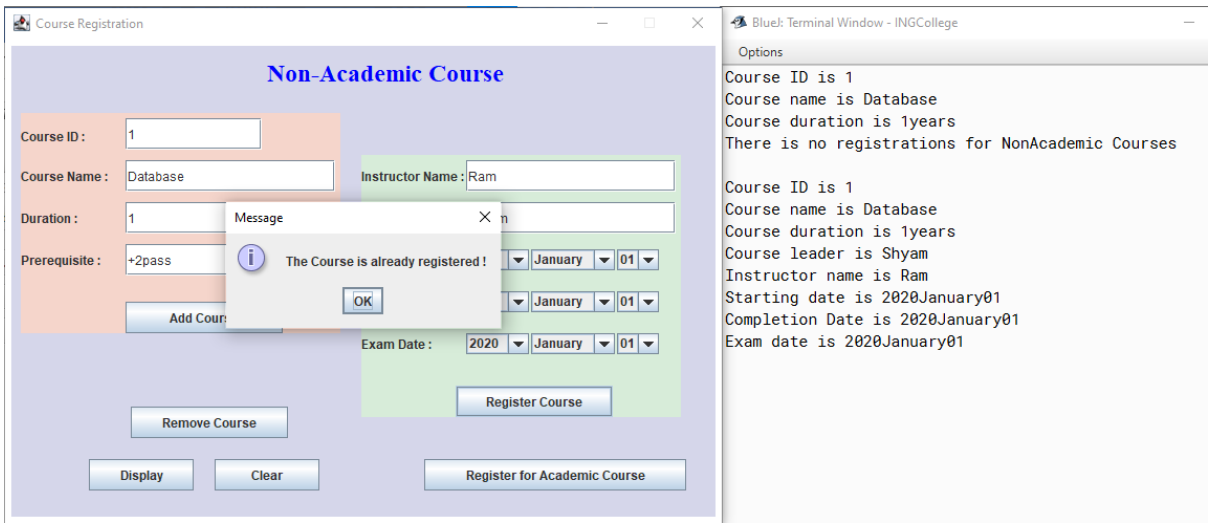


Figure 16: Logical Error Correction

Conclusion

In conclusion, this coursework made us to push ourselves to some of the extremes for its completion. Even though we had learnt the basics and all the concepts during the lectures and tutorial sessions of the module but putting them all together to build up a project was a task.

The building of the graphical user interface involved a bit of creativity and was completed without any such hassle but adding of the functionalities to the GUI was a main task. It had to be done with great precision for the outputs to turn out correct. Dialog boxes were used for displaying the validation of the actions performed, and for the warnings to fill all the fields before trying to add or register a course. Therefore, it was responsible to give all the warnings and confirmations to the action performed by the user on the GUI.

Additionally, I learnt about the types of errors that might occur when coding and how to fix them. I also learnt about various exceptions and ways to handle it with the help of try and catch methods. Furthermore, I learned about the inspection and testing of codes via the command prompt and GUI itself. I learnt to use various components of packages like java.awt and javax.swing to build up a functional GUI.

The coursework was tough to cope up with, but with all the support by the teachers, I have managed to complete it satisfactorily.

Appendix 1

For ING_College.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.border.*;
/**
 * Class ING_College consists of a GUI to input details about the Academic or a
NonAcademic type of Course.
 *
 * @author (Subriti Aryal)
 * @version (5.0.1)
 */
public class ING_College implements ActionListener
{
    // Instance Variable for Frame
    private JFrame frame;

    //Instance Variable for Panels
    private JPanel main_panel,add_panel,register_panel,add1_panel,register1_panel;
```

```
  //Instance Variable for JLabels of Academic Course

    private JLabel
label,id_label,name_label,duration_label,level_label,credit_label,assessment_label,a
ssessment1_label,

    lecturer_label,leader_label,start_label,end_label;


    //Instance Variables for JTextFields of Academic Course

    private JTextField
id_field,name_field,duration_field,level_field,credit_field,assessment_field,

    lecturer_field,leader_field;


    //Instance Variables for JComboBox of Academic Course

    private JComboBox
year_combo,month_combo,day_combo,years_combo,months_combo,days_combo;


    //Instance Variable for JLabels of NonAcademic Course

    private JLabel
id_label1,name_label1,duration_label1,pre_label,lecturer_label1,leader_label1,start_
label1,end_label1,exam_label;


    //Instance Variables for JTextFields of NonAcademic Course

    private JTextField
id_field1,name_field1,duration_field1,pre_field,lecturer_field1,leader_field1;
```

```
    //Instance Variables for JComboBox of NonAcademic Course

    private JComboBox
startyear_combo,startmonth_combo,startday_combo,endyear_combo,endmonth_co
mbo,endday_combo,examyear_combo,exammonth_combo,examday_combo;


    //Instance Variable for Buttons of Academic Course

    private JButton add_button,register_button,clear_button ,
nonacademic_button,display_button;

    //Instance Variable for Buttons of NonAcademic Course

    private JButton
add1_button,register1_button,clear1_button,remove_button,academic_button;


    //objects

    private Academic_Course object_academic;

    private NonAcademic_Course object_nonacademic;


    //declaration of arraylist

    private ArrayList<Course> alist= new ArrayList<Course>();


    public void GUI()

    {

        frame=new JFrame("Course Registration");

        //Jmain_panel

        main_panel= new JPanel();

        main_panel.setLayout(null);
```

```
//setting color using rgb
Color main_color= new Color(213,214,234);
main_panel.setBackground(main_color);

//adding border to the fram
Border border= BorderFactory.createLineBorder(Color.WHITE,6);
main_panel.setBorder(border);

//panel to add academic course
add_panel= new JPanel();
Color add_color= new Color(245,213,203);
add_panel.setBackground(add_color);
add_panel.setBounds(15,70,305,300);
add_panel.setLayout(null);
main_panel.add(add_panel);

//panel to register academic course
register_panel= new JPanel();
Color register_color= new Color(215,236,217);
register_panel.setBackground(register_color);
register_panel.setBounds(340,110,305,205);
register_panel.setLayout(null);
main_panel.add(register_panel);
//panel to add nonacademic course
```

```
add1_panel= new JPanel();
add1_panel.setBackground(add_color);
add1_panel.setBounds(15,70,305,210);
add1_panel.setLayout(null);
main_panel.add(add1_panel);

//panel to register nonacademic course
register1_panel= new JPanel();
register1_panel.setBackground(register_color);
register1_panel.setBounds(340,110,305,250);
register1_panel.setLayout(null);
main_panel.add(register1_panel);

//for text "academic course"
label= new JLabel("Academic Course");
label.setBounds(250,20,400,25);
label.setForeground(Color.blue);

//for font
Font ff= new Font("Times New Roman",Font.BOLD,23);
label.setFont(ff);

//adding label to main_panel
main_panel.add(label);
```

```
//jlabel- Course ID
id_label= new JLabel("Course ID :");
id_label.setBounds(0,10,400,25);
add_panel.add(id_label);
//jtextfield for Course id
id_field= new JTextField();
id_field.setBounds(100,5,130,30);
add_panel.add(id_field);

//jLabel- "Course Name"
name_label= new JLabel("Course Name :");
name_label.setBounds(0,50,120,20);
add_panel.add(name_label);

//jtextfield for Course name
name_field= new JTextField();
name_field.setBounds(100,45,200,30);
add_panel.add(name_field);

//jLabel- "Duration"
duration_label= new JLabel("Duration :");
duration_label.setBounds(0,90,120,20);
add_panel.add(duration_label);
```

```
//jtextfield for duration
duration_field= new JTextField();
duration_field.setBounds(100,85,200,30);
add_panel.add(duration_field);

//jLabel- "level"
level_label= new JLabel("Level :");
level_label.setBounds(0,130,120,20);
add_panel.add(level_label);

//jtextfield for level
level_field= new JTextField();
level_field.setBounds(100,125,200,30);
add_panel.add(level_field);

//jLabel- "credit"
credit_label= new JLabel("Credit :");
credit_label.setBounds(0,170,120,20);
add_panel.add(credit_label);

//jtextfield for credit
credit_field= new JTextField();
credit_field.setBounds(100,165,200,30);
add_panel.add(credit_field);
```

```
 //jLabel- "number of assessment"
assessment_label= new JLabel("No. of ");
assessment_label.setBounds(0,205,120,20);
add_panel.add(assessment_label);
assessment1_label= new JLabel("Assessments :");
assessment1_label.setBounds(0,215,120,20);
add_panel.add(assessment1_label);

//jtextfield for assessment
assessment_field= new JTextField();
assessment_field.setBounds(100,205,200,30);
add_panel.add(assessment_field);
 //Add Course Button for academic course
add_button= new JButton("Add Course");
add_button.setBounds(100,260,150,30);
add_button.addActionListener(this);
add_panel.add(add_button);

//Remove button for non academic course
remove_button= new JButton("Remove Course");
remove_button.setBounds(120,350,150,30);
remove_button.addActionListener(this);
main_panel.add(remove_button);          //adding this to main panel
```

```
//display Button
display_button= new JButton("Display");
display_button.setBounds(80,400,100,30);
display_button.addActionListener(this);
main_panel.add(display_button);            //adding to main panel

//clear Button for academic course
clear_button= new JButton("Clear");
clear_button.setBounds(200,400,100,30);
clear_button.addActionListener(this);
main_panel.add(clear_button);              //adding to main panel

//clear1 Button for non academic course
clear1_button= new JButton("Clear");
clear1_button.setBounds(200,400,100,30);
clear1_button.addActionListener(this);
main_panel.add(clear1_button);

//jLabel- "Lecturer's name"
lecturer_label= new JLabel("Lecturer Name :");
lecturer_label.setBounds(0,10,200,20);
register_panel.add(lecturer_label);

//jtextfield for lecturer name
```

```
lecturer_field= new JTextField();
lecturer_field.setBounds(100,5,200,30);
register_panel.add(lecturer_field);

//jLabel- "course leader"
leader_label= new JLabel("Course Leader :");
leader_label.setBounds(0,50,200,20);
register_panel.add(leader_label);

//jtextfield for course leader
leader_field= new JTextField();
leader_field.setBounds(100,45,200,30);
register_panel.add(leader_field);

//jLabel- "start date"
start_label= new JLabel("Start Date :");
start_label.setBounds(0,90,200,20);
register_panel.add(start_label);

//JComboBox for startdate
String[]months=
{"January","February","March","April","May","June","July","August","September","Oct
ober","November","December"};
```

```
String[]years={"2020","2021","2022","2023","2024","2025","2026","2027","2028","202
9","2030"};

String[]days={"01","02","03","04","05","06","07","08","09","10","11","12","13","14","15"
,"16","17","18","19","20","21","22","23","24","25","26","27","28","29","30","31"};
        year_combo=new JComboBox(years);
        month_combo=new JComboBox(months);
        day_combo=new JComboBox(days);
        year_combo.setBounds(100,90,60,20);
        month_combo.setBounds(162,90,80,20);
        day_combo.setBounds(244,90,40,20);
        register_panel.add(year_combo);
        register_panel.add(month_combo);
        register_panel.add(day_combo);

        //jLabel- "end date"
        end_label= new JLabel("End Date :");
        end_label.setBounds(0,130,200,20);
        register_panel.add(end_label);

        //JComboBox for enddate
        years_combo=new JComboBox(years);
        months_combo=new JComboBox(months);
        days_combo=new JComboBox(days);
```

```
        years_combo.setBounds(100,130,60,20);
        months_combo.setBounds(162,130,80,20);
        days_combo.setBounds(244,130,40,20);
        register_panel.add(years_combo);
        register_panel.add(months_combo);
        register_panel.add(days_combo);

    //jbutton for registering non academic course
        register_button= new JButton("Register Course");
        register_button.setBounds(90,170,150,30);
        register_button.addActionListener(this);
        register_panel.add(register_button);

        //jbutton for nonacademic. -> for switching to another panel
        nonacademic_button= new JButton("Register for Non-Academic Course");
        nonacademic_button.setBounds(400,400,250,30);
        nonacademic_button.addActionListener(this);
        main_panel.add(nonacademic_button);

        //for non academic panel
         //jlabel- Course ID
        id_label1= new JLabel("Course ID :");
        id_label1.setBounds(0,10,400,25);
        add1_panel.add(id_label1);
```

```
        //jtextfield for Course id
        id_field1= new JTextField();
        id_field1.setBounds(100,5,130,30);
        add1_panel.add(id_field1);

        //jLabel- "Course Name"
        name_label1= new JLabel("Course Name :");
        name_label1.setBounds(0,50,120,20);
        add1_panel.add(name_label1);

    //jtextfield for Course name
        name_field1= new JTextField();
        name_field1.setBounds(100,45,200,30);
        add1_panel.add(name_field1);

        //jLabel- "Duration"
        duration_label1= new JLabel("Duration :");
        duration_label1.setBounds(0,90,120,20);
        add1_panel.add(duration_label1);

        //jtextfield for duration
        duration_field1= new JTextField();
        duration_field1.setBounds(100,85,200,30);
        add1_panel.add(duration_field1);
```

```
//jLabel- "prerequisite"
pre_label= new JLabel("Prerequisite :");
pre_label.setBounds(0,130,120,20);
add1_panel.add(pre_label);

//jtextfield for prerequisite
pre_field= new JTextField();
pre_field.setBounds(100,125,200,30);
add1_panel.add(pre_field);

//Add Course Button for Non academic course
add1_button= new JButton("Add Course");
add1_button.setBounds(100,180,150,30);
add1_button.addActionListener(this);
add1_panel.add(add1_button);
//jLabel- "instructor's name"
lecturer_label1= new JLabel("Instructor Name :");
lecturer_label1.setBounds(0,10,200,20);
register1_panel.add(lecturer_label1);

//jtextfield for instructor name
lecturer_field1= new JTextField();
lecturer_field1.setBounds(100,5,200,30);
register1_panel.add(lecturer_field1);
```

```
 //jLabel- "course leader"
leader_label1= new JLabel("Course Leader :");
leader_label1.setBounds(0,50,200,20);
register1_panel.add(leader_label1);

//jtextfield for course leader
leader_field1= new JTextField();
leader_field1.setBounds(100,45,200,30);
register1_panel.add(leader_field1);

//jLabel- "start date"
start_label1= new JLabel("Start Date :");
start_label1.setBounds(0,90,200,20);
register1_panel.add(start_label1);

//JComboBox for startdate
startyear_combo=new JComboBox(years);
startmonth_combo=new JComboBox(months);
startday_combo=new JComboBox(days);
startyear_combo.setBounds(100,90,60,20);
startmonth_combo.setBounds(162,90,80,20);
startday_combo.setBounds(244,90,40,20);
register1_panel.add(startyear_combo);
register1_panel.add(startmonth_combo);
```

```
register1_panel.add(startday_combo);

//jLabel- "end date"
end_label1= new JLabel("End Date :");
end_label1.setBounds(0,130,200,20);
register1_panel.add(end_label1);

//JComboBox for enddate
endyear_combo=new JComboBox(years);
endmonth_combo=new JComboBox(months);
endday_combo=new JComboBox(days);
endyear_combo.setBounds(100,130,60,20);
endmonth_combo.setBounds(162,130,80,20);
endday_combo.setBounds(244,130,40,20);
register1_panel.add(endyear_combo);
register1_panel.add(endmonth_combo);
register1_panel.add(endday_combo);

//jLabel- "exam date"
exam_label= new JLabel("Exam Date :");
exam_label.setBounds(0,170,200,20);
register1_panel.add(exam_label);
```

```
//JComboBox for examdate
  examyear_combo=new JComboBox(years);
  exammonth_combo=new JComboBox(months);
  examday_combo=new JComboBox(days);
  examyear_combo.setBounds(100,170,60,20);
  exammonth_combo.setBounds(162,170,80,20);
  examday_combo.setBounds(244,170,40,20);
  register1_panel.add(examyear_combo);
  register1_panel.add(exammonth_combo);
  register1_panel.add(examday_combo);

  //jbutton for register1 -> non academic register button
  register1_button= new JButton("Register Course");
  register1_button.setBounds(90,220,150,30);
  register1_button.addActionListener(this);
  register1_panel.add(register1_button);

  //jbutton for academic -> switches to Academic Course panel
  academic_button= new JButton("Register for Academic Course");
  academic_button.setBounds(400,400,250,30);
  academic_button.addActionListener(this);
  main_panel.add(academic_button);
```

```
        //adding main_panel to frame
        frame.add(main_panel);

        // true for the Academic course panels
        add_panel.setVisible(true);
        register_panel.setVisible(true);
        nonacademic_button.setVisible(true); //button to switch to Non-Academic
        clear_button.setVisible(true);

          //false for the Non Academic course panels
        add1_panel.setVisible(false);
        register1_panel.setVisible(false);

        remove_button.setVisible(false);
        clear1_button.setVisible(false);
        academic_button.setVisible(false);  //button to switch back to Academic

        //Setting bounds, resizability and visibility of frame
        frame.setBounds(300,100,700,500);
        frame.setResizable(false);
        frame.setVisible(true);
    }
```

```
  //Main Method
   public static void main(String[]args)
   {
       ING_College ing= new ING_College();
       ing.GUI();
   }

   //Getters  method for fields in Academic_Course
   //Getters  method for CourseID's field
   public String getCourseID()
   {
       return this.id_field.getText();
   }
   //Getters  method for CourseName's field
   public String getCourseName()
   {
       return this.name_field.getText();
   }
   //Getters  method for Duration's field
   public int getDuration()
   {
       return Integer.parseInt(this.duration_field.getText());
   }
   //Getters  method for Level's field
```

```
    public String getLevel()
    {
        return this.level_field.getText();
    }
    //Getters  method for Credit's field
    public String getCredit()
    {
        return this.credit_field.getText();
    }
    //Getters  method for NumberOfAssessments's field
    public int getNumberOfAssessments()
    {
        return Integer.parseInt(this.assessment_field.getText());
    }
    //Getters  method for Courseleader's field
    public String getCourse_leader()
    {
        return this.leader_field.getText();
    }
    //Getters  method for Lecturername's field
    public String getLecturer_name()
    {
        return this.lecturer_field.getText();
    }
```

```
//Getters method for StartingDate's combobox
public String getStartingDate()
{
    String year=(year_combo.getSelectedItem()).toString();
    String month= (month_combo.getSelectedItem()).toString();
    String day=(day_combo.getSelectedItem()).toString();
    return (year+""+month+""+day);
}
//Getters method for CompletionDate's combobox
public String getCompletionDate()
{
    String years=(years_combo.getSelectedItem()).toString();
    String months= (months_combo.getSelectedItem()).toString();
    String days=(days_combo.getSelectedItem()).toString();
    return (years+""+months+""+days);
}
//Getters method for fields in NonAcademic_Course
//Getters method for CourseID's field
public String getCourseID1()
{
    return this.id_field1.getText();
}
//Getters method for CourseName's field
public String getCourseName1()
```

```
{
    return this.name_field1.getText();
}
//Getters  method for Duration's field
public int getDuration1()
{
    return Integer.parseInt(this.duration_field1.getText());
}
//Getters  method for Prerequisite's field
public String getPrerequisite()
{
    return this.pre_field.getText();
}
//Getters  method for Courseleader's field
public String getCourse_leader1()
{
    return this.leader_field1.getText();
}
//Getters  method for Lecturername's field
public String getLecturer_name1()
{
    return this.lecturer_field1.getText();
}
```

```
//Getters  method for StartingDate's combobox
 public String getStartingDate1()
 {
     String startyear=(startyear_combo.getSelectedItem()).toString();
     String startmonth= (startmonth_combo.getSelectedItem()).toString();
     String startday=(startday_combo.getSelectedItem()).toString();
     return (startyear+startmonth+startday);
 }
 //Getters  method for CompletionDate's combobox
 public String getCompletionDate1()
 {
     String endyear=(endyear_combo.getSelectedItem()).toString();
     String endmonth= (endmonth_combo.getSelectedItem()).toString();
     String endday=(endday_combo.getSelectedItem()).toString();
     return (endyear+endmonth+endday);
 }
 //Getters  method for ExamDate's combobox
 public String getExamDate()
 {
     String examyear=(examyear_combo.getSelectedItem()).toString();
     String exammonth= (exammonth_combo.getSelectedItem()).toString();
     String examday=(examday_combo.getSelectedItem()).toString();
     return (examyear+exammonth+examday);
 }
```

```
//Method where the functionalities of the buttons are specified
public void actionPerformed(ActionEvent e)
{
    /*The functionality for the button named "Register for Non-Academic Course"
    All the specified changes would take place on the press of this button
    The text or the heading of the form would change, likewise the buttons and
    panels would be adjusted accordingly*/

    if(e.getSource() == nonacademic_button)
    {
        add1_panel.setVisible(true);
        register1_panel.setVisible(true);

        label.setText("Non-Academic Course");
        remove_button.setVisible(true);
        academic_button.setVisible(true);
        clear_button.setVisible(false);
        clear1_button.setVisible(true);
        nonacademic_button.setVisible(false);

        add_panel.setVisible(false);
        register_panel.setVisible(false);
    }
```

```
/*The functionality for the button named "Register for Academic Course"
All the specified changes would take place on the press of this button
The text or the heading of the form would change, likewise the buttons and
panels would be adjusted accordingly */

else if(e.getSource() == academic_button)
{
    add_panel.setVisible(true);
    register_panel.setVisible(true);

    label.setText("Academic Course");
    remove_button.setVisible(false);
    nonacademic_button.setVisible(true);
    academic_button.setVisible(false);
    clear_button.setVisible(true);
    clear1_button.setVisible(false);

    add1_panel.setVisible(false);
    register1_panel.setVisible(false);
}

/* The function for the button Add is to get the input and
 * create a new object of Academic_Course type and later
 * add the object to an arraylist of Course class*/
```

```
 //For "Add Course" of Academic Course


else if(e.getSource() == add_button)
{
    boolean academic_add=false;
    int ct = 0;
    for (Course course: alist)
    {
        if(course.getCourseID().equals(getCourseID()))
        {
            academic_add=true;
            break;
        }
    }
    if (getCourseID().isEmpty() || getCourseName().isEmpty() || getLevel().isEmpty() || getCredit().isEmpty())
    {
        JOptionPane.showMessageDialog(frame, "Please fill in all the fields correctly","Error", JOptionPane.WARNING_MESSAGE);
        ct = 1;
    }
    if (ct == 0)
    {
        try
        {
```

```
                getDuration();
                try
                {
                    getNumberOfAssessments();
                }
                catch(NumberFormatException ex)
                {
                    JOptionPane.showMessageDialog(frame,"Enter a number in the text
field of No. of assessments","Alert",JOptionPane.WARNING_MESSAGE);
                    ct = 1;
                }
            }
            catch(NumberFormatException ex)
            {
                JOptionPane.showMessageDialog(frame,"Enter a number in the text
field of duration","Alert",JOptionPane.WARNING_MESSAGE);
                ct = 1;
            }
        }
        if (academic_add==true)
        {
            JOptionPane.showMessageDialog(frame,"The Course having Course ID :
" + getCourseID() + " is already added", "Error", JOptionPane.ERROR_MESSAGE);
        }
        else if (ct == 0)
```

```
            {
                //Calling the constructor of Academic Course class
                object_academic= new
Academic_Course(getCourseID(),getCourseName(),getDuration(),getLevel(),getCre
dit(),getNumberOfAssessments());
                alist.add(object_academic);
                JOptionPane.showMessageDialog(frame,"Successfully added to ArrayList
! \n\n Course ID: " +getCourseID() + "\n Course Name: " + getCourseName() + "\n
Duration: " + getDuration() + "\n Level: " +getLevel() + "\n Credit: " + getCredit() + "\n
Number of Assessments: " + getNumberOfAssessments());
            }
        }

        /* The function for the button Add is to get the input and
         * create a new object of NonAcademic_Course type and later
         * add the object to an arraylist of Course class*/

        //For "Add Course" of NonAcademic Course

        else if(e.getSource() == add1_button)
        {
            boolean nonacademic_add=false;
            int ct = 0;
            for (Course course: alist)
            {
                if(course.getCourseID().equals(getCourseID1()))
```

```
            {
                nonacademic_add=true;
                break;
            }
        }
        if (getCourseID1().isEmpty() || getCourseName1().isEmpty() || getPrerequisite().isEmpty())
        {
            JOptionPane.showMessageDialog(frame, "Please fill in all the fields correctly","Error", JOptionPane.WARNING_MESSAGE);
            ct = 1;
        }
        if (ct == 0)
        {
            try
            {
                getDuration1();
            }
            catch(NumberFormatException ex)
            {
                JOptionPane.showMessageDialog(frame,"Enter a number in the text field of duration","Alert",JOptionPane.WARNING_MESSAGE);
                ct = 1;
            }
        }
```

```
        if(nonacademic_add==true)
        {
            JOptionPane.showMessageDialog(frame,"The Course having Course ID : " + getCourseID1() + " is already added", "Error", JOptionPane.ERROR_MESSAGE);
        }
        else if(ct == 0)
        {
            //Calling the constructor of NonAcademic Course class
            object_nonacademic= new NonAcademic_Course(getCourseID1(),getCourseName1(),getDuration1(),getPrerequisite());
            alist.add(object_nonacademic);
            JOptionPane.showMessageDialog(frame,"Successfully added to ArrayList ! \n\n Course ID: " +getCourseID1() + "\n Course Name: " + getCourseName1() + "\n Duration: " + getDuration1() + "\n Prerequisite: " +getPrerequisite());
        }
    }
```

```
/* The function for the button Register is to get the input value of CourseID
* and compare with existing Course ID and if found valid, registers academic
course
* this is done by calling the method to register from the Academic_Course
class */


//For "Register Course" of Academic Course


else if(e.getSource() == register_button)
{
    boolean counter=false;
    int c=0;
    if (getCourseID().isEmpty() || getCourse_leader().isEmpty() ||
getLecturer_name().isEmpty() || getStartingDate().isEmpty() ||
getCompletionDate().isEmpty())
    {
        JOptionPane.showMessageDialog(frame, "Please fill in all the fields
correctly","Error", JOptionPane.WARNING_MESSAGE);
        counter=true;
    }
    if (counter==false)
    {
        for (Course course: alist)
        {
            if (course instanceof Academic_Course)
            {
```

```
                    c = 1;
                    if(course.getCourseID().equals(getCourseID()))
                    {
                        object_academic= (Academic_Course)course;
                        c = 2;
                        if(object_academic.getisRegistered()==true)
                        {
                            JOptionPane.showMessageDialog(frame,"The Course is already
registered ! \n\n Lecturer Name: "+ getLecturer_name() + "\n Start Date: "
+getStartingDate() + "\n End Date: " + getCompletionDate());
                            break;
                        }
                        else if(object_academic.getisRegistered()==false)
                        {
                            object_academic.setRegister(getCourse_leader(),
getLecturer_name(), getStartingDate(), getCompletionDate());
                            JOptionPane.showMessageDialog(frame,"Successfully
registered the course! \n\n Course ID: " + getCourseID() + "\n Lecturer Name: "+
getLecturer_name() + "\n Start Date: " +getStartingDate() + "\n End Date: " +
getCompletionDate());
                            break;
                        }
                    }
                }
            }
            if(c == 1)
```

```
            {
                JOptionPane.showMessageDialog(frame, "Please fill in the CourseID
correctly","Error", JOptionPane.ERROR_MESSAGE);


            }
            else if (c == 0)
            {
                JOptionPane.showMessageDialog(frame, "No Academic Course has
been added yet.","Error", JOptionPane.ERROR_MESSAGE);


            }
        }
    }


        /* The function for the button Register is to get the input value of CourseID
            * and compare with existing Course ID and if found valid, registers
        nonacademic course
     * this is done by calling the method to register from the NonAcademic_Course
class */


    //For "Register Course" of NonAcademic Course


    else if(e.getSource() == register1_button)
    {
        boolean counter=false;
```

```
        int c=0;
        if (getCourseID1().isEmpty() || getCourse_leader1().isEmpty() || getLecturer_name1().isEmpty())
        {
            JOptionPane.showMessageDialog(frame, "Please fill in all the fields correctly","Error", JOptionPane.WARNING_MESSAGE);
            counter=true;
        }
        if(counter==false)
        {
            for (Course course: alist)
            {
                if(course instanceof NonAcademic_Course)
                {
                    c = 1;
                    if (getCourseID1().equals(course.getCourseID()))
                    {
                        object_nonacademic= (NonAcademic_Course)course;
                        c = 2;
                        if(object_nonacademic.getisRegistered()==true)
                        {
                            JOptionPane.showMessageDialog(frame,"The Course is already registered ! ");
                            break;
                        }
```

```
                    else if(object_nonacademic.getisRegistered()==false)
                    {
                        object_nonacademic.setRegister(getCourse_leader1(),
getLecturer_name1() , getStartingDate1(), getCompletionDate1(), getExamDate());
                        JOptionPane.showMessageDialog(frame,"Successfully
registered the course! \n\n Course ID: " + getCourseID1() + "\n Instructor Name: " +
getLecturer_name1() + "\n Course Leader: " + getCourse_leader1() + "\n Start Date:
" +getStartingDate1() + "\n End Date: " + getCompletionDate1() + "\n Exam Date: " +
getExamDate());
                        break;
                    }
                }
            }
            }
            if (c == 1)
            {
                JOptionPane.showMessageDialog(frame, "Please fill in the CourseID
correctly","Error", JOptionPane.ERROR_MESSAGE);
            }
            else if (c == 0)
            {
                JOptionPane.showMessageDialog(frame, "No Non-Academic Course
has been added yet.","Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    }
```

```
/*Functionality for the button named "Display".
It is responsible for displaying the details of the course by
checking the instance and calling the class's method accordingly*/

else if(e.getSource()==display_button)
{
    if(getCourseID().isEmpty() && getCourseID1().isEmpty())
    {
        id_field.setText("1");
        id_field1.setText("1");
        JOptionPane.showMessageDialog(frame,"No Course to display ! ","Error",
JOptionPane.ERROR_MESSAGE);
    }
    String getLabel= label.getText();
    for (Course course: alist)
    {
        //For Academic Course
        if(getLabel.equals("Academic Course"))
        {
            if (course instanceof Academic_Course)
            {
                object_academic= (Academic_Course)course;
                object_academic.display();
                System.out.println("");
            }
```

```
            }
            //For NonAcademic Course
            else if(getLabel.equals("Non-Academic Course"))
            {
                if (course instanceof NonAcademic_Course)
                {
                    object_nonacademic= (NonAcademic_Course)course;
                    object_nonacademic.display();
                    System.out.println("");
                }
            }
        }
    }

    /*Functionality for the button named "Clear".
     * It clears all the field values and JComboBox input of the frame. */

    //For Academic Course
    else if(e.getSource()== clear_button)
    {
        id_field.setText("");
        name_field.setText("");
        duration_field.setText("");
        level_field.setText("");
```

```
            credit_field.setText("");
            assessment_field.setText("");
            leader_field.setText("");
            lecturer_field.setText("");
            year_combo.setSelectedIndex(0);
            month_combo.setSelectedIndex(0);
            day_combo.setSelectedIndex(0);
            years_combo.setSelectedIndex(0);
            months_combo.setSelectedIndex(0);
            days_combo.setSelectedIndex(0);
        }

  //For NonAcademic Course
        else if(e.getSource()== clear1_button)
        {
            id_field1.setText("");
            name_field1.setText("");
            duration_field1.setText("");
            pre_field.setText("");
            lecturer_field1.setText("");
            leader_field1.setText("");
            startyear_combo.setSelectedIndex(0);
            startmonth_combo.setSelectedIndex(0);
            startday_combo.setSelectedIndex(0);
```

```
        endyear_combo.setSelectedIndex(0);
        endmonth_combo.setSelectedIndex(0);
        endday_combo.setSelectedIndex(0);
        examyear_combo.setSelectedIndex(0);
        exammonth_combo.setSelectedIndex(0);
        examday_combo.setSelectedIndex(0);
    }

    /* Functionality for the button named "Remove" in NonAcademic Course.
    It is responsible for removing the registration of a particular course.
     It does so by checking the instance and calling the method to remove*/

    else if(e.getSource()== remove_button)
    {
        for(Course course: alist)
        {
            if(course.getCourseID().equals (getCourseID1()))
            {
                if (course instanceof NonAcademic_Course)
                {
                    object_nonacademic= (NonAcademic_Course) course;
                    if(object_nonacademic.getisRemoved()==true)
                    {
                        JOptionPane.showMessageDialog(frame,"The Course is already
removed ! ");
```

```
                    break;
                }
                else if(object_nonacademic.getisRemoved()==false)
                {
                    object_nonacademic.Remove();
                    JOptionPane.showMessageDialog(frame,"Successfully removed
the course!");
                    break;
                }
            }
        }
        else
        {
            JOptionPane.showMessageDialog(frame, "Please input valid
CourseID","Error", JOptionPane.ERROR_MESSAGE);
        }
    }
    if (getCourseID1().isEmpty())
    {
        JOptionPane.showMessageDialog(frame, "Please input the CourseID for
the course you would like to remove","Error", JOptionPane.ERROR_MESSAGE);
    }
  }
 }
}
```

Appendix 2

For Course.java

```
/**
 * The class Course is the parent class of two child classes namely
Academic_Course and NonAcademic_Course
 * It consists of basic information regarding a particular course.
 *
 * @author (Subriti Aryal)
 * @version (11.0.2)
 */

public class Course
{
    //Four instance variables are created with appropriate datatypes: CourseID,
Coursename, Duration and CourseLeader

    private String CourseID;
    private String Coursename;
    protected int Duration;
    private String CourseLeader;
```

```
/* A constructor is created with 4 parameters: Course_ID, Course_name and
duration
 * Constructors are responsible for assigning values to the instance variable
 * Here, the CourseLeader is initialized with an empty string (“”)
 */
Course(String Course_ID, String Course_name, int duration)
{
    this.CourseID=Course_ID;
    this.Coursename=Course_name;
    this.Duration=duration;
    this.CourseLeader=" ";
}
//Accessor methods:  Used to get the initialised value
public String getCourseID()
{
    return CourseID;
 }
  public String getCoursename()
{
    return Coursename;
}
public int getDuration()
{
    return Duration;
}
```

```
    public String getCourseLeader()
    {
        return CourseLeader;
    }

    //Mutator methods: Used to set some value in an instance variable
(CourseLeader)
    public void setCourseLeader(String Course_leader)
    {
        this.CourseLeader= Course_leader;
    }

    //Display method: Used to display the information about the course.

    public void display()
    {
       System.out.println("Course ID is "+ CourseID);
       System.out.println("Course name is "+ Coursename);
       System.out.println("Course duration is "+ Duration +"years");
       if(CourseLeader!=" ")
       {
           System.out.println("Course leader is "+ CourseLeader);
        }
    }
}
```

For Academic_Course. Java

```
/**
 * The class Academic_Course is a child class of the class "Course"
 * It consists of a detailed information regarding an academic course.
 * @author (Subriti Aryal)
 * @version (11.0.2)
 */

// Child class inherits the properties of its parent class "Course"

public class Academic_Course extends Course
{
    /* Seven instance variables are created with their appropriate datatypes:
     * Lecturername, Level, Credit, StartingDate, CompletionDate,
NumberOfAssignments and isRegistered */
    private String CourseID;
    private String Coursename;
    private int Duration;
    private String Lecturername;
    private String Level;
    private String Credit;
    private String StartingDate;
    private String CompletionDate;
    private int NumberOfAssessments;
```

```
    private boolean isRegistered;


    /*
     * A constructor is created with 6 parameters: CourseID, Coursename, Duration,
level, credit and numberOfAssessments
     * Constructors are responsible for assigning values to the instance variable
     * CourseID, Coursename and Duration are called from the superclass using the
"super" keyword
     * While, the Lecturername,StartingDate and CompletionDate is initialized with an
empty string (“”) and isRegistered status is initialized to false.
     */


    Academic_Course( String CourseID, String Coursename, int Duration, String
level, String credit, int numberOfAssessments)
    {
        super(CourseID,Coursename,Duration);
        this.Level= level;
        this.Credit= credit;
        this.NumberOfAssessments= numberOfAssessments;
        this.Lecturername=" ";
        this.StartingDate= " ";
        this.CompletionDate=" ";
        this.isRegistered= false;
    }
```

```
//Accessor method: Used to get the initialised value

public String getLecturername()
{
    return Lecturername;
}

public String getLevel()
{
    return Level;
}

public String getCredit()
{
    return Credit;
}

public String getStartingDate()
{
    return StartingDate;
}

public String getCompletionDate()
{
```

```
        return CompletionDate;
    }

    public int getNumberofAssessments()
    {
        return NumberOfAssessments;
    }

    public boolean getisRegistered()
    {
        return isRegistered;
    }

    //Mutator methods: Used to set some value in an instance variable (Lecturername
and NumberOfAssessments)

    public void setLecturername(String Lecturer_name)
    {
        this.Lecturername= Lecturer_name;
    }

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

```
    /* Register method: Used to register a particular academic course

        It accepts 4 parameters: Course_leader, Lecturer_name, Starting_date and
Completion_date */


    public void setRegister(String Course_leader, String Lecturer_name, String
Starting_date, String Completion_date)

    {

        //Condition to be processed if the course is already registered


        if(isRegistered== true)

        {

            System.out.println("Lecturer name is "+ Lecturername);

            System.out.println("Starting Date is "+ StartingDate);

            System.out.println("Completion Date is "+ CompletionDate);

        }


        //Condition to be processed if the course is not registered

        else

        {

            super.setCourseLeader(Course_leader);     //method to set the Courseleader;
which is called from the parent class

            this.Lecturername=Lecturer_name;              //initialising values

            this.StartingDate=Starting_date;

            this.CompletionDate=Completion_date;
```

```
            isRegistered= true;                    //default initialisation was false when there
was no any course registered
        }
    }

    //Display method: Used to display the information about any registered course.

    public void display()
    {
        //Display method called from the superclass "Course"
        super.display();

        if(isRegistered==true)
        {
            System.out.println("Lecturer name is "+ Lecturername);
            System.out.println("Level is "+ Level);
            System.out.println("Credit is "+ Credit);
            System.out.println("Starting Date is "+ StartingDate);
            System.out.println("Completion Date is "+ CompletionDate);
            System.out.println("Number of Assessments is "+ NumberOfAssessments);
        }
    }
}
```

For NonAcademic_Course.java

```
/**
 * The class NonAcademic_Course is a child class of the class "Course"
 * It consists of a detailed information regarding a non-academic course.
 * @author (Subriti Aryal)
 * @version (11.0.2)
 */

// Child class inherits the properties of its parent class "Course"

public class NonAcademic_Course extends Course
{
    /* Seven instance variables are created with their appropriate datatypes:
     * Instructorname, StartDate, CompletionDate, ExamDate, Prerequisite,
isRegistered and isRemoved */

    private String Instructorname;
    private String StartDate;
    private String CompletionDate;
    private String ExamDate;
    private String Prerequisite;
    private boolean isRegistered;
    private boolean isRemoved;
```

```
    /*
     * A constructor is created with 4 parameters: CourseID, Coursename, Duration
and prerequisite
     * Constructors are responsible for assigning values to the instance variable
     * CourseID, Coursename and Duration are called from the superclass using the
"super" keyword
     * Here, the StartDate, CompletionDate and ExamDate is initialized with an empty
string (“”)
     * isRegistered and isRemoved status is initialized to false.
     */

    NonAcademic_Course(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: Used to get the initialised value
```

```
public String getInstructorname()
{
    return Instructorname;
}

public String getStartDate()
{
    return StartDate;
}

public String getCompletionDate()
{
    return CompletionDate;
}

public String getExamDate()
{
    return ExamDate;
}

public String getPrerequisite()
{
    return Prerequisite;
}
```

```
 public Boolean getisRegistered()
{
    return isRegistered;
}

public Boolean getisRemoved()
{
    return isRemoved;
}

//Mutator methods: Used to set some value in an instance variable
(Instructorname)

public void setInstructorname( String Instructor_name)
{
    if (isRegistered== true)
    {
        System.out.println("Changing of instructor's name is not possible");
    }
    else
    {
         this.Instructorname=Instructor_name;
    }
}
//Register method: Used to register a particular non-academic course
```

```
        // It accepts 5 parameters: CourseLeader, Instructor_name, Start_date,
Completion_date and Exam_Date


    public void setRegister(String Course_leader, String Instructor_name, String
Start_Date, String Completion_Date, String Exam_Date)
    {
        //Condition to be processed if the course is already registered


        if(isRegistered== true)
        {
            System.out.println("The Course is already registered");
        }


        //Condition to be processed if the course is not registered
        else
        {
            setInstructorname(Instructor_name);
            super.setCourseLeader(Course_leader);  //method to set the CourseLeader;
which is called from the parent class
            this.StartDate= Start_Date;
            this.CompletionDate=Completion_Date;
            this.ExamDate=Exam_Date;
            isRegistered= true;
        }
    }
```

```
        //Remove method: Used to remove a particular non-academic course


    public void Remove()
    {
        if(isRemoved== true)
        {
            System.out.println("The course is already removed");
        }
        else
        {
            super.setCourseLeader(" ");                //method to set the CourseLeader;
which is called from the parent class
            this.Instructorname= " ";
            this.StartDate= " ";
            this.CompletionDate= " ";
            this.ExamDate= " ";
            this.isRegistered= false;
            this.isRemoved= true;
        }
    }
```

```
    //Display method: Used to display the information about any registered course.


    public void display()
    {
        //Display method called from the superclass "Course"
        super.display();


        if( isRegistered== true)
        {
            System.out.println("Instructor name is "+ Instructorname);
            System.out.println("Starting date is "+ StartDate);
            System.out.println("Completion Date is "+ CompletionDate);
            System.out.println("Exam date is "+ ExamDate);
        }
        else
        {
            System.out.println("There is no registrations for NonAcademic Courses");
        }
    }
}
```