LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CC4001NI Programming

COURSEWORK-2

Assessment Weightage & Type

30% Individual Coursework

Semester and Year

Spring 2021

Student Name: Rabina Shrestha

Group: C13

London Met ID: 20049416

College ID: NP01CP4S210039

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

Contents

List of Figures

List of Tables

1.Introduction.

1.1 About the Coursework.

One of the modules that we, the Computing students' study in Year 1 is "Programming". The first part of the coursework given in the 8th week is where we had to create a base in order to learn the proper implementation of object-oriented concept of Java while facing practical problems in the real world. The second part of the coursework which was the assessment given to us in the 20th week, accounts for 30% of the overall module. It contains the task to create a graphical user interface which is also known as GUI, for the system that stores the details about the parent class Course along its' two child classes: Academic Course and Non-Academic Course.

The first half of the project created in BlueJ which consists of three classes namely, Course, Academic Course, Non-Academic Course. For the second half, a new class was introduced whose name is INGCollege. This class consists of the GUI code, along with a main method, getters method, and actionPerformed. Both the projects will be inspected using the command prompt.

1.2 Tools used.

BlueJ:

BlueJ is an easy-to-use development program precisely intended for beginners like college students to develop small-scale Java projects. It also helps the novice programmers like myself adapt into the world of IDE. It is highly interactive as it allows us to inspect the objects values, interact with it, pass them as parameters and also call

methods on them. Not only interactive, but it is also simple and innovative as we can see several features that have not been introduced before. (Kölling & Institute, 1999)

Microsoft Word:

Microsoft Word also known as MS Word is mostly used to create documents. It is a popular word processing program and is used all over the world by businesses, both large and small, by students for their assessments. It is also used for creating certificates, letters and many more. Some of the features provided by MS Word are References where you can cite where you got the idea from, thesaurus from which you can find synonyms and antonyms for a specific word, it also provides autocorrect which can be used to correct any typos you have made. (Somani, 2021)

Draw.io:

Draw.io is a closed source software used for making charts and diagrams. It contains a large selection of different shapes and visual elements which helps our diagram to be one of a kind. One can also export the diagram in a file format that they prefer. The diagram can be dragged across the page and can be decorated with colours which helps the user create the aesthetic they want their diagram to have. (Computer Hope, 2020)

2.Class Diagram

```
ING College

- list_Course : ArrayList<Course>
- AcademicObject : AcademicCourse
- NonAcademicObject : NonAcademicCourse

- frame : JFrame
- panel : JPanel
- panel_Academic : JPanel
- panel_addAC : JPanel
- panel_registerAC : JPanel
- panel_NonAcademic : JPanel
- panel_addNAC : JPanel
- panel_registerNAC : JPanel
- label_CourseForm : JLabel
- button_clickAC : JButton
- button_clickNAC : JButton

- label_AcademicCourse : JLabel
- label_CourseID : JLabel
- label_CourseName : JLabel
- label_Duration : JLabel
- label_Level : JLabel
- label_Credit : JLabel
- label_NoOfAssessment : JLabel
- label_ACregisterID : JLabel
- label_CourseLeader : JLabel
- label_LecturerName : JLabel
- label_ACStartDate : JLabel
- label_ACEndDate : JLabel
- field_ID : JTextField
- field_CName : JTextField
- field_Duration : JTextField
- field_Level : JTextField
- field_Credit : JTextField
- field_NoOfAssessment : JTextField
- field_ACregisterID : JTextField
- field_ACLeader : JTextField
- field_LName : JTextField
```

```
- box_ACYear1 : JComboBox
- box_ACMonth1 : JComboBox
- box_ACDay1 : JComboBox
- box_ACYear2 : JComboBox
- box_ACMonth2 : JComboBox
- box_ACDay2 : JComboBox
- button_addAC : JButton
- button_registerAC : JButton
- button_Display : JButton
- ACStartingDate : String
- ACCompletionDate : String

- label_NonAcademicCourse : JLabel
- label_NACourseID : JLabel
- label_NACourseName : JLabel
- label_NADuration : JLabel
- label_Prerequisite : JLabel
- label_NACregisterID : JLabel
- label_NACourseLeader : JLabel
- label_InstructorName : JLabel
- label_NACStartDate : JLabel
- label_NACEndDate : JLabel
- label_ExamDate : JLabel
- fieldNA_ID : JTextField
- fieldNA_CName : JTextField
- field_NADuration : JTextField
- field_Prerequisite : JTextField
- field_NACregisterID : JTextField
- field_NACLeader : JTextField
- field_IName : JTextField
- box_NACYear1 : JComboBox
- box_NACMonth1 : JComboBox
- box_NACDay1 : JComboBox
- box_NACYear2 : JComboBox
- box_NACMonth2 : JComboBox
- box_NACDay2 : JComboBox
- box_NACYear3 : JComboBox
- box_NACMonth3 : JComboBox
- box_NACDay3 : JComboBox
```

```
- button_addNAC : JButton
- button_RemoveNAC : JButton
- button_registerNAC : JButton
- button_NADisplay : JButton
- NACStartingDate : String
- NACCompletionDate : String
- ExamDate : String
- button_Clear : JButton
  main : INGCollege
```

```
+ gUI() : void
+ main(args : String[]) : void

+ getACourseID() : String
+ getACourseName() : String
+ getACDuration() : int
+ getLevel() : String
+ getCredit() : String
+ getNoOfAssessment() : int
+ getACourseRegisterID() : String
+ getACourseLeader() : String
+ getACLecturerName() : String
+ getACStartDate() : String
+ getACCompleteDate() : String

+ getNACourseID() : String
+ getNACourseName() : String
+ getNACDuration() : int
+ getPrerequisite() : String
+ getNACourseRegisterID() : String
+ getNACourseLeader() : String
+ getInstructorName() : String
+ getNACStartDate() : String
+ getNACCompleteDate() : String
+ getExamDate() : String

+ actionPerformed(e : ActionEvent) : void
```

Figure 1: Class Diagram of INGCollege.

3.Pseudocode.

START

CREATE INGCollege **IMPLEMENTS** ActionListener.

DECLARE instance variable as follows:

INITIALIZE ArrayList to list_Course.
INITIALIZE AcademicCourse to AcademicObject and NonAcademicCourse to NonAcademicObject.

INITIALIZE JFrame to frame.
INITIALIZE JPanel to panel, panel_Academic, panel_addAC, panel_registerAC, panel_NonAcademic, panel_addNAC, and panel_registerNAC.

INITIALIZE JButton to button_clickAC, button_addAC, button_registerAC, button_Display, button_Clear, button_clickNAC, button_addNAC, button_registerNAC, button_RemoveNAC, and button_NADisplay.

INITIALIZE JLabel to label_CourseForm, label_AcademicCourse, label_CourseID, label_CourseName, label_Duration, label_Level, label_Credit, label_NoOfAssessment, label_ACregisterID, label_CourseLeader, label_LecturerName, label_ACStartDate, label_ACEndDate, label_NonAcademicCourse, label_NACourseID, label_NACourseName, label_NADuration, label_Prerequisite, label_NACregisterID, label_NACourseLeader, label_InstructorName, label_NACStartDate, label_NACEndDate, and label_ExamDate.

INITIALIZE JTextField to field_ID, field_CName, field_Duration, field_Level, field_Credit, field_NoOfAssessment, field_ACregisterID, field_ACLeader, field_LName, fieldNA_ID, fieldNA_CName, field_NADuration, field_Prerequisite, field_NACregisterID, field_NACLeader, and field_IName.

INITIALIZE JComboBox to box_ACYear1, box_ACMonth1, box_ACDay1, box_ACYear2, box_ACMonth2, box_ACDay2, box_NACYear1, box_NACMonth1, box_NACDay1, box_NACYear2, box_NACMonth2, box_NACDay2, box_NACYear3, box_NACMonth3, and box_NACDay3.

INITIALIZE String to ACStartingDate, ACCompletionDate, NACStartingDate, NACCompletionDate, and ExamDate.

INITIALIZE INGCollege to main.

CREATE gUI() as void.
DO

INITIALIZE JFrame to frame.
INITIALIZE JPanel to panel.
INITIALIZE JPanel to panel_Academic.
INITIALIZE JPanel to panel_addAC.
INITIALIZE JPanel to panel_registerAC.
INITIALIZE JPanel to panel_NonAcademic.
INITIALIZE JPanel to panel_addNAC.
INITIALIZE JPanel to panel_registerNAC.

SET Layout as null to panel.

SET Bounds of x-axis: 32px, y-axis: 115px, width: 772px, and height: 372px to panel_Academic.

SET Layout (null) to panel.

SET Bounds of x-axis: 42px, y-axis: 155px, width: 370px, and height: 320px to panel_addAC.
SET Layout (null) to panel_addAC.

SET Bounds of x-axis: 425px, y-axis: 155px, width: 370px, and height: 320px to panel_registerAC.
SET Layout (null) to panel_registerAC.

SET Bounds of x-axis: 32px, y-axis: 115px, width: 772px, and height: 372px to panel_NonAcademic.
SET Layout (null) to panel_NonAcademic.

SET Bounds of x-axis: 42px, y-axis: 155px, width: 370px, and height: 320px to panel_addNAC.
SET Layout (null) to panel_addNAC.

SET Bounds of x-axis: 425px, y-axis: 155px, width: 370px, and height: 320px to panel_registerNAC.
SET Layout (null) to panel_registerNAC.

INITIALIZE Color to color_bg with RGB (188, 235, 253).

SET color_bg to panel, panel_Academic, and panel_NonAcademic.

SET Background (white) to panel_addAC, panel_registerAC, panel_addNAC, and panel_registerNAC.

INITIALIZE Color to color_line with RGB (119, 202, 236).

SET Border (color_line) to panel_Academic, and panel_NonAcademic.

INITIALIZE JLabel to label_CourseForm.
SET text (“Course Registration Form”) to label_CourseForm.
SET Bounds of x-axis: 300px, y-axis: 20px, width: 250px, and height: 30px to label_CourseForm.
INITIALIZE Font ("Maiandra GD", Font.BOLD, 20) to font_titles.
SET font_titles to label_CourseForm.
INITIALIZE Color to color_Courseform with RGB (8, 170, 236).
INITIALIZE Color to color_Button with RGB (27, 187, 252).
SET Foreground (color_Courseform) to label_CourseForm.
ADD label_CourseForm to panel.

INITIALIZE Font ("Sitka Heading", Font.BOLD, 15) to font_button.
INITIALIZE Font ("Sitka Heading", Font.BOLD, 16) to font_buttonInside.

INITIALIZE JButton to button_clickAC.
SET text (“Click here for Academic Course.”) to button_clickAC.
SET Bounds of x-axis: 32px, y-axis: 70px, width: 300px, and height: 27px to button_clickAC.
SET font_button to button_clickAC.
SET Background (white) to button_clickAC.
SET Foreground (color_Button) to button_clickAC.
CALL addActionListener method of button_clickAC and pass the current object as a parameter.
ADD button_clickAC to panel.

SET text (“Click here for Non-Academic Course.”) to button_clickNAC.

SET Bounds of x-axis: 504px, y-axis: 70px, width: 300px, and height: 27px to button_clickNAC.
SET font_button to button_clickNAC.
SET Background (white) to button_clickNAC.
SET Foreground (color_Button) to button_clickNAC.
CALL addActionListener method of button_clickNAC and pass the current object as a parameter.
ADD button_clickNAC to panel.

// Start Of Academic Course Form.

INITIALIZE JLabel to label_AcademicCourse.
SET text (“Academic Course”) to label_AcademicCourse.
SET Bounds of x-axis: 307px, y-axis: 5px, width: 250px, and height: 30px to label_AcademicCourse.
SET Font (font_titles) to label_AcademicCourse.
ADD label_AcademicCourse to panel_Academic.

SET Border (black) to panel_addAC and panel_registerAC.

INITIALIZE Font ("Maiandra GD", Font.BOLD, 15) to font_form.
INITIALIZE Font ("Maiandra GD", Font.BOLD, 13) to font_formextra.

INITIALIZE JLabel to label_CourseID.
SET text (“Course ID: ”) to label_CourseID.
SET Bounds of x-axis: 15px, y-axis: 10px, width: 135px, and height: 20px to label_CourseID.
SET Font (font_form) to label_CourseID.
ADD label_CourseID to panel_addAC.

INITIALIZE JTextField to field_ID.
SET Bounds of x-axis: 160px, y-axis: 10px, width: 195px, and height: 27px to field_ID.
SET Border (color_line) to field_ID.
ADD field_ID to panel_addAC.

INITIALIZE JLabel to label_CourseName.
SET text (“Course Name: ”) to label_CourseName.
SET Bounds of x-axis: 15px, y-axis: 52px, width: 135px, and height: 20px to label_CourseName.
SET Font (font_form) to label_CourseName.
ADD label_CourseName to panel_addAC.

INITIALIZE JTextField to field_CName.
SET Bounds of x-axis: 160px, y-axis: 50px, width: 195px, and height: 27px to field_CName.
SET Border (color_line) to field_CName.
ADD field_CName to panel_addAC.

INITIALIZE JLabel to label_Duration.
SET text (“Duration: ”) to label_Duration.
SET Bounds of x-axis: 15px, y-axis: 92px, width: 135px, and height: 20px to label_Duration.
SET font (font_form) to label_Duration.
ADD label_Duration to panel_addAC.

INITIALIZE JTextField to field_Duration.
SET Bounds of x-axis: 160px, y-axis: 90px, width: 195px, and height: 27px to field_Duration.
SET Border (color_line) to field_Duration.

ADD field_Duration to panel_addAC.

INITIALIZE JLabel to label_Level.
SET text (“Level: ”) to label_Level.
SET Bounds of x-axis: 15px, y-axis: 132px, width: 135px, and height: 20px to label_Level.
SET Font (font_form) to label_Level.
ADD label_Level to panel_addAC.

INITIALIZE JTextField to field_Level.
SET Bounds of x-axis: 160px, y-axis: 130px, width: 195px, and height: 27px to field_Level.
SET Border (color_line) to field_Level.
ADD field_Level to panel_addAC.

INITIALIZE JLabel to label_Credit.
SET text (“Credit: ”) to label_Credit.
SET Bounds of x-axis: 15px, y-axis: 172px, width: 135px, and height: 20px to label_Credit.
SET Font (font_form) to label_Credit.
ADD label_Credit to panel_addAC.

INITIALIZE JTextField to field_Credit.
SET Bounds of x-axis: 160px, y-axis: 170px, width: 195px, and height: 27px to field_ Credit.
SET Border (color_line) to field_Credit.
ADD field_Credit to panel_addAC.

INITIALIZE JLabel to label_ NoOfAssessment.
SET text (“No. Of Assessment: ”) to label_ NoOfAssessment.

SET Bounds of x-axis: 15px, y-axis: 212px, width: 135px, and height: 20px to label_Level.
SET Font (font_formextra) to label_ NoOfAssessment.
ADD label_NoOfAssessment to panel_addAC.

INITIALIZE JTextField to field_ NoOfAssessment.
SET Bounds of x-axis: 160px, y-axis: 210px, width: 195px, and height: 27px to field_NoOfAssessment.
SET Border (color_line) to field_NoOfAssessment.
ADD field_NoOfAssessment to panel_addAC.

INITIALIZE JButton to button_addAC.
SET text (“Add Academic Course”) to button_addAC.
SET Bounds of x-axis: 57px, y-axis: 260px, width: 260px, and height: 35px to button_ addAC.
SET Font (font_buttonInside) to button_addAC.
SET Background (white) to button_addAC.
SET Foreground (color_line) to button_addAC.
CALL addActionListener method of button_addAC and pass the current object as a parameter.
ADD button_addAC to panel_addAC.

INITIALIZE JLabel to label_ACregisterID.
SET text (“Course ID: ”) to label_ACregisterID.
SET Bounds of x-axis: 15px, y-axis: 10px, width: 135px, and height: 20px to label_ACregisterID.
SET Font (font_form) to label_ACregisterID.
SET Foreground (color_line) to label_ACregisterID.
ADD label_ACregisterID to panel_registerAC.

INITIALIZE JTextField to field_ACregisterID.
SET Bounds of x-axis: 160px, y-axis: 10px, width: 195px, and height: 27px to field_ACregisterID.
ADD field_ACregisterID to panel_registerAC.

INITIALIZE JLabel to label_CourseLeader.
SET text (“Course Leader: ”) to label_CourseLeader.
SET Bounds of x-axis: 15px, y-axis: 52px, width: 135px, and height: 20px to label_CourseLeader.
SET Font (font_form) to label_CourseLeader.
SET Foreground (color_line) to label_CourseLeader.
ADD label_CourseLeader to panel_registerAC.

INITIALIZE JTextField to field_ACLeader.
SET Bounds of x-axis: 160px, y-axis: 50px, width: 195px, and height: 27px to field_ACLeader.
ADD field_ACLeader to panel_registerAC.

INITIALIZE JLabel to label_LecturerName.
SET text (“Lecturer Name: ”) to label_LecturerName.
SET Bounds of x-axis: 15px, y-axis: 92px, width: 135px, and height: 20px to label_LecturerName.
SET Font (font_form) to label_LecturerName.
SET Foreground (color_line) to label_LecturerName.
ADD label_LecturerName to panel_registerAC.

INITIALIZE JTextField to field_LName.
SET Bounds of x-axis: 160px, y-axis: 90px, width: 195px, and height: 27px to field_LName.
ADD field_LName to panel_registerAC.

INITIALIZE JLabel to label_ACStartDate.
SET text (“Starting Date: ”) to label_ACStartDate.
SET Bounds of x-axis: 15px, y-axis: 132px, width: 135px, and height: 20px to label_ACStartDate.
SET Font (font_form) to label_ACStartDate.
SET Foreground (color_line) to label_ACStartDate.
ADD label_ACStartDate to panel_registerAC.

INITIALIZE Font ("Maiandra GD", Font.BOLD, 12) to font_box.

INITIALIZE Integer to yeardate [] with 11 integer values.
DECLARE int year as 2020.
FOR int i is equal to 0, i is less than or equal to 10; increment i.
DO
    INITIALIZE yeardate [i] as year.
    INCREMENT year.
END DO.

INITIALIZE JCombobox to box_ACYear1.
SET yeardate to box_ACYear1.
SET Bounds of x-axis: 160px, y-axis: 130px, width: 60px, and height: 27px to box_ACYear1.
SET Font (font_box) to box_ACYear1.
SET Background (white) to box_ACYear1.
ADD box_ACYear1 to panel_registerAC.

INITIALIZE months [] as {"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"}

INITIALIZE JCombobox to box_ACMonth1.

SET months to box_ACMonth1.

SET Bounds of x-axis: 221px, y-axis: 130px, width: 88px, and height: 27px to box_ACMonth1.

SET Font (font_box) to box_ACMonth1.

SET Background (white) to box_ACMonth1.

ADD box_ACMonth1 to panel_registerAC.

INITIALIZE Integer to day [] with 31 integer values.

FOR int i is equal to 0, i is less than 31; increment i.

DO

    INITIALIZE day [i] as i +1.

END DO.

INITIALIZE JCombobox to box_ACDay1.

SET day to box_ ACDay1.

SET Bounds of x-axis: 310px, y-axis: 130px, width: 44px, and height: 27px to box_ACDay1.

SET Font (font_box) to box_ACDay1.

SET Background (white) to box_ACDay1.

ADD box_ACDay1 to panel_registerAC.

INITIALIZE JLabel to label_ACEndDate.

SET text (“Completion Date: ”) to label_ACEndDate.

SET Bounds of x-axis: 15px, y-axis: 172px, width: 135px, and height: 20px to label_ACEndDate.

SET Font (font_form) to label_ACEndDate.

SET Foreground (color_line) to label_ACEndDate.

ADD label_ACEndDate to panel_registerAC.

INITIALIZE JCombobox to box_ACYear2.
SET yeardate to box_ACYear2.
SET Bounds of x-axis: 160px, y-axis: 170px, width: 60px, and height: 27px to box_ACYear2.
SET Font (font_box) to box_ACYear2.
SET Background (white) to box_ACYear2.
ADD box_ACYear2 to panel_registerAC.

INITIALIZE JCombobox to box_ACMonth2.
SET months to box_ACMonth2.
SET Bounds of x-axis: 221px, y-axis: 170px, width: 88px, and height: 27px to box_ACMonth2.
SET Font (font_box) to box_ACMonth2.
SET Background (white) to box_ACMonth2.
ADD box_ACMonth2 to panel_registerAC.

INITIALIZE JCombobox to box_ACDay2.
SET day to box_ ACDay2.
SET Bounds of x-axis: 310px, y-axis: 170px, width: 44px, and height: 27px to box_ACDay2.
SET Font (font_box) to box_ACDay2.
SET Background (white) to box_ACDay2.
ADD box_ACDay2 to panel_registerAC.

INITIALIZE JButton to button_registerAC.
SET text (“Register Academic Course”) to button_registerAC.
SET Bounds of x-axis: 57px, y-axis: 260px, width: 260px, and height: 35px to button_registerAC.
SET Font (font_buttonInside) to button_registerAC.
SET Background (white) to button_registerAC.

SET Foreground (color_line) to button_registerAC.
CALL addActionListener method of button_registerAC and pass the current object as a parameter.
ADD button_registerAC to panel_registerAC.

INITIALIZE JButton to button_Display.
SET text (“Display”) to button_Display.
SET Bounds of x-axis: 595px, y-axis: 505px, width: 100px, and height: 30px to button_Display.
SET Font (font_buttonInside) to button_Display.
SET Background (white) to button_Display.
SET Foreground (color_line) to button_Display.
CALL addActionListener method of button_Display and pass the current object as a parameter.
ADD button_Display to panel.

// Start Of Non-Academic Course Form.

INITIALIZE JLabel to label_NonAcademicCourse.
SET text (“Non-Academic Course”) to label_NonAcademicCourse.
SET Bounds of x-axis: 280px, y-axis: 5px, width: 250px, and height: 30px to label_NonAcademicCourse.
SET Font (font_titles) to label_NonAcademicCourse.
ADD label_NonAcademicCourse to panel_NonAcademic.

SET Border (black) to panel_addNAC and panel_registerNAC.

INITIALIZE JLabel to label_NACourseID.
SET text (“Course ID: ”) to label_NACourseID.

SET Bounds of x-axis: 15px, y-axis: 10px, width: 135px, and height: 20px to label_NACourseID.
SET Font (font_form) to label_NACourseID.
ADD label_NACourseID to panel_addNAC.

INITIALIZE JTextField to fieldNA_ID.
SET Bounds of x-axis: 160px, y-axis: 10px, width: 195px, and height: 27px to fieldNA_ID.
SET Border (color_line) to fieldNA_ID.
ADD fieldNA_ID to panel_addNAC.

INITIALIZE JLabel to label_NACourseName.
SET text (“Course Name: ”) to label_NACourseName.
SET Bounds of x-axis: 15px, y-axis: 52px, width: 135px, and height: 20px to label_NACourseName.
SET Font (font_form) to label_NACourseName.
ADD label_NACourseName to panel_addNAC.

INITIALIZE JTextField to fieldNA_CName.
SET Bounds of x-axis: 160px, y-axis: 50px, width: 195px, and height: 27px to fieldNA_CName.
SET Border (color_line) to field_CName.
ADD field_CName to panel_addAC.

INITIALIZE JLabel to label_NADuration.
SET text (“Duration: ”) to label_NADuration.
SET Bounds of x-axis: 15px, y-axis: 92px, width: 135px, and height: 20px to label_NADuration.
SET font (font_form) to label_NADuration.
ADD label_NADuration to panel_addNAC.

INITIALIZE JTextField to field_NADuration.
SET Bounds of x-axis: 160px, y-axis: 90px, width: 195px, and height: 27px to field_NADuration.
SET Border (color_line) to field_NADuration.
ADD field_NADuration to panel_addNAC.

INITIALIZE JLabel to label_Prerequisite.
SET text (“Prerequisite: ”) to label_Prerequisite.
SET Bounds of x-axis: 15px, y-axis: 132px, width: 135px, and height: 20px to label_Prerequisite.
SET Font (font_form) to label_Prerequisite.
ADD label_Prerequisite to panel_addNAC.

INITIALIZE JTextField to field_Prerequisite.
SET Bounds of x-axis: 160px, y-axis: 130px, width: 195px, and height: 27px to field_Prerequisite.
SET Border (color_line) to field_Prerequisite.
ADD field_Prerequisite to panel_addNAC.

INITIALIZE JButton to button_addNAC.
SET text (“Add Non-Academic Course”) to button_addNAC.
SET Bounds of x-axis: 57px, y-axis: 195px, width: 260px, and height: 35px to button_addNAC.
SET Font (font_buttonInside) to button_addNAC.
SET Background (white) to button_addNAC.
SET Foreground (color_line) to button_addNAC.
CALL addActionListener method of button_addNAC and pass the current object as a parameter.
ADD button_addNAC to panel_addNAC.

INITIALIZE JButton to button_RemoveNAC.

SET text (“Remove Course”) to button_RemoveNAC.

SET Bounds of x-axis: 57px, y-axis: 260px, width: 260px, and height: 35px to button_RemoveNAC.

SET Font (font_buttonInside) to button_RemoveNAC.

SET Background (white) to button_RemoveNAC.

SET Foreground (color_line) to button_RemoveNAC.

CALL addActionListener method of button_RemoveNAC and pass the current object as a parameter.

ADD button_RemoveNAC to panel_addNAC.

INITIALIZE JLabel to label_NACregisterID.

SET text (“Course ID: ”) to label_NACregisterID.

SET Bounds of x-axis: 15px, y-axis: 10px, width: 135px, and height: 20px to label_NACregisterID.

SET Font (font_form) to label_NACregisterID.

SET Foreground (color_line) to label_NACregisterID.

ADD label_NACregisterID to panel_registerNAC.

INITIALIZE JTextField to field_NACregisterID.

SET Bounds of x-axis: 160px, y-axis: 10px, width: 195px, and height: 27px to field_NACregisterID.

ADD field_NACregisterID to panel_registerNAC.

INITIALIZE JLabel to label_NACourseLeader.

SET text (“Course Leader: ”) to label_NACourseLeader.

SET Bounds of x-axis: 15px, y-axis: 52px, width: 135px, and height: 20px to label_NACourseLeader.

SET Font (font_form) to label_NACourseLeader.

SET Foreground (color_line) to label_NACourseLeader.
ADD label_NACourseLeader to panel_registerNAC.

INITIALIZE JTextField to field_NACLeader.
SET Bounds of x-axis: 160px, y-axis: 50px, width: 195px, and height: 27px to field_NACLeader.
ADD field_NACLeader to panel_registerNAC.

INITIALIZE JLabel to label_InstructorName.
SET text (“Instructor Name: ”) to label_InstructorName.
SET Bounds of x-axis: 15px, y-axis: 92px, width: 135px, and height: 20px to label_InstructorName.
SET Font (font_form) to label_InstructorName.
SET Foreground (color_line) to label_InstructorName.
ADD label_InstructorName to panel_registerNAC.

INITIALIZE JTextField to field_IName.
SET Bounds of x-axis: 160px, y-axis: 90px, width: 195px, and height: 27px to field_IName.
ADD field_IName to panel_registerNAC.

INITIALIZE JLabel to label_NACStartDate.
SET text (“Starting Date: ”) to label_NACStartDate.
SET Bounds of x-axis: 15px, y-axis: 132px, width: 135px, and height: 20px to label_NACStartDate.
SET Font (font_form) to label_NACStartDate.
SET Foreground (color_line) to label_NACStartDate.
ADD label_NACStartDate to panel_registerNAC.

INITIALIZE JCombobox to box_NACYear1.

SET yeardate to box_NACYear1.

SET Bounds of x-axis: 160px, y-axis: 130px, width: 60px, and height: 27px to box_NACYear1.

SET Font (font_box) to box_NACYear1.

SET Background (white) to box_NACYear1.

ADD box_NACYear1 to panel_registerNAC.

INITIALIZE JCombobox to box_NACMonth1.

SET months to box_NACMonth1.

SET Bounds of x-axis: 221px, y-axis: 130px, width: 88px, and height: 27px to box_NACMonth1.

SET Font (font_box) to box_NACMonth1.

SET Background (white) to box_NACMonth1.

ADD box_NACMonth1 to panel_registerNAC.

INITIALIZE JCombobox to box_NACDay1.

SET day to box_NACDay1.

SET Bounds of x-axis: 310px, y-axis: 130px, width: 44px, and height: 27px to box_NACDay1.

SET Font (font_box) to box_NACDay1.

SET Background (white) to box_NACDay1.

ADD box_NACDay1 to panel_registerNAC.

INITIALIZE JLabel to label_NACEndDate.

SET text (“Completion Date: ”) to label_NACEndDate.

SET Bounds of x-axis: 15px, y-axis: 172px, width: 135px, and height: 20px to label_NACEndDate.

SET Font (font_form) to label_NACEndDate.

SET Foreground (color_line) to label_NACEndDate.

ADD label_NACEndDate to panel_registerNAC.

INITIALIZE JCombobox to box_NACYear2.
SET yeardate to box_NACYear2.
SET Bounds of x-axis: 160px, y-axis: 170px, width: 60px, and height: 27px to box_NACYear2.
SET Font (font_box) to box_NACYear2.
SET Background (white) to box_NACYear2.
ADD box_NACYear2 to panel_registerNAC.

INITIALIZE JCombobox to box_NACMonth2.
SET months to box_NACMonth2.
SET Bounds of x-axis: 221px, y-axis: 170px, width: 88px, and height: 27px to box_NACMonth2.
SET Font (font_box) to box_NACMonth2.
SET Background (white) to box_NACMonth2.
ADD box_NACMonth2 to panel_registerNAC.

INITIALIZE JCombobox to box_NACDay2.
SET day to box_ NACDay2.
SET Bounds of x-axis: 310px, y-axis: 170px, width: 44px, and height: 27px to box_NACDay2.
SET Font (font_box) to box_NACDay2.
SET Background (white) to box_NACDay2.
ADD box_NACDay2 to panel_registerNAC.

INITIALIZE JLabel to label_ExamDate.
SET text (“Exam Date: ”) to label_ExamDate.
SET Bounds of x-axis: 15px, y-axis: 208px, width: 135px, and height: 20px to label_ExamDate.
SET Font (font_form) to label_ExamDate.

SET Foreground (color_line) to label_ExamDate.
ADD label_ExamDate to panel_registerNAC.

INITIALIZE JCombobox to box_NACYear3.
SET yeardate to box_NACYear3.
SET Bounds of x-axis: 160px, y-axis: 210px, width: 60px, and height: 27px to box_NACYear3.
SET Font (font_box) to box_NACYear3.
SET Background (white) to box_NACYear3.
ADD box_NACYear3 to panel_registerNAC.

INITIALIZE JCombobox to box_NACMonth3.
SET months to box_NACMonth3.
SET Bounds of x-axis: 221px, y-axis: 210px, width: 88px, and height: 27px to box_NACMonth3.
SET Font (font_box) to box_NACMonth3.
SET Background (white) to box_NACMonth3.
ADD box_NACMonth3 to panel_registerNAC.

INITIALIZE JCombobox to box_NACDay3.
SET day to box_NACDay3.
SET Bounds of x-axis: 310px, y-axis: 210px, width: 44px, and height: 27px to box_NACDay3.
SET Font (font_box) to box_NACDay3.
SET Background (white) to box_NACDay3.
ADD box_NACDay3 to panel_registerNAC.

INITIALIZE JButton to button_registerNAC.
SET text (“Register Non-Academic Course”) to button_registerNAC.

SET Bounds of x-axis: 57px, y-axis: 260px, width: 260px, and height: 35px to button_registerNAC.

SET Font (font_buttonInside) to button_registerNAC.

SET Background (white) to button_registerNAC.

SET Foreground (color_line) to button_registerNAC.

CALL addActionListener method of button_registerNAC and pass the current object as a parameter.

ADD button_registerNAC to panel_registerNAC.

INITIALIZE JButton to button_NADisplay.

SET text (“Display”) to button_NADisplay.

SET Bounds of x-axis: 595px, y-axis: 505px, width: 100px, and height: 30px to button_NADisplay.

SET Font (font_buttonInside) to button_NADisplay.

SET Background (white) to button_NADisplay.

SET Foreground (color_line) to button_NADisplay.

CALL addActionListener method of button_NADisplay and pass the current object as a parameter.

ADD button_NADisplay to panel.

INITIALIZE JButton to button_Clear.

SET text (“Clear”) to button_Clear.

SET Bounds of x-axis: 704px, y-axis: 505px, width: 100px, and height: 30px to button_Clear.

SET Font (font_buttonInside) to button_Clear.

SET Background (white) to button_Clear.

SET Foreground (color_line) to button_Clear.

CALL addActionListener method of button_Clear and pass the current object as a parameter.

ADD button_Clear to panel.

SET Visible (false) for button_NADisplay, panel_addNAC, panel_registerNAC, and panel_NonAcademic.

ADD panel_addAC, panel_registerAC, panel_Academic, panel_addNAC, panel_registerNAC, panel_NonAcademic, and panel to frame.
SET Bounds of x-axis: 345px, y-axis: 132px, width: 850px, and height: 595px to frame.
SET Title ("Course Registration Form") to frame.
SET Resizable (false) of frame.
SET Visible (true) to frame.

END DO.

CREATE main (String[] args) as public static void.
DO
CALL the gUI method.
END DO.

// Start Of Academic Course Getters Method.

CREATE getACourseID() as String type.
DO
RETURN this.field_ID.getText().
END DO.

CREATE getACourseName() as String type
DO
RETURN this.field_CName.getText().
END DO.

```
CREATE getACDuration() as int type
DO
        RETURN Integer.parseInt(this.field_Duration.getText()).
END DO.

CREATE getLevel() as String type
DO
        RETURN this.field_Level.getText().
END DO.

CREATE getCredit() as String type
DO
        RETURN this.field.getCredit.getText().
END DO.

CREATE getNoOfAssessment() as int type
DO
        RETURN Integer.parseInt(this.field_getNoOfAssessment.getText()).
END DO.

CREATE getACourseRegisterID() as String type.
DO
        RETURN this.field_ACregisterID.getText().
END DO.

CREATE getACourseLeader() as String type
DO
        RETURN this.field_ACLeader.getText().
END DO.
```

```
CREATE getACLecturerName() as String type
DO
    RETURN this.field_LName.getText().
END DO.

CREATE getACStartDate() as String type
DO
    ACStartingDate = box_ACYear1.getSelectedItem() + " " +
    box_ACMonth1.getSelectedItem() + " " + box_ACDay1.getSelectedItem()
    RETURN ACStartingDate.
END DO.

CREATE getACCompleteDate() as String type
DO
    ACCompletionDate = box_ACYear2.getSelectedItem() + " " +
    box_ACMonth2.getSelectedItem() + " " + box_ACDay2.getSelectedItem()
    RETURN ACCompletionDate.
END DO.

// Start Of Non-Academic Course Getters Method.

CREATE getNACourseID() as String type.
DO
    RETURN this.fieldNA_ID.getText().
END DO.

CREATE getNACourseName() as String type
DO
    RETURN this.fieldNA_CName.getText().
END DO.
```

```
CREATE getNACDuration() as int type
DO
    RETURN Integer.parseInt(this.field_NADuration.getText()).
END DO.

CREATE getPrerequisite() as String type
DO
    RETURN this.field_Prerequisite.getText().
END DO.

CREATE getNACourseRegisterID() as String type.
DO
    RETURN this.field_NACregisterID.getText().
END DO.

CREATE getNACourseLeader() as String type
DO
    RETURN this.field_NACLeader.getText().
END DO.

CREATE getInstructorName() as String type
DO
    RETURN this.field_IName.getText().
END DO.

CREATE getNACStartDate() as String type
DO
```

```
        NACStartingDate = box_NACYear1.getSelectedItem() + " " +
        box_NACMonth1.getSelectedItem() + " " +
        box_NACDay1.getSelectedItem()
        RETURN NACStartingDate.
    END DO.

    CREATE getNACCompleteDate() as String type
    DO
        NACCompletionDate = box_NACYear2.getSelectedItem() + " " +
        box_NACMonth2.getSelectedItem() + " " +
        box_NACDay2.getSelectedItem()
        RETURN NACCompletionDate.
    END DO.

    CREATE getExamDate() as String type
    DO
        ExamDate= box_NACYear3.getSelectedItem() + " " +
        box_NACMonth3.getSelectedItem() + " " +
        box_NACDay3.getSelectedItem()
        RETURN ExamDate.
    END DO.

    // Implementing actionPerformed Method.

    CREATE actionPerformed (ActionEvent e) as public void.
    DO
        IF (e.getSource () == button_clickAC)
            SET Visible (true) for button_Display.
            SET Visible (true) for panel_addAC.
            SET Visible (true) for panel_registerAC.
```

```
        SET Visible (true) for panel_Academic.

        SET Visible (false) for button_NADisplay.
        SET Visible (false) for panel_addNAC.
        SET Visible (false) for panel_registerNAC.
        SET Visible (false) for panel_NonAcademic.
    END IF.

    ELSE IF (e.getSource () == button_ClickNAC)
        SET Visible (true) for button_NADisplay.
        SET Visible (true) for panel_addNAC.
        SET Visible (true) for panel_registerNAC.
        SET Visible (true) for panel_NonAcademic.

        SET Visible (false) for button_Display.
        SET Visible (false) for panel_addAC.
        SET Visible (false) for panel_registerAC.
        SET Visible (false) for panel_Academic.
    END ELSE IF.

    //Academic Buttons.

    ELSE IF (e.getSource () == button_addAC)
        INITIALIZE int to count and count2 with it equal to 0.
        IF (getACourseID().equals("") || getACourseName().equals("") ||
        getLevel().equals("") || getCredit().equals(""))
            SHOW MessageDialog Error Message: "Please ensure that
            all fields are filled." "ATTENTION!"
            SET count as 1.
        END IF.
```

```
IF (count == 0)
    FOR (Course alist: list_Course)
    DO
        IF (getACourseID().equals(alist.getCourseID()))
            SHOW MessageDialog Error Message: "The Course ID " + getACourseID() + "\n"+ " with Course Name" + getACourseName() + "\n"+ " has already been added." "ATTENTION!"
            SET count2 as 1.
            BREAK.
        END IF.
        TRY
        DO
            GET Integer.parseInt text of this.field_Duration
            TRY
            DO
                GET Integer.parseInt text of this.field_NoOfAssessment.
            END DO.
            CATCH (NumberFormatException ex)
            DO
                SHOW MessageDialog Error Message: “The Assessment must be an integer number.” “ATTENTION!”
                SET count2 as 1.
            END DO.
        END DO.
    END DO.
    CATCH (NumberFormatException ex)
    DO
```

```
                    SHOW MessageDialog Error Message: “The Duration
                    must be an integer number.”  “ATTENTION!”
                    SET count2 as 1.
                END DO.
                IF (count2 == 0)
                    SHOW MessageDialog "The following values has been
                    added \n"+ "Course ID: " + getACourseID() + "\n"+
                    "Course Name: " + getACourseName() + "\n"+
                    "Duration: " + getACDuration() + "\n"+ "Level: " +
                    getLevel() + "\n" + "Credit: " + getCredit() + "\n"+ "No.
                    Of Assessment: " + getNoOfAssessment()+"\n" + "to
                    Academic Course."
                    INITIALIZE AcademicCourse as AcademicObject with
                    (getACourseID(), getACourseName(),
                    getACDuration(), getLevel(), getCredit(),
                    getNoOfAssessment()).
                    ADD AcademicObject to list_Course.
                END IF.
            END DO.
        END ELSE IF.

        ELSE IF (e.getSource() == button_registerAC)
            INITIALIZE int to count with it equal to 0.
            IF (getACourseRegisterID().equals("") || getACourseLeader().equals("")  ||
            getACLecturerName().equals(""))
                SHOW MessageDialog Error Message: "Please ensure that
                all fields are filled." "ATTENTION!"
                SET count as 1.
            END IF.
            IF (count == 0)
```

```
FOR (Course alist: list_Course).
DO
    IF (getACourseRegisterID().equals(alist.getCourseID()) &&
    alist instanceof AcademicCourse).
        INITIALIZE AcademicCourse to
        AcademicObject alist.
        SET count as 1.
        IF (AcademicObject.getisRegistered() == true).
            SHOW MessageDialog "This course is
            already registered."
            SET count as 1.
        END IF.
        ELSE IF (AcademicObject.getisRegistered() ==
        false)
            GET AcademicObject.register
            (getACourseLeader(),
            getACLecturerName(), getACStartDate()
            ,getACCompleteDate())
            SHOW MessageDialog "The Academic
            Course has been registered. \n"
            +"Course ID: " + getACourseRegisterID()
            + "\n" + "Course Leader: "+
            getACourseLeader() + "\n" + "Lecturer
            Name: " + getACLecturerName() + "\n"+
            "Starting Date:" + getACStartDate() +
            "\n" +"Completion Date: "+
            getACCompleteDate().
            SET count as 1.
        END ELSE IF.
    END IF.
```

```
                END DO.
            END IF.
            IF (count == 0)
                SHOW MessageDialog Warning Message: "Course ID does
                not match. Please enter the correct value." "ATTENTION!".
            END IF.
        END ELSE IF.

        ELSE IF (e.getSource() == button_Display)
            FOR (Course alist: list_Course)
            DO
                IF (alist instanceof AcademicCourse)
                    INITIALIZE AcademicCourse to AcademicObject alist.
                    PRINT (“”)
                    DISPLAY AcademicObject.
                END IF.
            END DO.
        END ELSE IF.

        //NonAcademic Buttons.

        ELSE IF (e.getSource() == button_addNAC)
            INITIALIZE int to count and count2 with it equal to 0.
            IF (getNACourseID().equals("") || getNACourseName().equals("") ||
            getPrerequisite().equals(""))
                SHOW MessageDialog Error Message: "Please ensure that
                all fields are filled." "ATTENTION!"
                SET count as 1.
            END IF.
            IF (count == 0)
```

```
FOR (Course alist: list_Course)
DO
    IF (getNACourseID().equals(alist.getCourseID()))
        SHOW MessageDialog Error Message: "The Course ID " + getNACourseID() + "\n"+ " with Course Name " + getNACourseName() + "\n" + " has already been added."  "ATTENTION!"
        SET count2 as 1.
        BREAK.
    END IF.
END DO.
TRY
DO
    GET Integer.parseInt text of this.field_NADuration.
END DO.
CATCH (NumberFormatException ex)
DO
    SHOW MessageDialog Error Message: “The Duration must be an integer number.”  “ATTENTION!”
    SET count2 as 1.
END DO.
IF (count2 == 0)
    SHOW MessageDialog "The following values has been added \n"+ "Course ID: " + getNACourseID() + "\n"+ "Course Name: " + getNACourseName() + "\n"+ "Duration: " + getNACDuration() + "\n" + "Prerequisite: " + getPrerequisite() + "\n" + "to Non-Academic Course."
    INITIALIZE NonAcademicCourse as NonAcademicObject with (getNACourseID(),
```

```
                    getNACourseName(), getNACDuration(),
                    getPrerequisite()).
                    ADD NonAcademicObject to list_Course.
                END IF.
            END IF.
        END ELSE IF.

        ELSE IF (e.getSource() == button_registerNAC)
            INITIALIZE int to count with it equal to 0.
            IF (getNACourseRegisterID().equals("") || getNACourseLeader().equals("") || getInstructorName().equals(""))
                SHOW MessageDialog Error Message: "Please ensure that all fields are filled." "ATTENTION!"
                SET count as 1.
            END IF.
            IF (count == 0)
                FOR (Course alist: list_Course).
                DO
                    IF (getNACourseRegisterID().equals(alist.getCourseID()) && alist instanceof NonAcademicCourse)
                        INITIALIZE NonAcademicCourse to NonAcademicObject alist.
                        SET count as 1.
                        IF (NonAcademicObject.getisRegistered() == true).
                            SHOW MessageDialog "This course is already registered."
                            SET count as 1.
                        END IF.
                        ELSE IF (NonAcademicObject.getisRegistered() == false)
```

```
                    GET NonAcademicObject.register
                    (getNACourseRegisterID(),
                    getInstructorName(), getNACStartDate()
                    ,getNACCompleteDate(),
                    getExamDate())
                    SHOW MessageDialog "The Non-
                    Academic Course has been registered.
                    \n"+ "Course ID: "+
                    getNACourseRegisterID() + "\n" +
                    "InstructorName: " + getInstructorName()
                    + "\n" + "Starting Date: " +
                    getNACStartDate() + "\n" + "Completion
                    Date: " + getNACCompleteDate() + "\n"+
                    "Exam Date: " + getExamDate().
                    SET count as 1.
                END ELSE IF.
            END IF.
        END DO.
    END IF.
    IF (count == 0)
        SHOW MessageDialog Warning Message: "Course ID does
        not match. Please enter the correct value." "ATTENTION!".
    END IF.
END ELSE IF.

ELSE IF (e.getSource() == button_RemoveNAC)
    INITIALIZE int to count with it equal to 0.
    IF (getNACourseID().equals("") || getNACourseRegisterID().equals("") )
        SHOW MessageDialog Error Message: "Please ensure that
        all fields are filled." "ATTENTION!"
```

```
        SET count as 1.
    END IF.
    IF (count == 0)
        FOR (Course alist: list_Course)
        DO
            IF (getNACourseID().equals(getNACourseID()) &&
            alist instanceof NonAcademicCourse)
                INITIALIZE NonAcademicCourse to NACourse
                alist.
                IF (NACourse.getisRemoved()==false)
                    IF (NACourse.getisRemoved()==false)
                        SHOW MessageDialog Error
                        Message: "The Non-Academic
                        Course with "+ "\n"+ "Course ID:
                        "+ getNACourseID() + " has been
                        removed.""ATTENTION!”
                    END IF.
                    ELSE IF (NACourse.getisRemoved()==true)
                        SHOW MessageDialog Error
                        Message: "The Non-Academic
                        Course with "+ "\n"+ "Course ID:
                        "+ getNACourseID() + " has
                        already been removed."
                        "ATTENTION!”
                        BREAK.
                    END ELSE IF.
                END IF.
            END IF.
        END DO.
    END IF.
```

```
            END ELSE IF.

            ELSE IF (e.getSource() == button_NADisplay)
                FOR (Course alist: list_Course)
                DO
                    IF (alist instanceof NonAcademicCourse)
                        INITIALIZE NonAcademicCourse to
                        NonAcademicObject alist.
                        PRINT (“”)
                        DISPLAY NonAcademicObject.
                    END IF.
                END DO.
            END ELSE IF.

            ELSE IF (e.getSource() == button_Clear)
                SET Text as (“”) to field_ID, field_CName, field_Duration,
                field_Level, field_Credit, field_NoOfAssessment, field_ACregisterID,
                field_ACLeader, field_LName, fieldNA_ID, fieldNA_CName,
                field_NADuration, field_Prerequisite, field_NACregisterID,
                field_NACLeader, field_IName, box_ACYear1, box_ACMonth1,
                box_ACDay1, box_ACYear2, box_ACMonth2, box_ACDay2,
                box_NACYear1, box_NACMonth1, box_NACDay1, box_NACYear2,
                box_NACMonth2, box_NACDay2, box_NACYear3,
                box_NACMonth3, and box_NACDay3.
            END ELSE IF.
        END DO.
    END DO.
```

4. Method Description.

public void gUI() This method contains an interface that allows the user to interact with devices through graphic elements. It is commonly known as GUI which stands for Graphic User Interface. Different visual components like JFrame, JPanel, JLabel, JTextField, JComboBox, and JButton is used to make it user-friendly as all the possible functions present in the code can be displayed without the end user having to input a command of codes. The appearance of the interface has been designed in this method to fit the aesthetics of the programmer, and it can be redesigned as per the requirement.
public static void main(String[] args) This is a main method that is used to call the gUI() method.
public String getACourseID() It is an accessor method used to return the value provided by the user in the field_ID, JTextField, which is the Course ID for the Academic Course.
public String getACourseName() It is an accessor method used to return the value provided by the user in the field_CName, JTextField, which is the Academic Course Name.
public int getACDuration() It is an accessor method used to return the value provided by the user in the field_Duration, JTextField. It is accepted as a string which is then converted into an Integer i.e., the Duration of the Academic Course.
public String getLevel() It is an accessor method used to return the value provided by the user in the field_Level, JTextField, which is the Level of Academic Course.

public String getCredit() It is an accessor method used to return the value provided by the user in the field_Credit, JTextField, which is the Credit of Academic Course.
public int getNoOfAssessment() It is an accessor method used to return the value provided by the user in the field_NoOfAssessment, JTextField which is accepted as a string which is then converted into an Integer i.e., the Number of Assessment of the Academic Course.
public String getACourseRegisterID() It is an accessor method used to return the value provided by the user in the field_ACregisterID, JTextField, which is the Register Course ID of the Academic Course.
public String getACourseLeader() It is an accessor method used to return the value provided by the user in the field_ACLeader, JTextField, which is the Academic Course Leader.
public String getACLecturerName() It is an accessor method used to return the value provided by the user in the field_LName, JTextField, which is the Academic Course Lecturer's Name.
public String getACStartDate() It is an accessor method used to return the values stored in String ACStartingDate which has items selected by the user in the JComboBox's: box_ACYear1, box_ACMonth1, and box_ACDay1 is the Academic Course's Starting Date.
public String getACCompleteDate()

It is an accessor method used to return the values stored in String ACCompletionDate which has items selected by the user in the JComboBox's: box_ACYear2, box_ACMonth2, and box_ACDay2 is the Academic Course's Completion Date.
public String getNACourseID() It is an accessor method used to return the value provided by the user in the fieldNA_ID, JTextField, which is the Course ID of the Non-Academic Course.
public String getNACourseName() It is an accessor method used to return the value provided by the user in the fieldNA_CName, JTextField, which is the Non-Academic Course Name.
public int getNACDuration() It is an accessor method used to return the value provided by the user in the field_NADuration, JTextField. It is accepted as a string which is then converted into an Integer i.e., the Duration of the Non-Academic Course.
public String getPrerequisite() It is an accessor method used to return the value provided by the user in the field_Prerequisite, JTextField, which is the Prerequisite of Non-Academic Course.
public String getNACourseRegisterID() It is an accessor method used to return the value provided by the user in the field_NACregisterID, JTextField, which is the Register Course ID of the Non-Academic Course.
public String getNACourseLeader() It is an accessor method used to return the value provided by the user in the field_NACLeader, JTextField, which is the Non-Academic Course Leader.

public String getInstructorName() It is an accessor method used to return the value provided by the user in the field_IName, JTextField, which is the Academic Course Instructor's Name.
public String getNACStartDate() It is an accessor method used to return the values stored in String NACStartingDate which has items selected by the user in the JComboBox's: box_NACYear1, box_NACMonth1, and box_NACDay1 is the Non-Academic Course's Starting Date.
public String getNACCompleteDate() It is an accessor method used to return the values stored in String NACCompletionDate which has items selected by the user in the JComboBox's: box_NACYear2, box_NACMonth2, and box_NACDay2 is the Non-Academic Course's Completion Date.
public String getExamDate() It is an accessor method used to return the values stored in String ExamDate which has items selected by the user in the JComboBox's: box_NACYear3, box_NACMonth3, and box_NACDay3 is the Non-Academic Course's Exam Date.
public void actionPerformed(ActionEvent e) This method is used to execute actionPerformed on command. Every button has its unique feature which is discussed below: 1. **if(e.getSource() == button_clickAC)** When the button_clickAC is clicked only the components which belongs to Academic Course is shown. The visibility of button_Display, panel_addAC, panel_registerAC, and panel_Academic is set to true whereas, button_NADisplay, panel_addNAC, panel_registerNAC, and panel_NonAcademic's visibility is set to false.

2. **else if(e.getSource() == button_clickNAC)**

   When the button_clickNAC is clicked only the components which belongs to Non-Academic Course is shown. The visibility of button_NADisplay, panel_addNAC, panel_registerNAC, and panel_NonAcademic is set to true whereas, button_Display, panel_addAC, panel_registerAC, and panel_Academic's visibility is set to false.

3. **else if(e.getSource() == button_addAC)**

   When the button_addAC is clicked, it first makes sure that no JTextFields is left empty, in case it is empty, an error message pops up. If the CourseID has already been added another relevant error message pops up. Moving on, it makes sure that Assessment and Duration's JTextField contains an integer. After clearing all the possible errors, it adds the Academic Course to the array list i.e., list_Course.

4. **else if(e.getSource() == button_registerAC)**

   When the button_registerAC is clicked, it first makes sure that no JTextFields is left empty, in case it is empty then an error message pops up. If the CourseID has already been registered another relevant error message pops up. After clearing all the possible errors, it tries to register the Academic Course but if the CourseID does not match with the previously added CourseID, another error message pops up asking the user to check the CourseID.

5. **else if(e.getSource() == button_Display)**

   When the button_Display is clicked, it searches for the list_Course instanceof AcademicCourse and then calls the AcademicCourse class's Display method. The details of the Course is printed in the terminal.

6. **else if(e.getSource() == button_addNAC)**

   When the button_addNAC is clicked, it first makes sure that no JTextFields is left empty, in case it is empty then an error message pops up. If the CourseID has already been added another relevant error message pops up. Moving on, it makes sure that Duration's JTextField contains an integer. After clearing all the possible errors, it adds the Non-Academic Course to the array list i.e., list_Course.

7. **else if(e.getSource() == button_registerNAC)**

   When the button_registerNAC is clicked, it first makes sure that no JTextFields is left empty, in case it is left empty then an error message pops up. If the CourseID has already been registered another relevant error message pops up. After clearing all the possible errors, it then tries to register the Academic Course but if the CourseID does not match with the previously added CourseID, another error message pops up asking the user check the CourseID.

8. **else if(e.getSource() == button_RemoveNAC)**

   When the button_RemoveNAC is clicked, it first checks if the Course has been added previously or registered with the help of CourseID's JTextField. Then if the course has not been removed it calls the remove method from the NonAcademicCourse class and removes the course. If incase the user tries to remove the course again, an error message pops up saying, "The course has already been removed."

9. **else if(e.getSource() == button_NADisplay)**

   When the button_Display is clicked, it searches for the list_Course instanceof NonAcademicCourse and then calls the NonAcademicCourse class's Display method. The details of the Course is printed in the terminal.

10. else if(e.getSource() == button_Clear) When the button_Clear is clicked, all the values in JTextField and the selected JComboBox's will be cleared.

Table 1: Method Description.

5. Testing (Inspection)

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

Test No.	**1.**
Objective:	To test if the program can be compiled and run using the command prompt.
Action:	➢ Go to the folder in which the project has been saved. ➢ Click on the address bar, which is situated at the top of file explorer. ➢ Type in cmd in the address bar and enter. ➢ The cmd also known as command prompt opens. ➢ To compile the program type in: "javac Course.java" "javac AcademicCourse.java" "javac NonAcademicCourse.java" "javac INGCollege.java" in the command prompt. ➢ To run the program type in "java INGCollege" in the command prompt.
Expected Result:	The program would get complied and run successfully using cmd.
Actual Result:	The program was compiled and run successfully using cmd.
Conclusion:	The test was successful.

Table 2: 5.1 Test 1: Test that the program can be compiled and run using the command prompt.

Evidence:

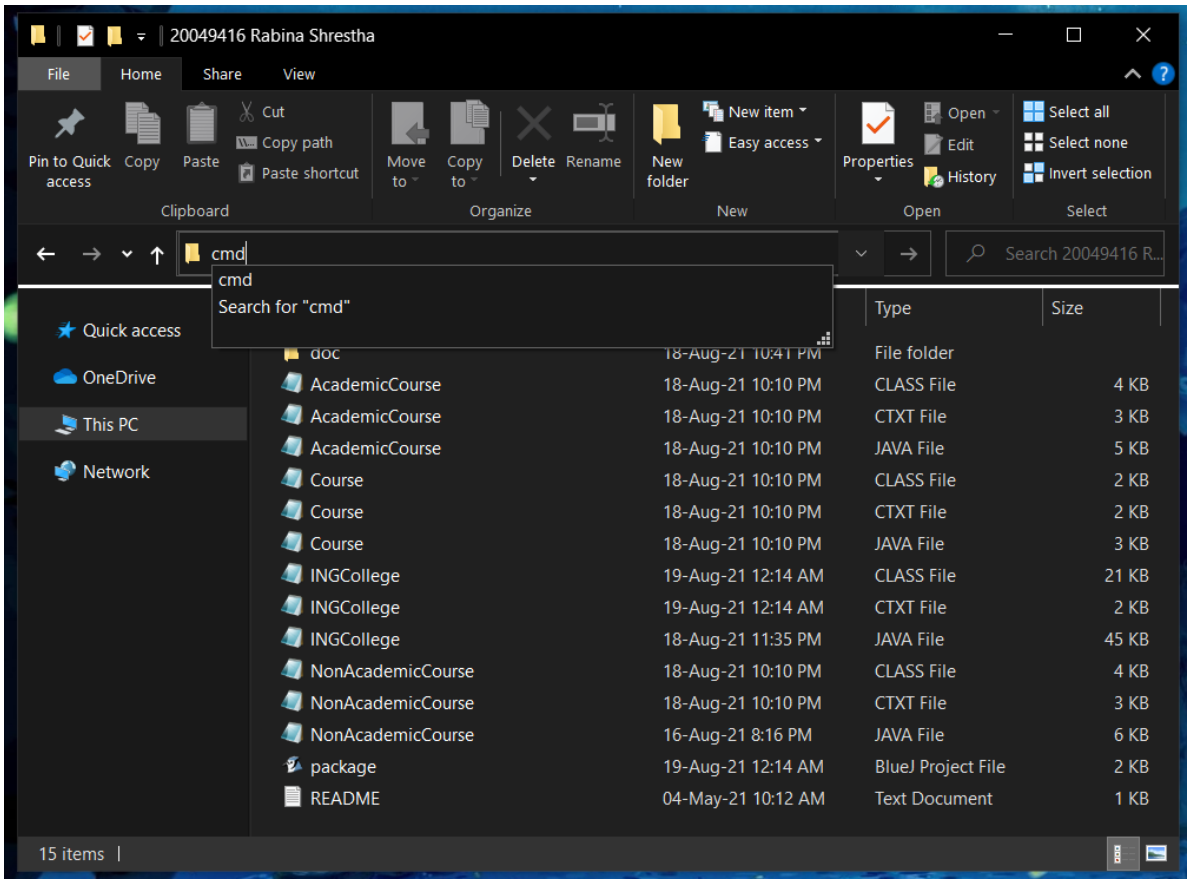

Figure 2: Browsing through the folder and opening cmd using the address bar.

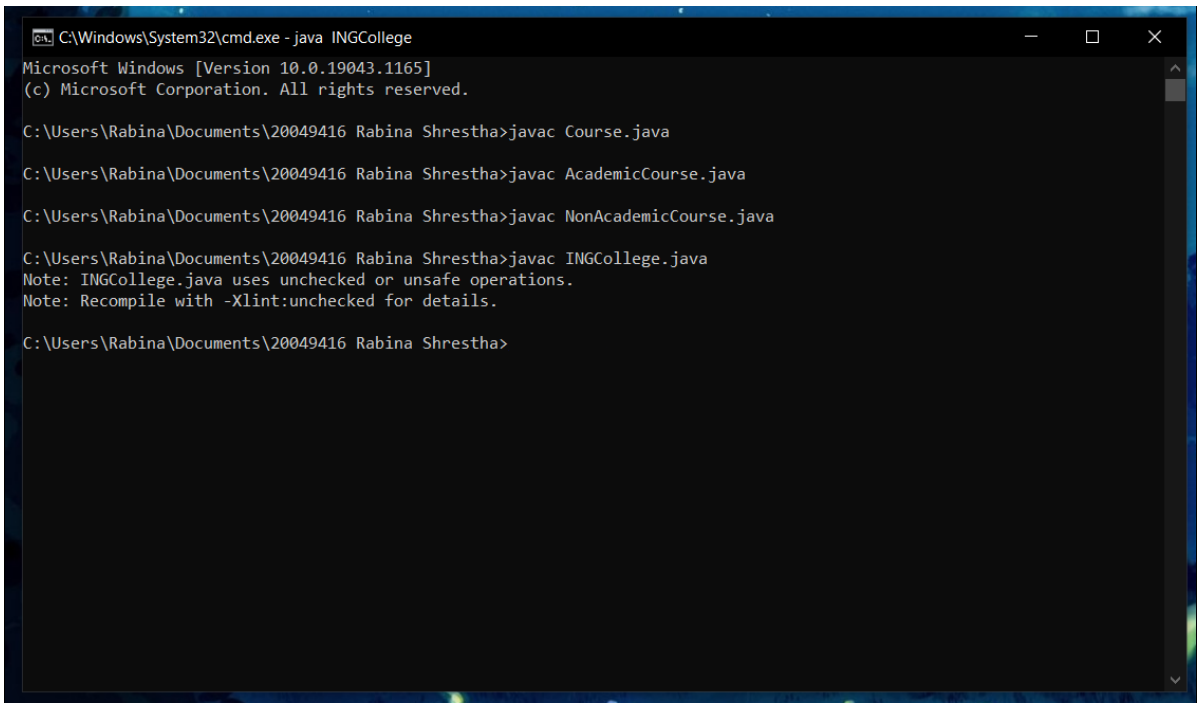

Figure 3: Compiling the classes in cmd.

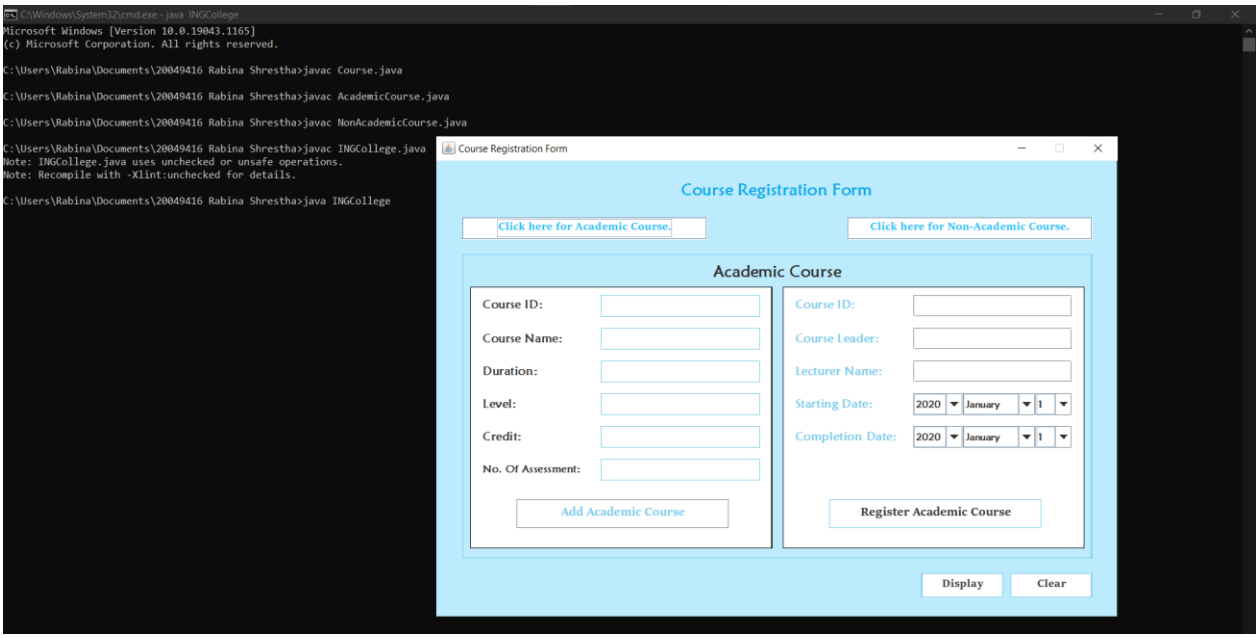

Figure 4: Running the INGCollege class in cmd.

5.2 Test 2: Adding, Registering, and Removing Courses.

Test 2.1: Add course for Academic course.

Test No.	**2.1**
Objective:	To test the working of Add Academic Course.
Action:	➢ Open the BlueJ project and right click INGCollege class. ➢ Run the main method of INGCollege. ➢ Fill out the Academic Course form on the left-hand side with values as the following: ○ Course ID: CS4001NI ○ Course Name: Programming ○ Duration: 3 ○ Level: 4 ○ Credit: 30 ○ No. Of Assessment: 2 ➢ Click on the "Add Academic Course" button once you have filled every field.
Expected Result:	A message box should pop up saying, "The following values has been added to Academic Course", with the values being the ones typed above.
Actual Result:	A message box popped up along with the input values.
Conclusion:	The test was successful.

Table 3: Test 2.1: Add course for Academic course.

Evidence:

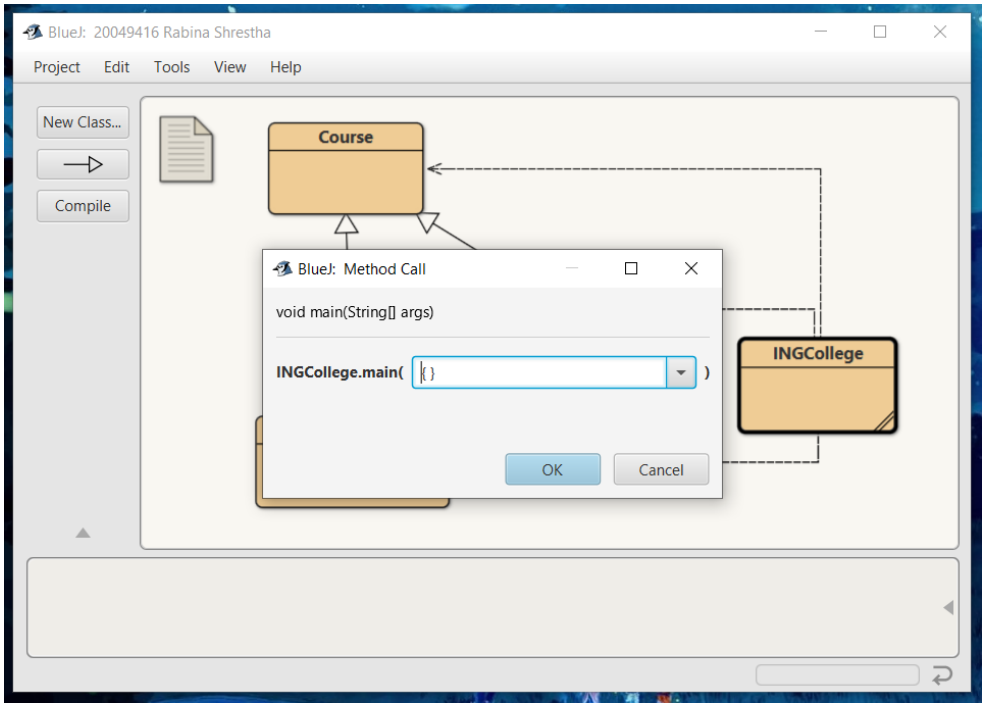

Figure 5: Running the main method of INGCollege class.

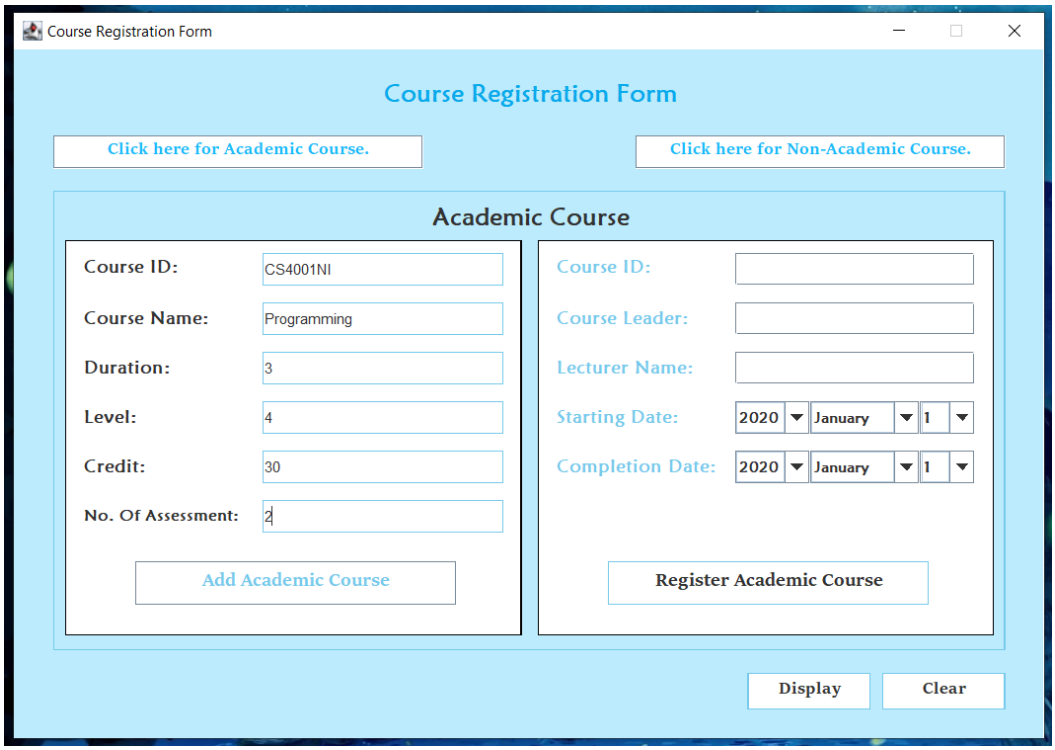

Figure 6: Filling out the Academic Add Course form.

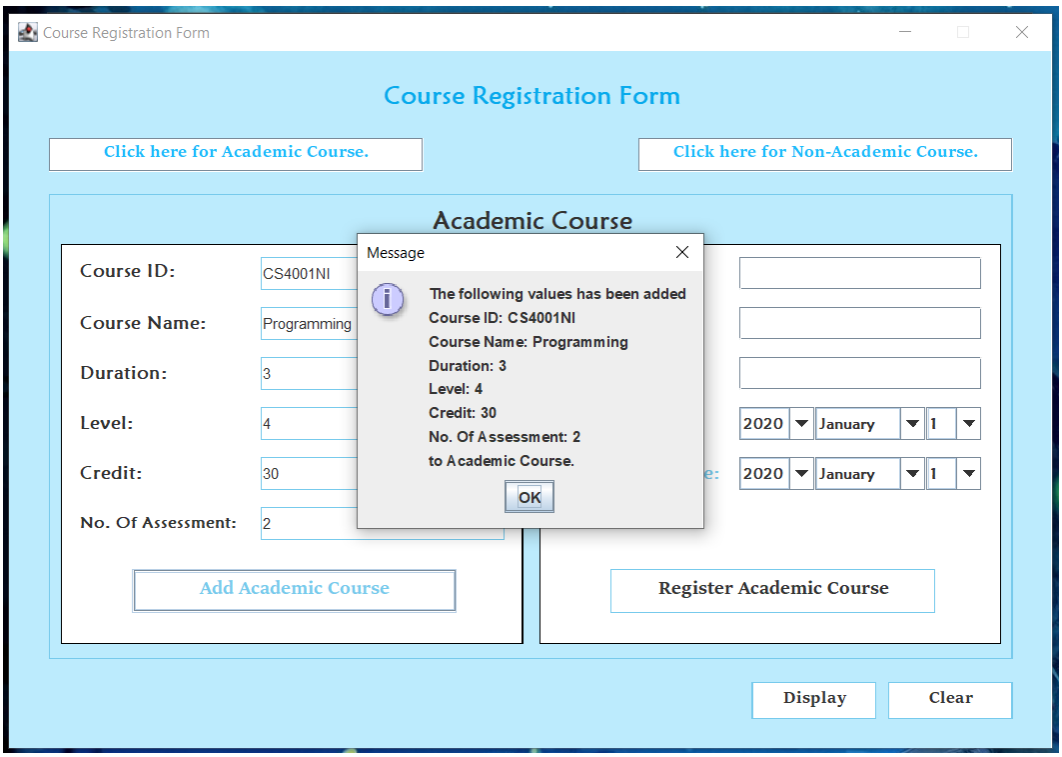

Figure 7: Course has been successfully added.

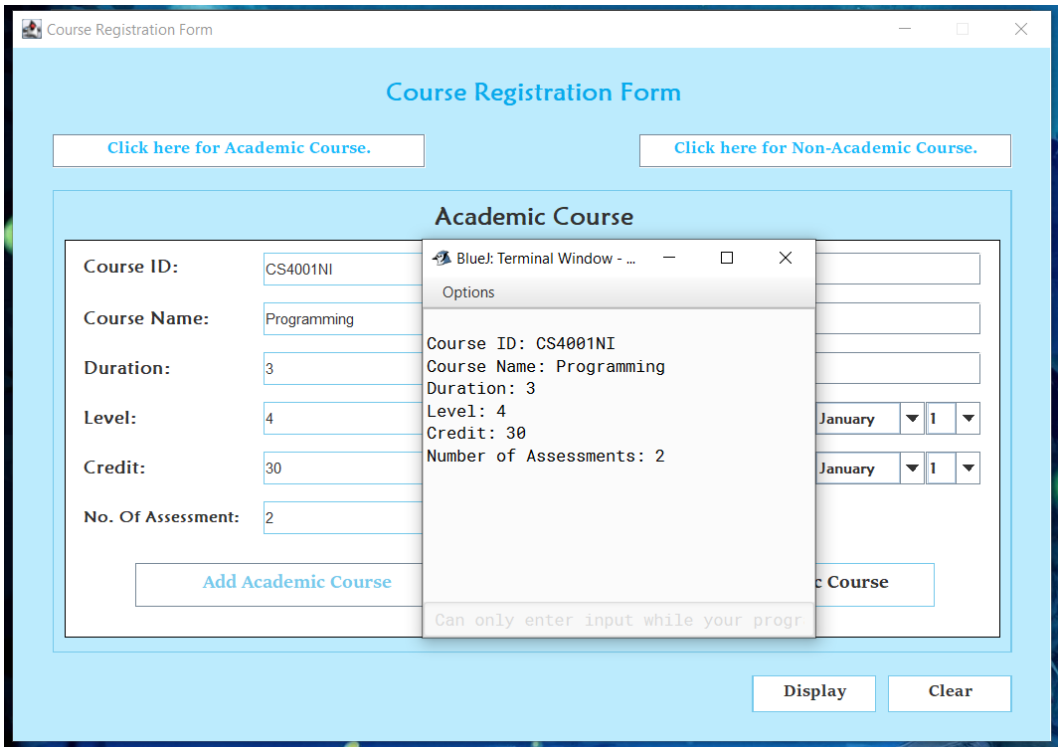

Figure 8: Display after Adding Academic Course.

Test 2.2: Add course for Non-Academic Course.

Test No.	**2.2**
Objective:	To test the working of Add Non-Academic Course.
Action:	➢ Open the BlueJ project and right click INGCollege class. ➢ Run the main method of INGCollege. ➢ Click on the "Click here for Non-Academic Course" button. ➢ Fill out the Non-Academic Course form on the left-hand side with values as the following: ○ Course ID: CS4002NI ○ Course Name: Programming Tutorial ○ Duration: 1 ○ Prerequisite: Debugging Skills ➢ Click on the "Add Non-Academic Course" button once you have filled every field.
Expected Result:	A message box should pop up saying, "The following value has been added to Non-Academic Course", with the values being the ones typed above.
Actual Result:	A message box popped up along with the input values.
Conclusion:	The test was successful.

Table 4: Test 2.2: Add course for Non-Academic Course.

Evidence:

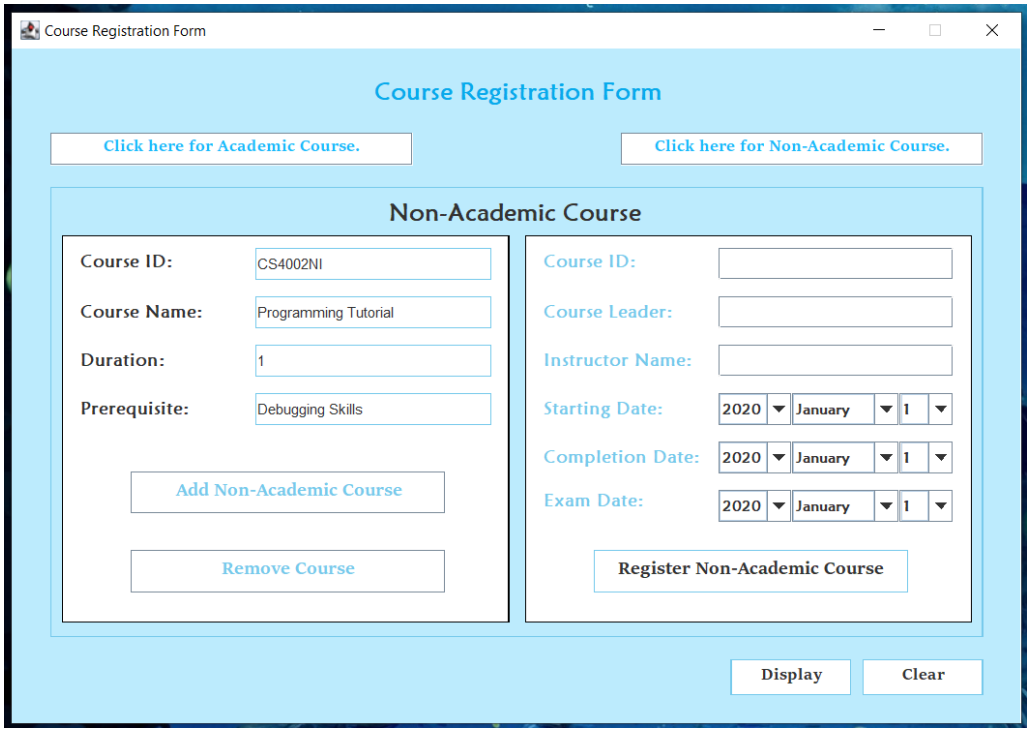

Figure 9: Filling out the Add Non-Academic Form.

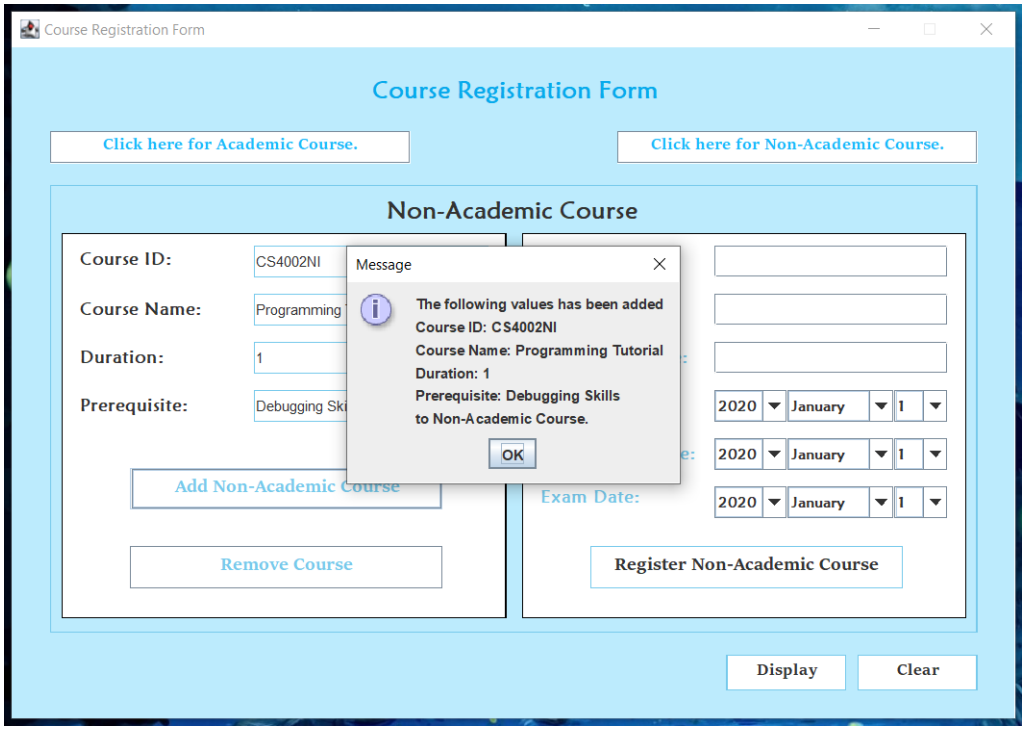

Figure 10: Course has been successfully added.

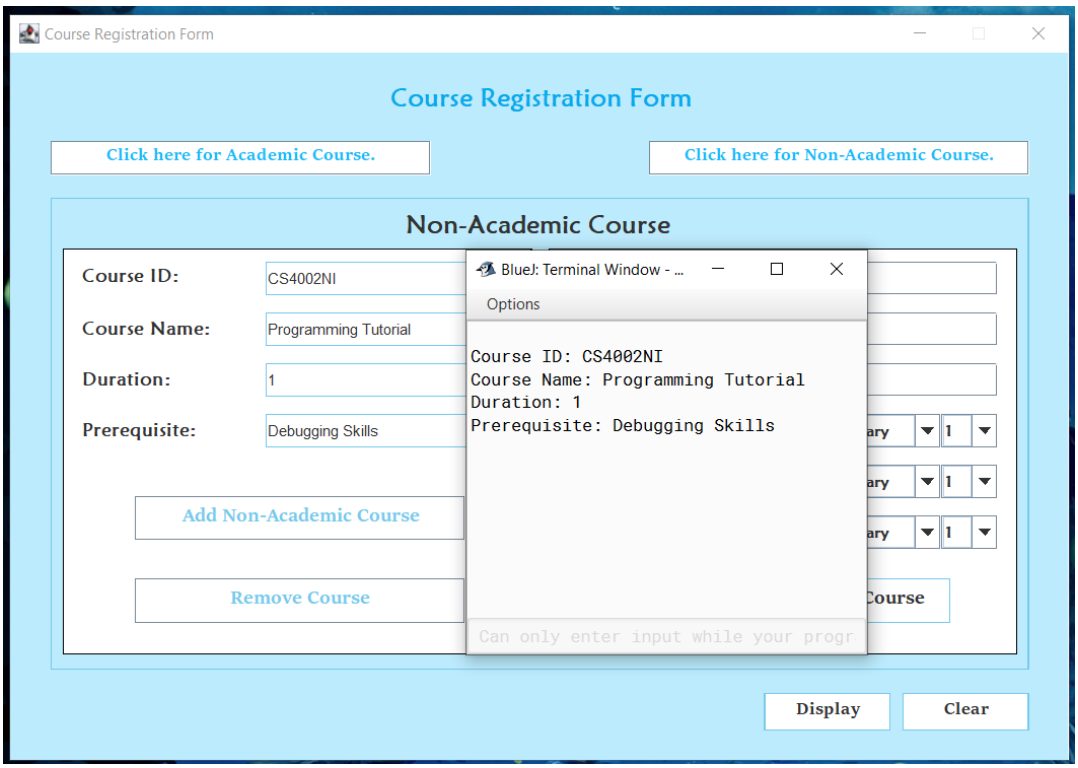

Figure 11: Display after Adding Non-Academic Course.

Test 2.3: Register Academic Course.

Test No.	**2.3**
Objective:	To test the working of Register Academic Course.
Action:	➢ Once Academic Course has been added. Fill out the right-hand side of the Academic Course form with values as the following: ○ Course ID: CS4001NI ○ Course Leader: Dhruba Sen ○ Lecturer Name: Prithivi Maharjan ○ Starting Date: 2020, October, 18 ○ Completion Date: 2021, November, 27 ➢ Click on the “Register Academic Course” button.
Expected Result:	A message box should pop up saying, “The Academic Course has been registered. Values”, with the values being the ones typed above.
Actual Result:	A message box popped up along with the along with the input values.
Conclusion:	The test was successful.

Table 5: Test 2.3: Register Academic Course.

Evidence:

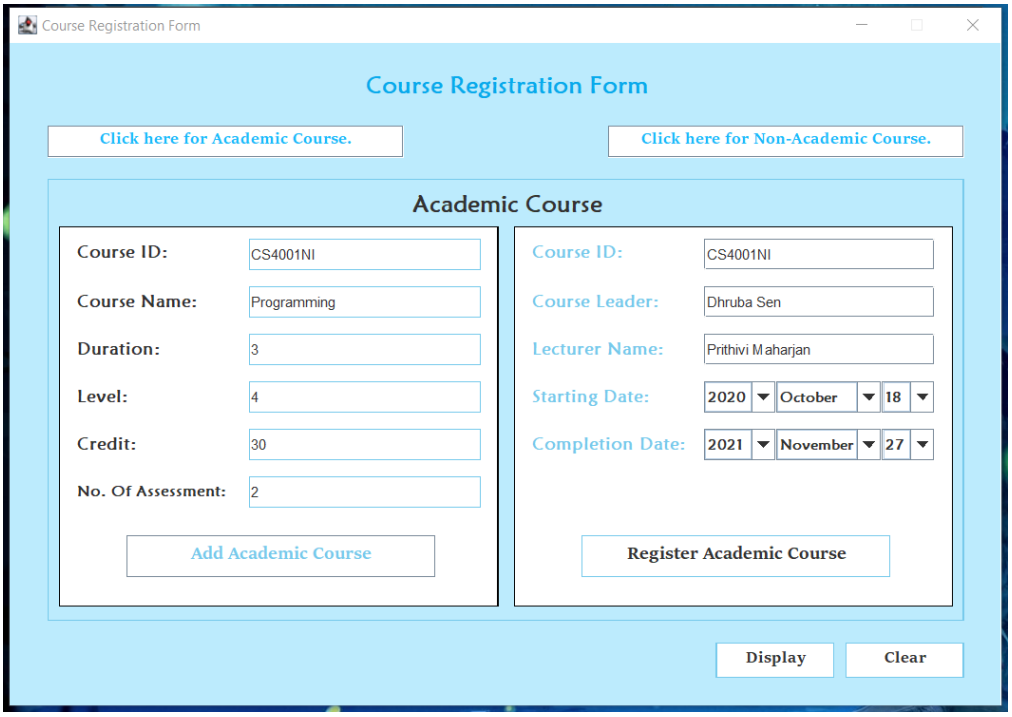

Figure 12: Filling out the Register Academic Form.

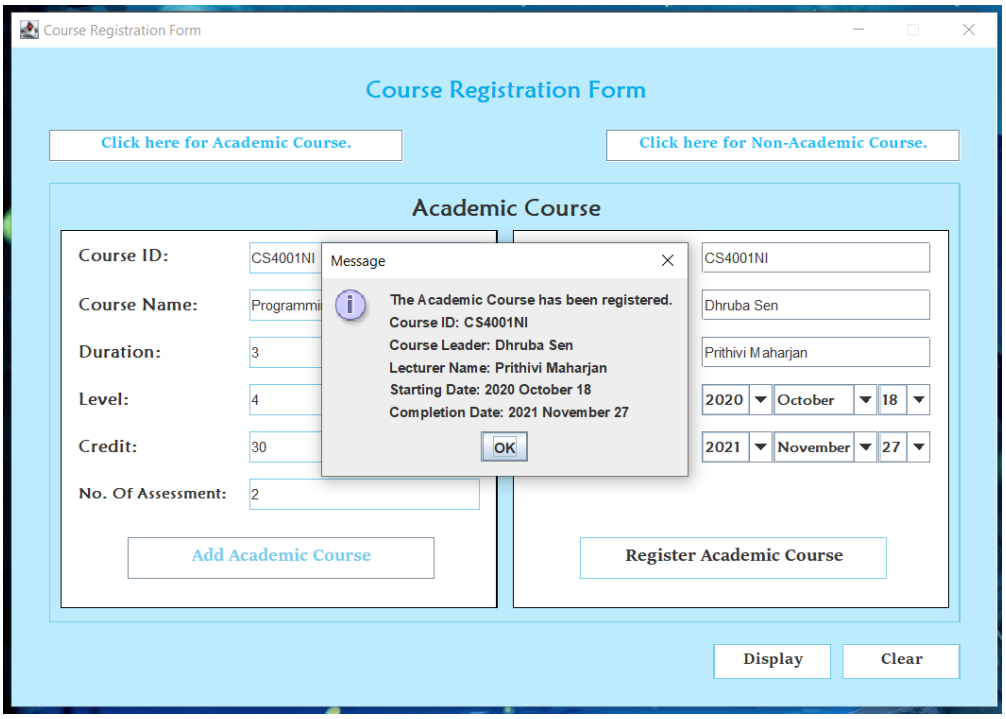

Figure 13: Course has been successfully registered.

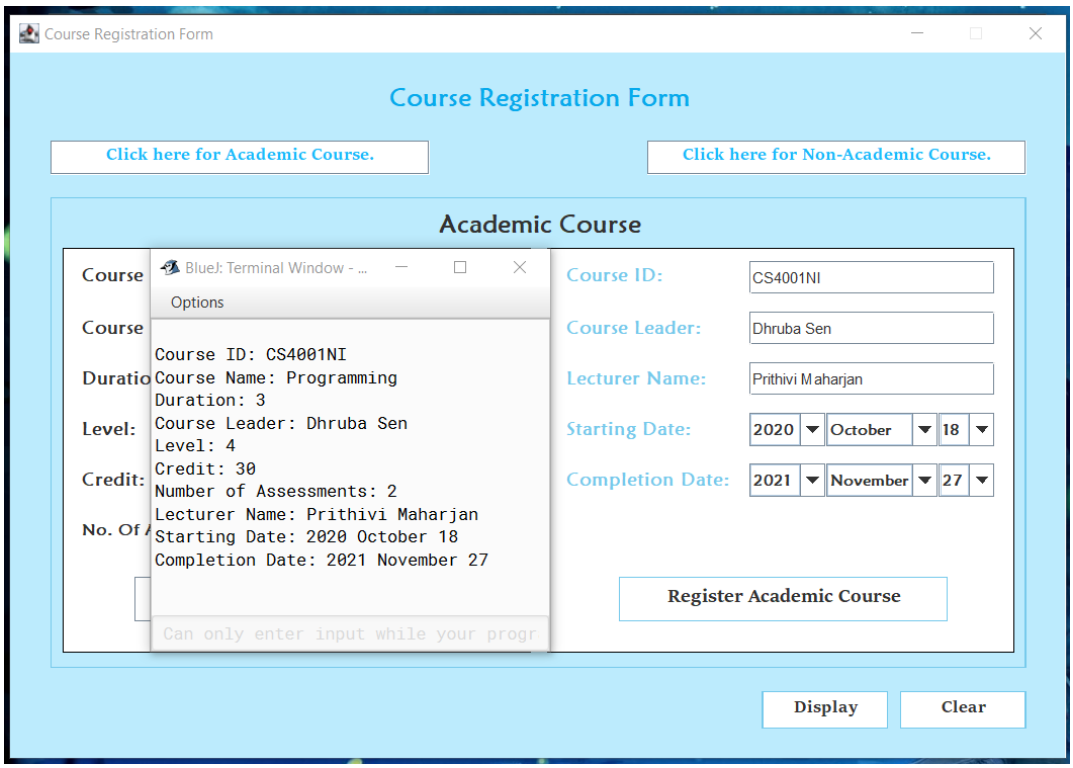

Figure 14: Display after Registering Academic Course.

Test 2.4: Register Non-Academic course.

Test No.	**2.4**
Objective:	To test the working of Register Non-Academic Course.
Action:	➢ Once Non-Academic Course has been added. Fill out the right-hand side of the Non-Academic Course form with values as the following: ○ Course ID: CS4002NI ○ Course Leader: Dhruba Sen ○ Instructor Name: Ujjwal Subedi ○ Starting Date: 2020, October, 18 ○ Completion Date: 2021, November, 18 ○ Exam Date: 2021, November, 27 ➢ Click on the “Register Non-Academic Course” button.
Expected Result:	A message box should pop up saying, “The Non-Academic Course has been registered. Values”, with the values being the ones typed above.
Actual Result:	A message box popped up along with the along with the input values.
Conclusion:	The test was successful.

Table 6: Test 2.4: Register Non-Academic course.

Evidence:

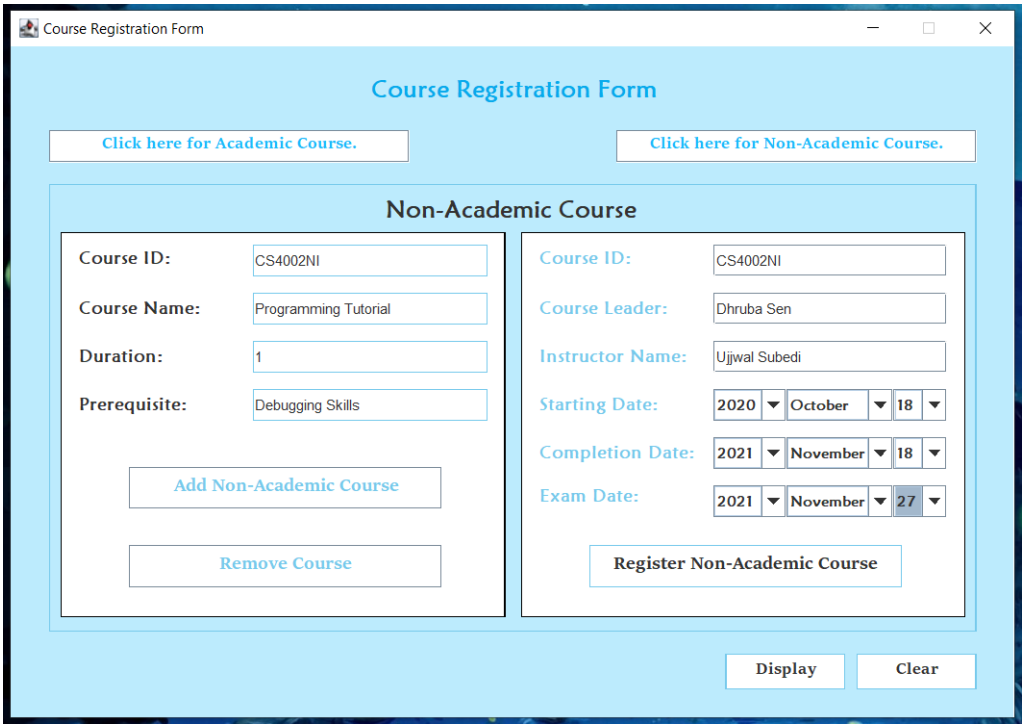

Figure 15: Filling out the Register Non-Academic Form.

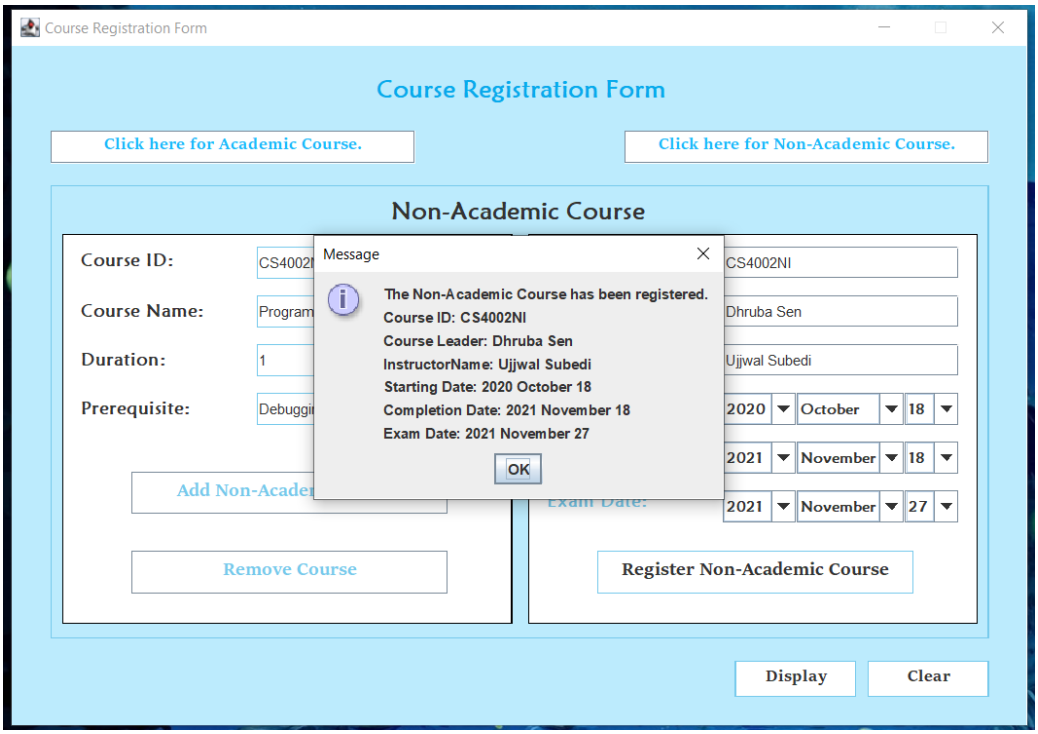

Figure 16: Course has been successfully registered.

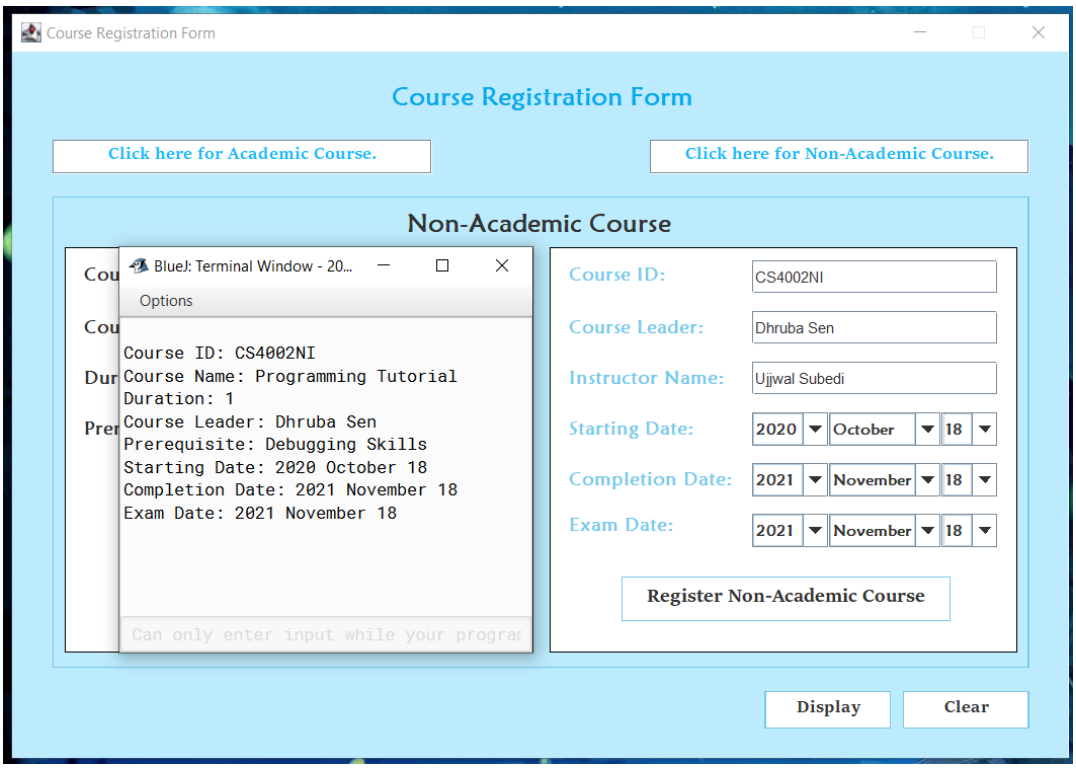

Figure 17: Display after Registering Non-Academic Course.

Test 2.5: Remove Non-Academic course.

Test No.	**2.5**
Objective:	To test the working of Remove Non-Academic Course.
Action:	➢ Once Non-Academic Course has been added and registered. ➢ Click the “Remove Course” button.
Expected Result:	A message box should pop up saying, “The Non-Academic Course with Course ID: _ has been removed.”, with the _ being the Course ID of Non-Academic Course.
Actual Result:	A message box popped up along with the Course ID.
Conclusion:	The test was successful.

Table 7: Test 2.5: Remove Non-Academic course.

Evidence:

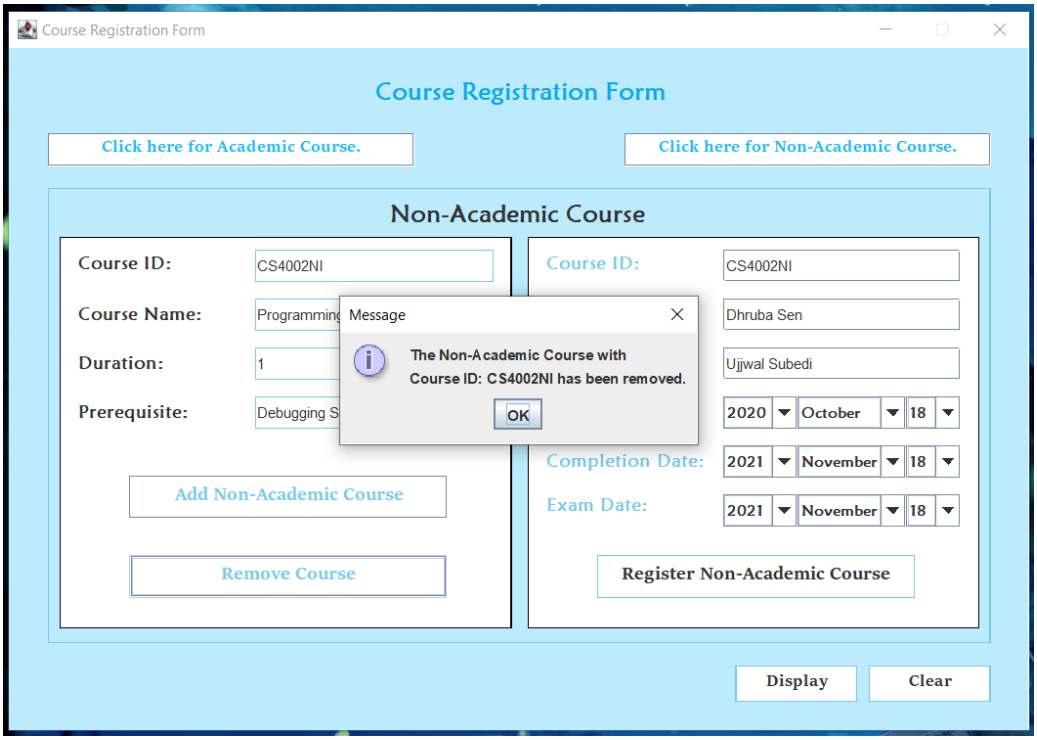

Figure 18: Non-Academic Course has been removed.

5.3 Test 3: Testing appropriate dialog boxes.

Test 3.1: Dialog box while trying to add duplicate Course ID.

Test No.	**3.1**
Objective:	To test the working of dialog box while trying to add duplicate Course ID.
Action:	Academic Course ➢ Fill out the Academic Course form on the left-hand side with values as the following: ○ Course ID: CS4001NI ○ Course Name: Programming ○ Duration: 3 ○ Level: 4 ○ Credit: 30 ○ No. Of Assessment: 2 ➢ Click on the “Add Academic Course” button to add course. ➢ Click the “Add Academic Course” button once again. Non-Academic Course ➢ Fill out the Non-Academic Course form on the left-hand side with values as the following: ○ Course ID: CS4001NI ○ Course Name: Programming ○ Duration: 3 ○ Level: 4 ○ Credit: 30 ○ No. Of Assessment: 2 ➢ Click on the “Add Non-Academic Course” button to add course. ➢ Click the “Add Non-Academic Course” button once again.

Expected Result:	A message box should pop up saying, “The Course ID: _ with Course Name: _ has already been added”, with the _ being the Course ID and Course Name given.
Actual Result:	A message box popped up along with the along with the input values.
Conclusion:	The test was successful.

Table 8: Test 3.1: Dialog box while trying to add duplicate Course ID.

Evidence:

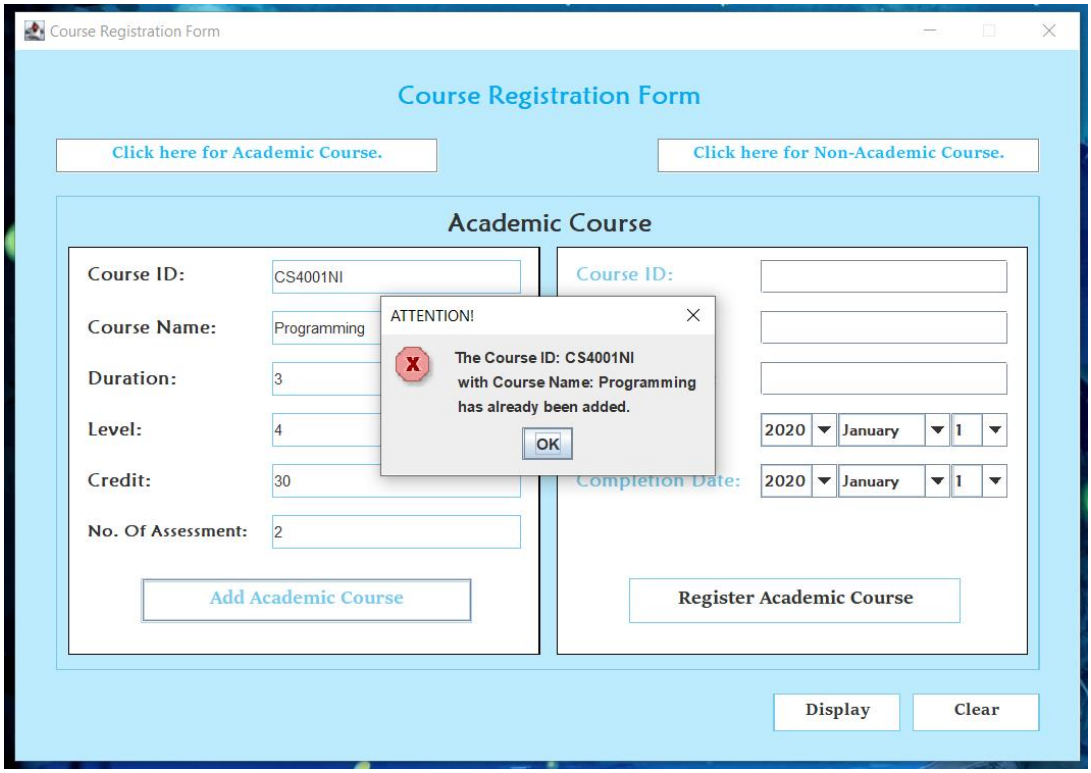

Figure 19: Trying to add duplicate Course ID in Academic Course.

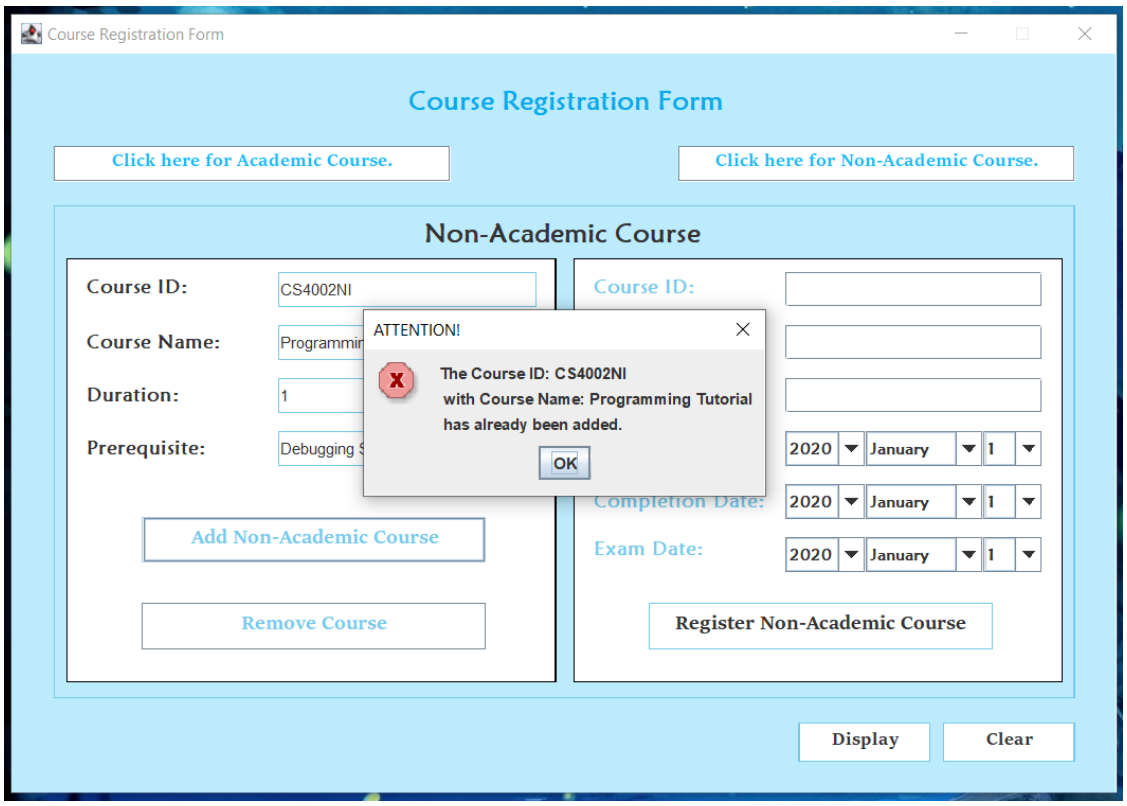

Figure 20: Trying to add duplicate Course ID in Non- Academic Course.

Test 3.2: Trying to Register already Registered Course.

Test No.	**3.2**
Objective:	To test the working of dialog box while trying to register already registered course.
Action:	Academic Course ➢ Fill out the Academic Course form on the right-hand side with values as the following: ○ Course ID: CS4001NI ○ Course Leader: Dhruba Sen ○ Lecturer Name: Prithivi Maharjan ○ Starting Date: 2020, October, 18 ○ Completion Date: 2021, November, 27 ➢ Click on the “Register Academic Course” button to register course. ➢ Click the “Register Academic Course” button once again. Non-Academic Course ➢ Fill out the Non-Academic Course form on the right-hand side with values as the following: ○ Course ID: CS4002NI ○ Course Leader: Dhruba Sen ○ Instructor Name: Ujjwal Subedi ○ Starting Date: 2020, October, 18 ○ Completion Date: 2021, November, 18 ○ Exam Date: 2021, November, 27 ➢ Click on the “Register Non-Academic Course” button to register course. ➢ Click the “Register Non-Academic Course” button once again.

Expected Result:	A message box should pop up saying, “This course is already registered”.
Actual Result:	A message box popped up.
Conclusion:	The test was successful.

Table 9: Test 3.2: Trying to Register already Registered Course.

Evidence:

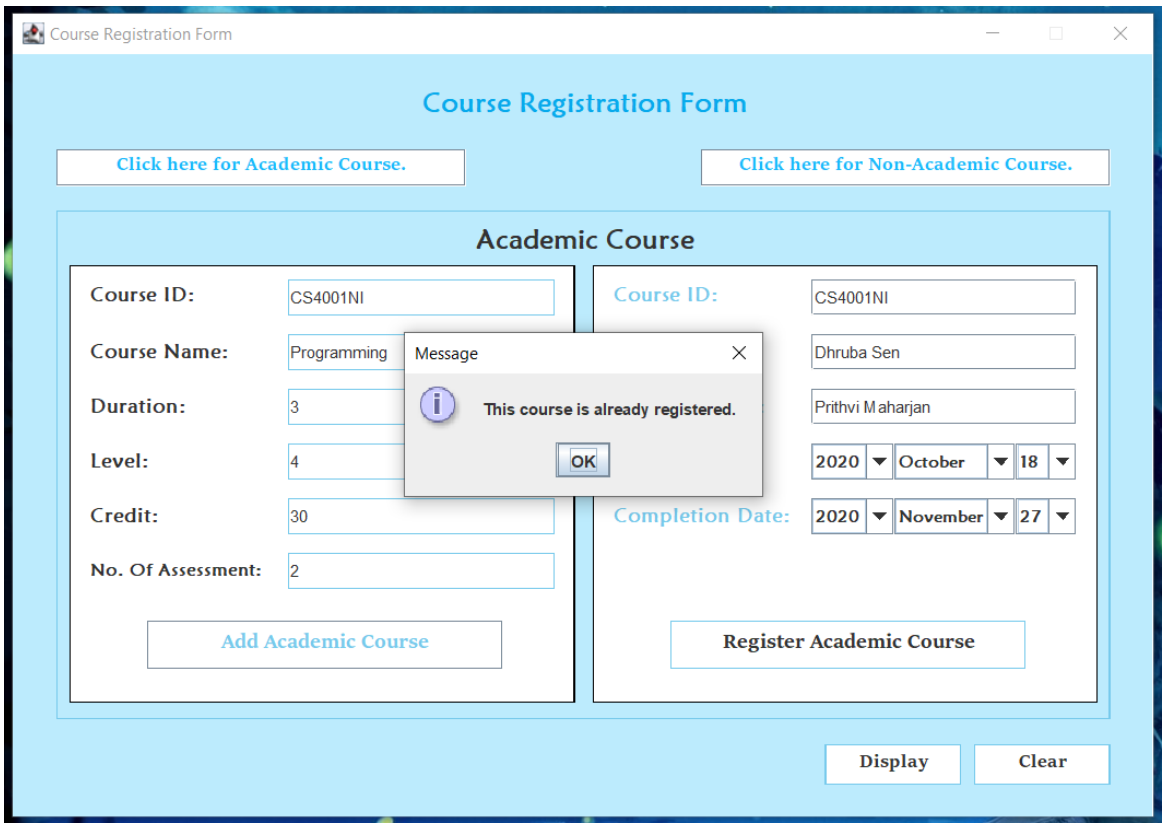

Figure 21: Trying to Register already Registered Academic Course.

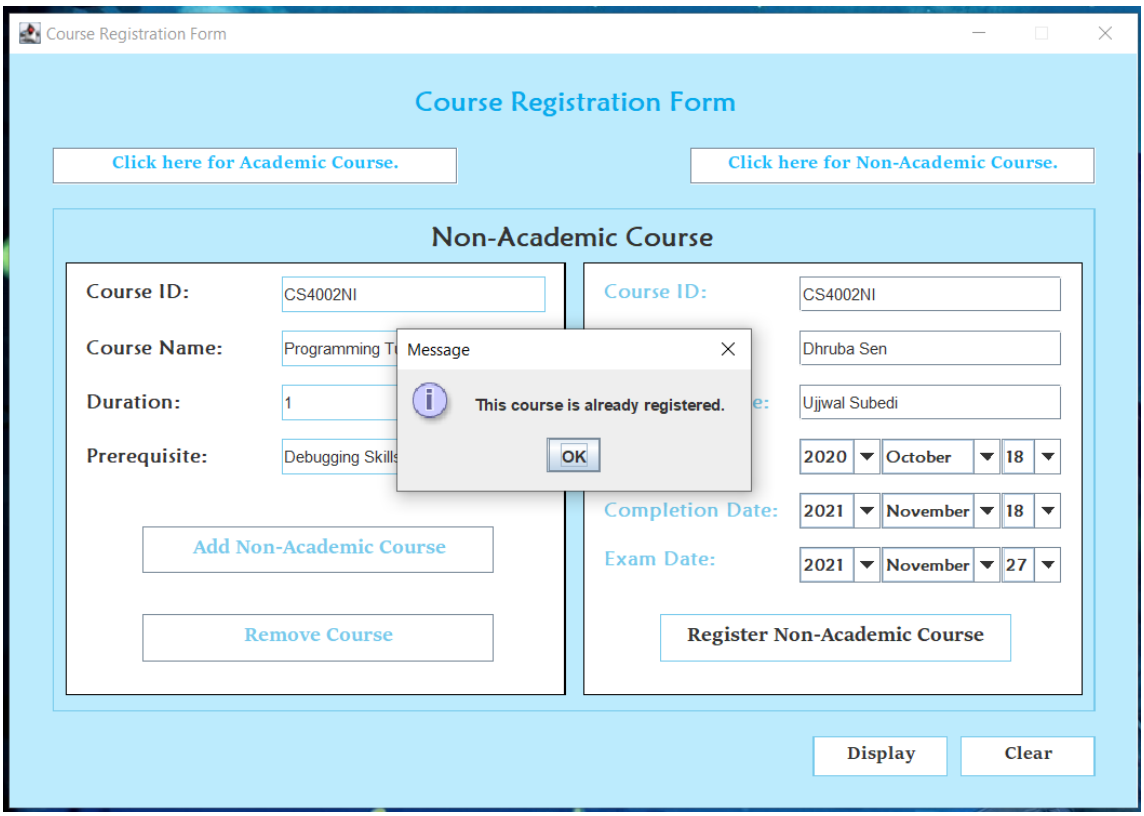

Figure 22: Trying to Register already Registered Non-Academic Course.

Test 3.3: Trying to remove the Non-Academic Course which is already removed.

Test No.	**3.3**
Objective:	To test the working of dialog box while trying to Remove Academic Course which is already removed.
Action:	➢ Fill out the Non-Academic Course form on the left-hand side with values as the following: ○ Course ID: CS4001NI ○ Course Name: Programming ○ Duration: 3 ○ Level: 4 ○ Credit: 30 ○ No. Of Assessment: 2 ➢ Click on the "Add Non-Academic Course" button to add course. ➢ Fill out the Non-Academic Course form on the right-hand side with values as the following: ○ Course ID: CS4002NI ○ Course Leader: Dhruba Sen ○ Instructor Name: Ujjwal Subedi ○ Starting Date: 2020, October, 18 ○ Completion Date: 2021, November, 18 ○ Exam Date: 2021, November, 27 ➢ Click on the "Register Non-Academic Course" button to register course. ➢ Click on "Remove Course" Button two times.
Expected Result:	A message box should pop up saying, "The Non-Academic Course with Course ID: _ has already been removed.", with the _ being the Course ID typed above.

Actual Result:	A message box popped up along with the Course ID.
Conclusion:	The test was successful.

Table 10: Test 3.3: Trying to remove the Non-Academic Course which is already removed.

Evidence:

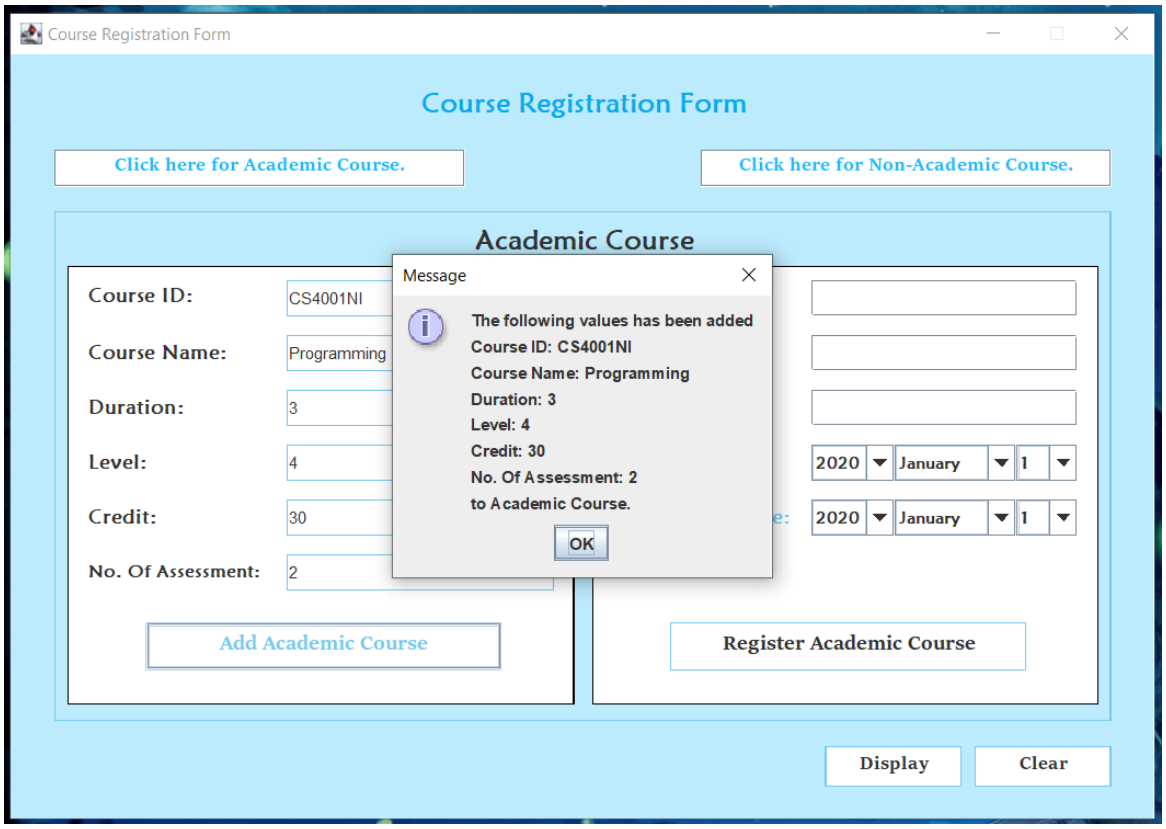

Figure 23: Adding Non-Academic Course.

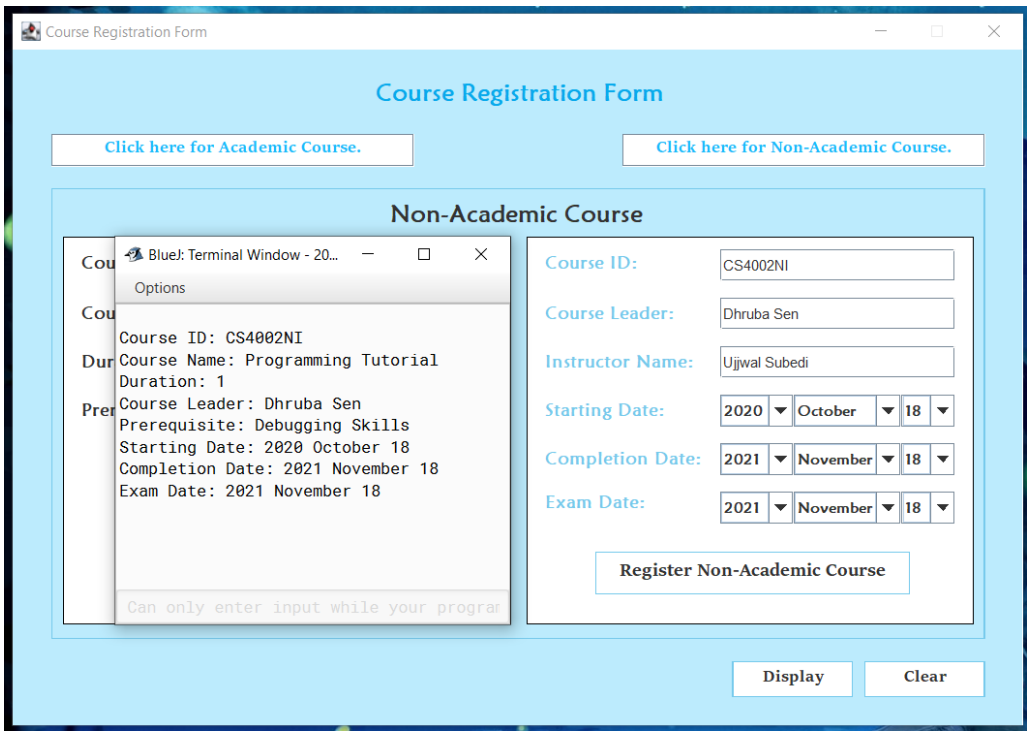

Figure 24: Registering Non-Academic Course.

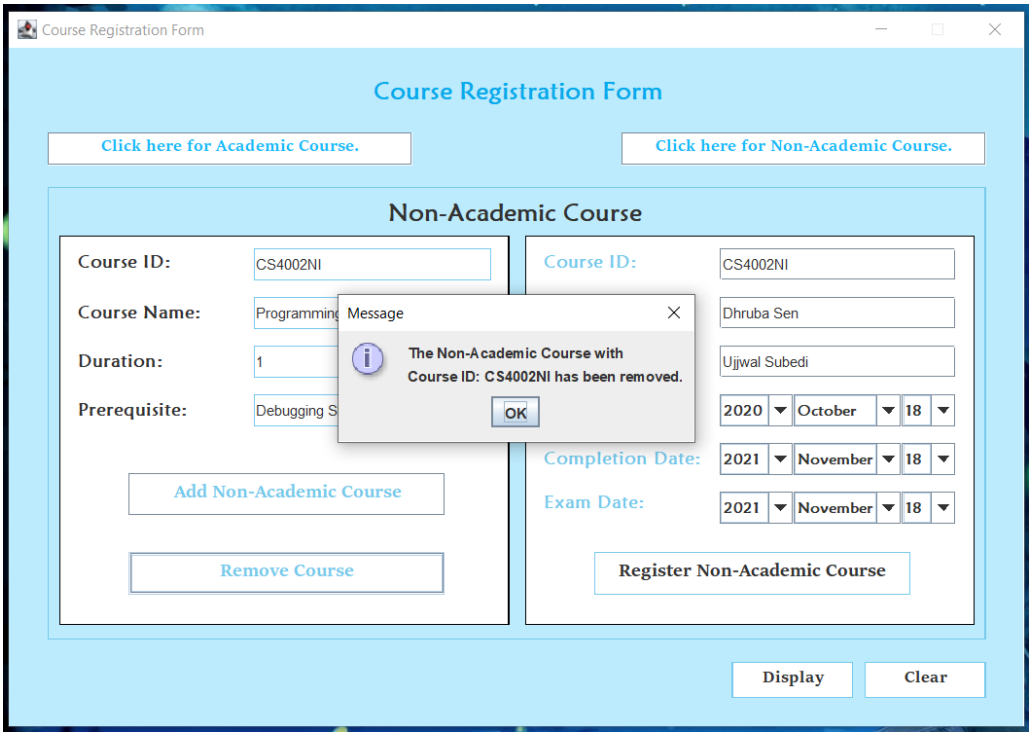

Figure 25: Removing Non-Academic Course.

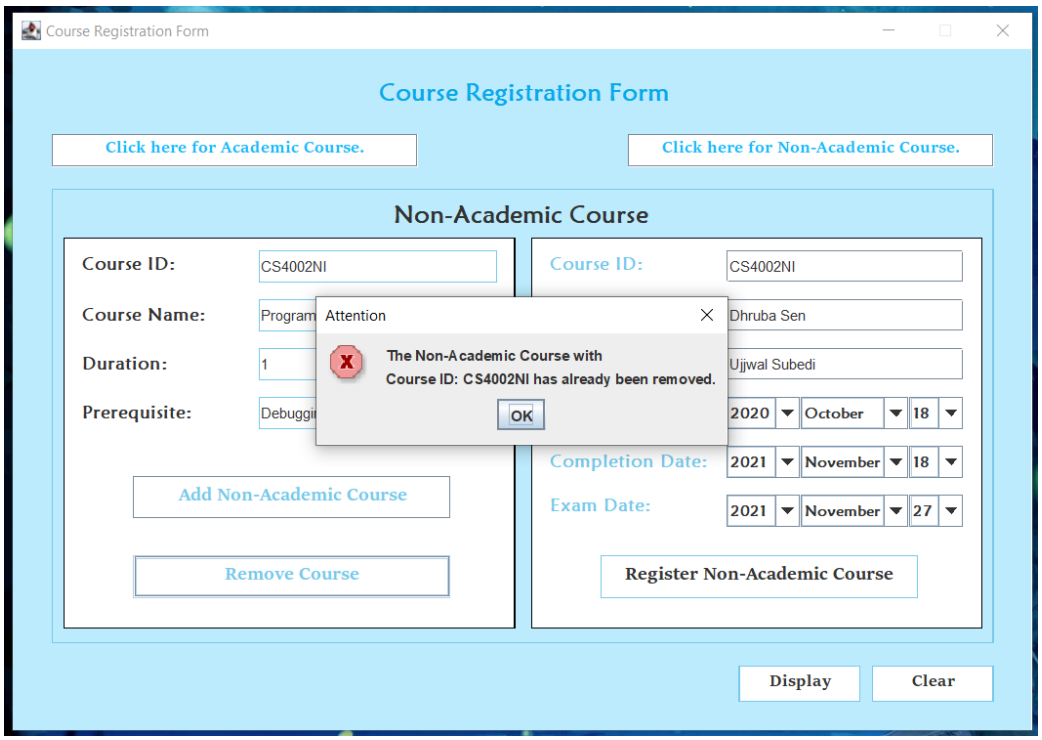

Figure 26: Trying to Remove Non-Academic Course again.

6. Errors:

Having some errors in the program could prevent proper execution of the program or may give an incorrect output. Although errors are bad, experiencing them develops skills in the long run. A programmer starts to get comfortable crossing the bugs they have created and quickly fixes them. (Shrestha, 2021)

Here are some errors I dealt with, while programming the coursework:

6.1. Error 1: Syntax Error.

Finding the error:

```
    JOptionPane.showMessageDialog(frame, "The Duration must be an integer number.","ATTENTION!", JOptionPane.ERROR_MESSAGE);
    count2 = 1;
}
if (count2 == 0)
{
    JOptionPane.showMessageDialog(frame, "The following values has been added \n"+ "Course ID: " + getNACourseID() + "\n"+
                                  "Course Name: " + getNACourseName() "\n"+ "Duration: " + getNACDuration() + "\n" +
                                  "Prerequisite: " + getPrerequisite() + "\n" + "to Non-Academic Course.");
    NonAcademicObject = new NonAcademicCourse(getNACourseID(), getNACourseName(), getNACDuration(), getPrerequisite());
    list_Course.add(NonAcademicObject);
}
```

Figure 27: Syntax error.

Description of the error: While writing the program, in order to change lines + “\n” + had being added but due to carelessness I forgot to add + which caused a syntax error, which can be seen in the above Figure 27.

Analyzing and solving the error:

```
    JOptionPane.showMessageDialog(frame, "The Duration must be an integer number.","ATTENTION!", JOptionPane.ERROR_MESSAGE);
    count2 = 1;
}
if (count2 == 0)
{
    JOptionPane.showMessageDialog(frame, "The following values has been added \n"+ "Course ID: " + getNACourseID() + "\n"+
                                  "Course Name: " + getNACourseName() + "\n"+ "Duration: " + getNACDuration() + "\n" +
                                  "Prerequisite: " + getPrerequisite() + "\n" + "to Non-Academic Course.");
    NonAcademicObject = new NonAcademicCourse(getNACourseID(), getNACourseName(), getNACDuration(), getPrerequisite());
    list_Course.add(NonAcademicObject);
}
```

Figure 28: Solving the error.

How to solve: Add + which is required for the code to work. This type of error is known as “**Syntax Error**” which is caused due to the programmer not following the syntax of the programming language.

6.2. Error 2: Semantic Error.

Finding the error:

```
public String getACDuration()
{
    return Integer.parseInt(this.field_Duration.getText());
}
```

Figure 29: Semantic Error.

Description of the error: While writing the program, the data type of ACDuration was written as String instead of the data type int, the error can be seen in the above Figure 29.

Analyzing and solving the error:

```
public int getACDuration()
{
    return Integer.parseInt(this.field_Duration.getText());
}
```

Figure 30: Solving the error.

How to solve: Assign the correct data type to ACDuration, which is int. This type of error is known as "**Semantic Error**" which is caused due to the programmer's improper use of the Java statements.

6.3. Error 3: Logical Error.

Error:

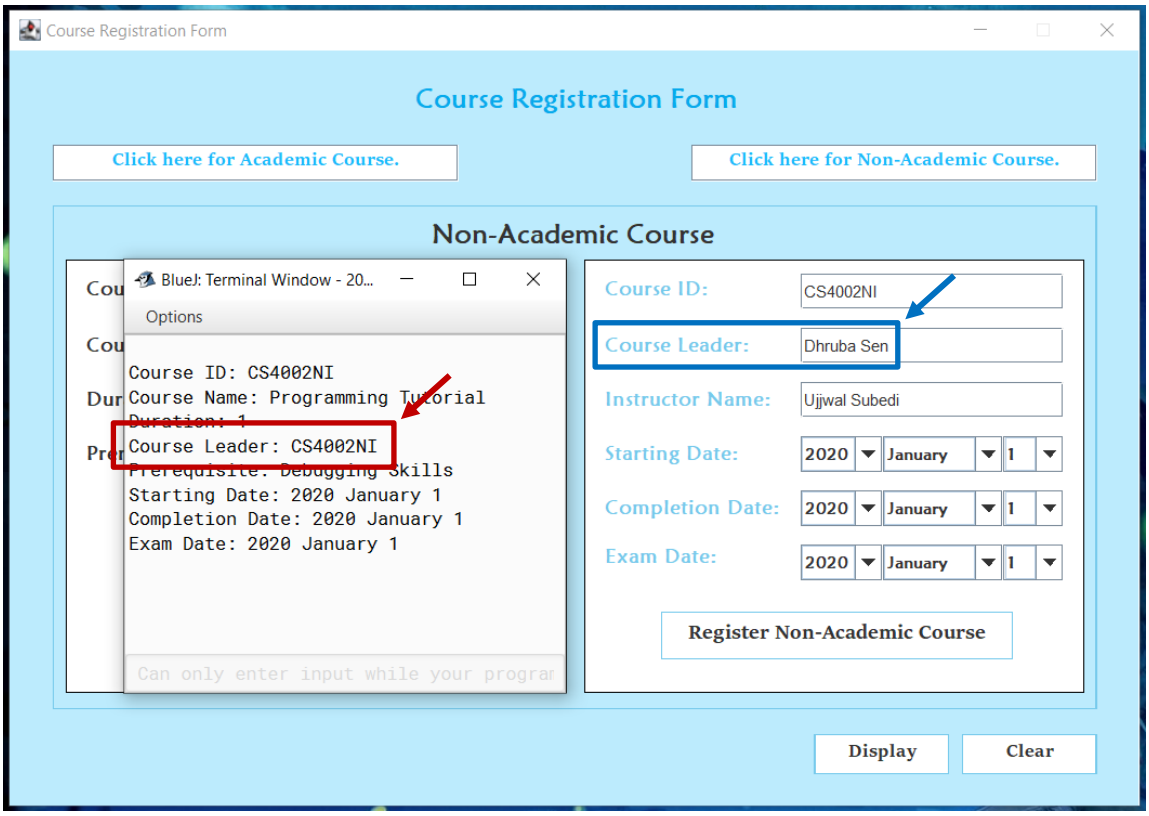

Figure 31: Logical Error.

Finding the error:

```
else if (NonAcademicObject.getisRegistered() == false)
{
    NonAcademicObject.register(getNACourseRegisterID(), getInstructorName(), getNACStartDate() ,getNACCompleteDate(),
                               getExamDate());
    JOptionPane.showMessageDialog(frame, "The Non-Academic Course has been registered. \n"
                                + "Course ID: " + getNACourseRegisterID() + "\n"
                                + "Course Leader: " + getNACourseLeader() + "\n"
                                + "InstructorName: " + getInstructorName() + "\n"
                                + "Starting Date: " + getNACStartDate() + "\n"
                                + "Completion Date: " + getNACCompleteDate() + "\n"
                                + "Exam Date: " + getExamDate());
    count = 1;
}
```

Figure 32: Error found.

Description of the error: The output that should have been printed on the “Course Leader: “should have been Course Leader’s Name which is Dhruba Sen as pointed by the blue

box and arrow, not the Course ID. The error was due to a mistake which can be seen in the above Figure 31.

Solving the error:

```
else if (NonAcademicObject.getisRegistered() == false)
{
    NonAcademicObject.register(getNACourseLeader(), getInstructorName(), getNACStartDate() ,getNACCompleteDate(),
                                getExamDate());
    JOptionPane.showMessageDialog(frame, "The Non-Academic Course has been registered. \n"
                                + "Course ID: " + getNACourseRegisterID() + "\n"
                                + "Course Leader: " + getNACourseLeader() + "\n"
                                + "InstructorName: " + getInstructorName() + "\n"
                                + "Starting Date: " + getNACStartDate() + "\n"
                                + "Completion Date: " + getNACCompleteDate() + "\n"
                                + "Exam Date: " + getExamDate());
    count = 1;
}
```

Figure 33: Solving the error.

Executing the program and now the desired output is shown:

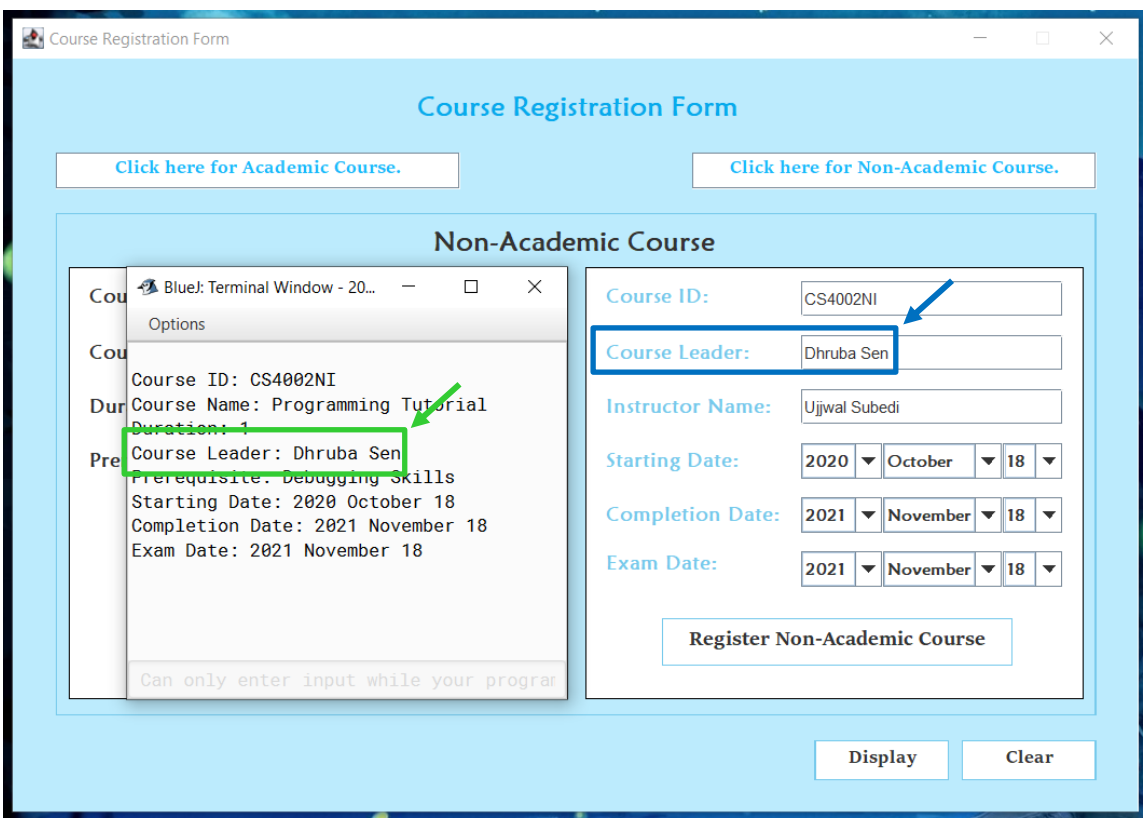

Figure 34: Error Solved.

How to solve: The error was found and was corrected immediately by changing the NACourseRegisterID to NACourseLeader and the desired output was shown. This type of error is known as “**Logical Error**” which is caused due to mistake in logic. The program was compiled and executed without showing any errors, but the output did not generate the requested result. Logical errors are hardest to track due to the above reason.

7. Conclusion.

The report is based on a project given to the students in hopes for us to add a class to the project that we developed for the first part of the coursework. This project specifically aims at the usage of GUI (Graphical User Interface) along with its other functions as a system to store the details of Course that includes both Academic and Non-Academic Course. After understanding the object-oriented concepts of Java, we were able to implement the use of GUI in a real-life scenario as well. Although restricted by errors and mistakes along the way, with the help of teachers, the lecture slides, recordings provided to us and sites we were able to see the end of it.

Through this project we were able to learn and develop our skills in the field of Java GUI. Pseudocode helped us create a reference sketch of the code which made the coding step considerably easier and the data flow clear. Complications arose like using exception handlings, and validating the form in a way that was user-friendly. With my limited knowledge on exceptions, I was faced with a problem regarding pop up. One click could trigger a series of errors at the same time without letting the user fix it. Downcasting and upcasting objects to access different methods and to add objects to the arraylist was a new concept and hence, caused confusions.

At last, it was a challenging experience yet fun nonetheless. We were fully allowed to tap into our aesthetics while designing the format but at the same time the codes kept us grounded to the technical aspects. We learnt to acknowledge the particular approaches in which GUI could be made in different ways.

8. Appendix1.

INGCollege.

```
/**
 * INGCollege class represents the GUI of Course.
 *
 * @author (Rabina Shrestha)
 * @version (5.0.0)
 */

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;

public class INGCollege implements ActionListener
{
    /* Declaring Instance Variables.
       ArrayList */
    private ArrayList<Course> list_Course = new ArrayList<Course>();
    private AcademicCourse AcademicObject;
    private NonAcademicCourse NonAcademicObject;

    // JFrame and JPanel
    private JFrame frame;
    private JPanel panel, panel_Academic, panel_addAC, panel_registerAC,
panel_NonAcademic, panel_addNAC, panel_registerNAC;

    //Course JLabel and JButton
```

```
    private JLabel label_CourseForm;
    private JButton button_clickAC, button_clickNAC;

    /* Academic Course.
       1. JLabel */
    private JLabel label_AcademicCourse, label_CourseID, label_CourseName,
label_Duration, label_Level, label_Credit, label_NoOfAssessment,
            label_ACregisterID, label_CourseLeader, label_LecturerName,
label_ACStartDate, label_ACEndDate;

    //2. JTextField
    private JTextField field_ID, field_CName, field_Duration, field_Level, field_Credit,
field_NoOfAssessment, field_ACregisterID,
            field_ACLeader, field_LName;

    //3. JComboBox
    private JComboBox box_ACYear1, box_ACMonth1, box_ACDay1, box_ACYear2,
box_ACMonth2, box_ACDay2;

    //4. JButton
    private JButton button_addAC, button_registerAC, button_Display;

    //5. String Dates
    private String ACStartingDate, ACCompletionDate;

    /* Non-Academic Course.
       1. JLabel */
    private JLabel label_NonAcademicCourse, label_NACourseID,
label_NACourseName, label_NADuration, label_Prerequisite, label_NACregisterID,
```

```
        label_NACourseLeader, label_InstructorName, label_NACStartDate,
label_NACEndDate, label_ExamDate;

    //2. JTextField
    private JTextField fieldNA_ID, fieldNA_CName, field_NADuration, field_Prerequisite,
field_NACregisterID, field_NACLeader, field_IName;

    //3. JComboBox
    private JComboBox box_NACYear1, box_NACMonth1, box_NACDay1,
box_NACYear2, box_NACMonth2, box_NACDay2, box_NACYear3, box_NACMonth3,
box_NACDay3;

    //4. JButton
    private JButton button_addNAC, button_RemoveNAC, button_registerNAC,
button_NADisplay;

    //5. String Dates
    private String NACStartingDate, NACCompletionDate, ExamDate;

    //Clear JButton.
    private JButton button_Clear;

    //For main method
    static INGCollege main = new INGCollege();

    //Method for GUI
    public void gUI()
    {
        frame = new JFrame();
        panel = new JPanel();
```

```
/* Panels used for Academic Course(AC).
   Academic, addAC, registerAC */
panel_Academic = new JPanel();
panel_addAC = new JPanel();
panel_registerAC = new JPanel();

/* Panels used for Non-Academic Course(NAC).
   NonAcademic, addNAC, registerNAC */
panel_NonAcademic = new JPanel();
panel_addNAC = new JPanel();
panel_registerNAC = new JPanel();

//Layout and Bounds of panels.
panel.setLayout(null);
//Academic
panel_Academic.setBounds(32,115,772,372);
panel_Academic.setLayout(null);
panel_addAC.setBounds(42,155,370,320);
panel_addAC.setLayout(null);
panel_registerAC.setBounds(425,155,370,320);
panel_registerAC.setLayout(null);
//Non-Academic
panel_NonAcademic.setBounds(32,115,772,372);
panel_NonAcademic.setLayout(null);
panel_addNAC.setBounds(42,155,370,320);
panel_addNAC.setLayout(null);
panel_registerNAC.setBounds(425,155,370,320);
panel_registerNAC.setLayout(null);
```

```
/* Colors used for Background and Border.
   1.Background */
Color color_bg =  new Color(188, 235, 253);
panel.setBackground(color_bg);
//Academic
panel_Academic.setBackground(color_bg);
panel_addAC.setBackground(Color.WHITE);
panel_registerAC.setBackground(Color.WHITE);
//Non Academic
panel_NonAcademic.setBackground(color_bg);
panel_addNAC.setBackground(Color.WHITE);
panel_registerNAC.setBackground(Color.WHITE);

// 2. Border
Color color_line = new Color(119, 202, 236);
panel_Academic.setBorder(BorderFactory.createLineBorder(color_line));
panel_NonAcademic.setBorder(BorderFactory.createLineBorder(color_line));

//Main Heading
label_CourseForm = new JLabel("Course Registration Form");
label_CourseForm.setBounds(300,20,250,30);
Font font_titles = new Font("Maiandra GD", Font.BOLD, 20);
label_CourseForm.setFont(font_titles);

/* Color used for:
   Text and Button */
Color color_Courseform = new Color(8, 170, 236);
Color color_Button = new Color(27, 187, 252);
label_CourseForm.setForeground(color_Courseform);
panel.add(label_CourseForm);
```

```
//Font for buttons
Font font_button = new Font("Sitka Heading", Font.BOLD, 15);
Font font_buttonInside = new Font("Sitka Heading", Font.BOLD, 16);

//JButton for Academic Course
button_clickAC = new JButton("Click here for Academic Course.");
button_clickAC.setBounds(32,70,300,27);
button_clickAC.setFont(font_button);
button_clickAC.setBackground(Color.WHITE);
button_clickAC.setForeground(color_Button);
button_clickAC.addActionListener(this);
panel.add(button_clickAC);

//JButton for Non-Academic Course
button_clickNAC = new JButton("Click here for Non-Academic Course.");
button_clickNAC.setBounds(504,70,300,27);
button_clickNAC.setFont(font_button);
button_clickNAC.setBackground(Color.WHITE);
button_clickNAC.setForeground(color_Button);
button_clickNAC.addActionListener(this);
panel.add(button_clickNAC);

/* Start of Academic Course.
   JLabel Academic Course*/
label_AcademicCourse = new JLabel("Academic Course");
label_AcademicCourse.setBounds(307,5,250,30);
label_AcademicCourse.setFont(font_titles);
panel_Academic.add(label_AcademicCourse);
```

```
//Border for Add Academic Course and Register Academic Course.
panel_addAC.setBorder(BorderFactory.createLineBorder(Color.BLACK));
panel_registerAC.setBorder(BorderFactory.createLineBorder(Color.BLACK));

//Font for JLabel Form
Font font_form = new Font("Maiandra GD", Font.BOLD, 15);
Font font_formextra = new Font("Maiandra GD", Font.BOLD, 13);

/* Add Academic Course:
   JLabel and JTextField for Add Academic Course ID */
label_CourseID = new JLabel("Course ID: ");
label_CourseID.setBounds(15,10,135,20);
label_CourseID.setFont(font_form);
panel_addAC.add(label_CourseID);

field_ID = new JTextField();
field_ID.setBounds(160, 10, 195, 27);
field_ID.setBorder(BorderFactory.createLineBorder(color_line));
panel_addAC.add(field_ID);

//JLabel and JTextField for Academic Course Name
label_CourseName = new JLabel("Course Name: ");
label_CourseName.setBounds(15,52,135,20);
label_CourseName.setFont(font_form);
panel_addAC.add(label_CourseName);

field_CName = new JTextField();
field_CName.setBounds(160, 50, 195, 27);
field_CName.setBorder(BorderFactory.createLineBorder(color_line));
panel_addAC.add(field_CName);
```

```
//JLabel and JTextField for Academic Course Duration
label_Duration = new JLabel("Duration: ");
label_Duration.setBounds(15,92,135,20);
label_Duration.setFont(font_form);
panel_addAC.add(label_Duration);

field_Duration = new JTextField();
field_Duration.setBounds(160, 90, 195, 27);
field_Duration.setBorder(BorderFactory.createLineBorder(color_line));
panel_addAC.add(field_Duration);

//JLabel and JTextField for Level
label_Level = new JLabel("Level: ");
label_Level.setBounds(15,132,135,20);
label_Level.setFont(font_form);
panel_addAC.add(label_Level);

field_Level = new JTextField();
field_Level.setBounds(160, 130, 195, 27);
field_Level.setBorder(BorderFactory.createLineBorder(color_line));
panel_addAC.add(field_Level);

//JLabel and JTextField for Credit
label_Credit = new JLabel("Credit: ");
label_Credit.setBounds(15,172,135,20);
label_Credit.setFont(font_form);
panel_addAC.add(label_Credit);

field_Credit = new JTextField();
```

```
        field_Credit.setBounds(160, 170, 195, 27);
        field_Credit.setBorder(BorderFactory.createLineBorder(color_line));
        panel_addAC.add(field_Credit);

        //JLabel and JTextField for Number Of Assessment
        label_NoOfAssessment = new JLabel("No. Of Assessment: ");
        label_NoOfAssessment.setBounds(15,212,135,20);
        label_NoOfAssessment.setFont(font_formextra);
        panel_addAC.add(label_NoOfAssessment);

        field_NoOfAssessment = new JTextField();
        field_NoOfAssessment.setBounds(160, 210, 195, 27);
        field_NoOfAssessment.setBorder(BorderFactory.createLineBorder(color_line));
        panel_addAC.add(field_NoOfAssessment);

        //JButton to Add Academic Course
        button_addAC = new JButton("Add Academic Course");
        button_addAC.setBounds(57,260,260,35);
        button_addAC.setFont(font_buttonInside);
        button_addAC.setBackground(Color.WHITE);
        button_addAC.setForeground(color_line);
        button_addAC.addActionListener(this);
        panel_addAC.add(button_addAC);

        /* Register Academic Course:
           JLabel and JTextField to Register Academic Course ID */
        label_ACregisterID = new JLabel("Course ID: ");
        label_ACregisterID.setBounds(15,10,135,20);
        label_ACregisterID.setFont(font_form);
        label_ACregisterID.setForeground(color_line);
```

```
panel_registerAC.add(label_ACregisterID);

field_ACregisterID = new JTextField();
field_ACregisterID.setBounds(160, 10, 195, 27);
panel_registerAC.add(field_ACregisterID);

//JLabel and JTextField for Academic Course Leader
label_CourseLeader = new JLabel("Course Leader: ");
label_CourseLeader.setBounds(15,52,135,20);
label_CourseLeader.setFont(font_form);
label_CourseLeader.setForeground(color_line);
panel_registerAC.add(label_CourseLeader);

field_ACLeader = new JTextField();
field_ACLeader.setBounds(160, 50, 195, 27);
panel_registerAC.add(field_ACLeader);

//JLabel and JTextField for Academic Course Lecturer Name
label_LecturerName = new JLabel("Lecturer Name: ");
label_LecturerName.setBounds(15,92,135,20);
label_LecturerName.setFont(font_form);
label_LecturerName.setForeground(color_line);
panel_registerAC.add(label_LecturerName);

field_LName = new JTextField();
field_LName.setBounds(160, 90, 195, 27);
panel_registerAC.add(field_LName);

/*JLabel and JComboBox for Academic Course
  1. Starting Date*/
```

```
label_ACStartDate = new JLabel("Starting Date: ");
label_ACStartDate.setBounds(15,132,135,20);
label_ACStartDate.setFont(font_form);
label_ACStartDate.setForeground(color_line);
panel_registerAC.add(label_ACStartDate);

//Font for JComboBox
Font font_box = new Font("Maiandra GD", Font.BOLD, 12);

//For Year to start from 2020 - 2030.
Integer yeardate[] = new Integer[11];
int year = 2020;
for(int i = 0; i <=10; i++)
{
    yeardate[i]=year;
    year++;
}

//JComboBox Year
box_ACYear1 = new JComboBox(yeardate);
box_ACYear1.setBounds(160, 130, 60, 27);
box_ACYear1.setFont(font_box);
box_ACYear1.setBackground(Color.WHITE);
panel_registerAC.add(box_ACYear1);

//Months in a Year.
String months[] = {"January", "February", "March", "April", "May", "June",
                   "July", "August", "September", "October", "November", "December"};

//JComboBox Month
```

```
box_ACMonth1 = new JComboBox(months);
box_ACMonth1.setBounds(221, 130, 88, 27);
box_ACMonth1.setFont(font_box);
box_ACMonth1.setBackground(Color.WHITE);
panel_registerAC.add(box_ACMonth1);

//Days in a month.
Integer day[] = new Integer[31];
for(int i = 0; i < 31; i++)
{
    day[i] = i+1;
}

//JComboBox Date
box_ACDay1= new JComboBox(day);
box_ACDay1.setBounds(310, 130, 44, 27);
box_ACDay1.setFont(font_box);
box_ACDay1.setBackground(Color.WHITE);
panel_registerAC.add(box_ACDay1);

//2. Completion Date
label_ACEndDate = new JLabel("Completion Date: ");
label_ACEndDate.setBounds(15,172,135,20);
label_ACEndDate.setFont(font_form);
label_ACEndDate.setForeground(color_line);
panel_registerAC.add(label_ACEndDate);

//JComboBox Year Month & Date.
box_ACYear2 = new JComboBox(yeardate);
box_ACYear2.setBounds(160, 170, 60, 27);
```

```
box_ACYear2.setFont(font_box);
box_ACYear2.setBackground(Color.WHITE);
panel_registerAC.add(box_ACYear2);

box_ACMonth2 = new JComboBox(months);
box_ACMonth2.setBounds(221, 170, 88, 27);
box_ACMonth2.setFont(font_box);
box_ACMonth2.setBackground(Color.WHITE);
panel_registerAC.add(box_ACMonth2);

box_ACDay2 = new JComboBox(day);
box_ACDay2.setBounds(310, 170, 44, 27);
box_ACDay2.setFont(font_box);
box_ACDay2.setBackground(Color.WHITE);
panel_registerAC.add(box_ACDay2);

//JButton to Register Academic Course
button_registerAC = new JButton("Register Academic Course");
button_registerAC.setBounds(57,260,260,35);
button_registerAC.setFont(font_buttonInside);
button_registerAC.setBackground(Color.WHITE);
button_registerAC.setBorder(BorderFactory.createLineBorder(color_line));
button_registerAC.addActionListener(this);
panel_registerAC.add(button_registerAC);

//JButton to Display Academic Course
button_Display = new JButton("Display");
button_Display.setBounds(595,505,100,30);
button_Display.setFont(font_button);
button_Display.setBackground(Color.WHITE);
```

```
button_Display.setBorder(BorderFactory.createLineBorder(color_line));
button_Display.addActionListener(this);
panel.add(button_Display);

/* Start of Non-Academic Course.
   JLabel Non-Academic Course*/
label_NonAcademicCourse = new JLabel("Non-Academic Course");
label_NonAcademicCourse.setBounds(280,5,250,30);
label_NonAcademicCourse.setFont(font_titles);
panel_NonAcademic.add(label_NonAcademicCourse);

//Border for Add Academic Course and Register Academic Course.
panel_addNAC.setBorder(BorderFactory.createLineBorder(Color.BLACK));
panel_registerNAC.setBorder(BorderFactory.createLineBorder(Color.BLACK));

/* Add Non-Academic Course:
   JLabel and JTextField for Add Non-Academic Course ID */
label_NACourseID = new JLabel("Course ID: ");
label_NACourseID.setBounds(15,10,135,20);
label_NACourseID.setFont(font_form);
panel_addNAC.add(label_NACourseID);

fieldNA_ID = new JTextField();
fieldNA_ID.setBounds(160, 10, 195, 27);
fieldNA_ID.setBorder(BorderFactory.createLineBorder(color_line));
panel_addNAC.add(fieldNA_ID);

//JLabel and JTextField for Non Academic Course Name
label_NACourseName = new JLabel("Course Name: ");
label_NACourseName.setBounds(15,52,135,20);
```

```
label_NACourseName.setFont(font_form);
panel_addNAC.add(label_NACourseName);

fieldNA_CName = new JTextField();
fieldNA_CName.setBounds(160, 50, 195, 27);
fieldNA_CName.setBorder(BorderFactory.createLineBorder(color_line));
panel_addNAC.add(fieldNA_CName);

//JLabel and JTextField for Non Academic Course Duration
label_NADuration = new JLabel("Duration: ");
label_NADuration.setBounds(15,92,135,20);
label_NADuration.setFont(font_form);
panel_addNAC.add(label_NADuration);

field_NADuration = new JTextField();
field_NADuration.setBounds(160, 90, 195, 27);
field_NADuration.setBorder(BorderFactory.createLineBorder(color_line));
panel_addNAC.add(field_NADuration);

//JLabel and JTextField for Prerequisite
label_Prerequisite = new JLabel("Prerequisite: ");
label_Prerequisite.setBounds(15,132,135,20);
label_Prerequisite.setFont(font_form);
panel_addNAC.add(label_Prerequisite);

field_Prerequisite = new JTextField();
field_Prerequisite.setBounds(160, 130, 195, 27);
field_Prerequisite.setBorder(BorderFactory.createLineBorder(color_line));
panel_addNAC.add(field_Prerequisite);
```

```
//JButton to Add Non-Academic Course
button_addNAC = new JButton("Add Non-Academic Course");
button_addNAC.setBounds(57,195,260,35);
button_addNAC.setFont(font_buttonInside);
button_addNAC.setBackground(Color.WHITE);
button_addNAC.setForeground(color_line);
button_addNAC.addActionListener(this);
panel_addNAC.add(button_addNAC);

//JButton to Remove Non-Academic Course
button_RemoveNAC = new JButton("Remove Course");
button_RemoveNAC.setBounds(57,260,260,35);
button_RemoveNAC.setFont(font_buttonInside);
button_RemoveNAC.setBackground(Color.WHITE);
button_RemoveNAC.setForeground(color_line);
button_RemoveNAC.addActionListener(this);
panel_addNAC.add(button_RemoveNAC);

/* Register Non Academic Course:
   JLabel and JTextField to Register Non-Academic Course ID */
label_NACregisterID = new JLabel("Course ID: ");
label_NACregisterID.setBounds(15,10,135,20);
label_NACregisterID.setFont(font_form);
label_NACregisterID.setForeground(color_line);
panel_registerNAC.add(label_NACregisterID);

field_NACregisterID = new JTextField();
field_NACregisterID.setBounds(160, 10, 195, 27);
panel_registerNAC.add(field_NACregisterID);
```

```
//JLabel and JTextField for Non Academic Course Leader
label_NACourseLeader = new JLabel("Course Leader: ");
label_NACourseLeader.setBounds(15,52,135,20);
label_NACourseLeader.setFont(font_form);
label_NACourseLeader.setForeground(color_line);
panel_registerNAC.add(label_NACourseLeader);

field_NACLeader = new JTextField();
field_NACLeader.setBounds(160, 50, 195, 27);
panel_registerNAC.add(field_NACLeader);

//JLabel and JTextField for Non Academic Course Instructor Name
label_InstructorName = new JLabel("Instructor Name: ");
label_InstructorName.setBounds(15,92,135,20);
label_InstructorName.setFont(font_form);
label_InstructorName.setForeground(color_line);
panel_registerNAC.add(label_InstructorName);

field_IName = new JTextField();
field_IName.setBounds(160, 90, 195, 27);
panel_registerNAC.add(field_IName);

/*JLabel and JComboBox for Non Academic Course
  1. Starting Date*/
label_NACStartDate = new JLabel("Starting Date: ");
label_NACStartDate.setBounds(15,132,135,20);
label_NACStartDate.setFont(font_form);
label_NACStartDate.setForeground(color_line);
panel_registerNAC.add(label_NACStartDate);
```

```
//JComboBox Year
box_NACYear1 = new JComboBox(yeardate);
box_NACYear1.setBounds(160, 130, 60, 27);
box_NACYear1.setFont(font_box);
box_NACYear1.setBackground(Color.WHITE);
panel_registerNAC.add(box_NACYear1);

//JComboBox Month
box_NACMonth1 = new JComboBox(months);
box_NACMonth1.setBounds(221, 130, 88, 27);
box_NACMonth1.setFont(font_box);
box_NACMonth1.setBackground(Color.WHITE);
panel_registerNAC.add(box_NACMonth1);

//JComboBox Date
box_NACDay1 = new JComboBox(day);
box_NACDay1.setBounds(310, 130, 44, 27);
box_NACDay1.setFont(font_box);
box_NACDay1.setBackground(Color.WHITE);
panel_registerNAC.add(box_NACDay1);

//2. Completion Date
label_NACEndDate = new JLabel("Completion Date: ");
label_NACEndDate.setBounds(15,172,135,20);
label_NACEndDate.setFont(font_form);
label_NACEndDate.setForeground(color_line);
panel_registerNAC.add(label_NACEndDate);

//JComboBox Year Month & Date
box_NACYear2 = new JComboBox(yeardate);
```

```
box_NACYear2.setBounds(160, 170, 60, 27);
box_NACYear2.setFont(font_box);
box_NACYear2.setBackground(Color.WHITE);
panel_registerNAC.add(box_NACYear2);

box_NACMonth2 = new JComboBox(months);
box_NACMonth2.setBounds(221, 170, 88, 27);
box_NACMonth2.setFont(font_box);
box_NACMonth2.setBackground(Color.WHITE);
panel_registerNAC.add(box_NACMonth2);

box_NACDay2 = new JComboBox(day);
box_NACDay2.setBounds(310, 170, 44, 27);
box_NACDay2.setFont(font_box);
box_NACDay2.setBackground(Color.WHITE);
panel_registerNAC.add(box_NACDay2);

//3. Exam Date
label_ExamDate = new JLabel("Exam Date: ");
label_ExamDate.setBounds(15,208,135,20);
label_ExamDate.setFont(font_form);
label_ExamDate.setForeground(color_line);
panel_registerNAC.add(label_ExamDate);

//JComboBox Year Month & Date
box_NACYear3 = new JComboBox(yeardate);
box_NACYear3.setBounds(160, 210, 60, 27);
box_NACYear3.setFont(font_box);
box_NACYear3.setBackground(Color.WHITE);
panel_registerNAC.add(box_NACYear3);
```

```
box_NACMonth3 = new JComboBox(months);
box_NACMonth3.setBounds(221, 210, 88, 27);
box_NACMonth3.setFont(font_box);
box_NACMonth3.setBackground(Color.WHITE);
panel_registerNAC.add(box_NACMonth3);

box_NACDay3 = new JComboBox(day);
box_NACDay3.setBounds(310, 210, 44, 27);
box_NACDay3.setFont(font_box);
box_NACDay3.setBackground(Color.WHITE);
panel_registerNAC.add(box_NACDay3);

//JButton to Register Non Academic Course
button_registerNAC = new JButton("Register Non-Academic Course");
button_registerNAC.setBounds(57,260,260,35);
button_registerNAC.setFont(font_buttonInside);
button_registerNAC.setBackground(Color.WHITE);
button_registerNAC.setBorder(BorderFactory.createLineBorder(color_line));
button_registerNAC.addActionListener(this);
panel_registerNAC.add(button_registerNAC);

//JButton to Display Non-Academic Course
button_NADisplay = new JButton("Display");
button_NADisplay.setBounds(595,505,100,30);
button_NADisplay.setFont(font_button);
button_NADisplay.setBackground(Color.WHITE);
button_NADisplay.setBorder(BorderFactory.createLineBorder(color_line));
button_NADisplay.addActionListener(this);
panel.add(button_NADisplay);
```

```
//JButton to Clear
button_Clear = new JButton("Clear");
button_Clear.setBounds(704,505,100,30);
button_Clear.setFont(font_button);
button_Clear.setBackground(Color.WHITE);
button_Clear.setBorder(BorderFactory.createLineBorder(color_line));
button_Clear.addActionListener(this);
panel.add(button_Clear);

//Setting Non-Academic Course's Course Visibility to false
button_NADisplay.setVisible(false);
panel_addNAC.setVisible(false);
panel_registerNAC.setVisible(false);
panel_NonAcademic.setVisible(false);

//Adding panel to frame
frame.add(panel_addAC);
frame.add(panel_registerAC);
frame.add(panel_Academic);
frame.add(panel_addNAC);
frame.add(panel_registerNAC);
frame.add(panel_NonAcademic);
frame.add(panel);

//Bounds, Title, Resizability and Visibility of the Frame.
frame.setBounds(345,132,850,595);
frame.setTitle("Course Registration Form");
frame.setResizable(false);
frame.setVisible(true);
```

```
    }

    //Main Method to call GUI.
    public static void main(String[] args)
    {
        main.gUI();
    }

    /* Using Accessor Method / Getters Method for Academic Course.
       1. Add Academic Course*/

    public String getACourseID()
    {
        return this.field_ID.getText();
    }

    public String getACourseName()
    {
        return this.field_CName.getText();
    }

    public int getACDuration()
    {
        return Integer.parseInt(this.field_Duration.getText());
    }

    public String getLevel()
    {
        return this.field_Level.getText();
    }
```

```
    public String getCredit()
    {
        return this.field_Credit.getText();
    }

    public int getNoOfAssessment()
    {
        return Integer.parseInt(this.field_NoOfAssessment.getText());
    }

    //2. Register Academic Course

    public String getACourseRegisterID()
    {
        return this.field_ACregisterID.getText();
    }

    public String getACourseLeader()
    {
        return this.field_ACLeader.getText();
    }

    public String getACLecturerName()
    {
        return this.field_LName.getText();
    }

    public String getACStartDate()
    {
```

```
        ACStartingDate = box_ACYear1.getSelectedItem()+ " " +
box_ACMonth1.getSelectedItem()+ " " + box_ACDay1.getSelectedItem();
        return ACStartingDate;
    }

    public String getACCompleteDate()
    {
        ACCompletionDate = box_ACYear2.getSelectedItem()+ " " +
box_ACMonth2.getSelectedItem()+ " " + box_ACDay2.getSelectedItem();
        return ACCompletionDate;
    }

    /* Using Accessor Method / Getters Method for Non Academic Course.
       1. Add Non Academic Course*/

    public String getNACourseID()
    {
        return this.fieldNA_ID.getText();
    }

    public String getNACourseName()
    {
        return this.fieldNA_CName.getText();
    }

    public int getNACDuration()
    {
        return Integer.parseInt(this.field_NADuration.getText());
    }
```

```
    public String getPrerequisite()
    {
        return this.field_Prerequisite.getText();
    }

    //2. Register Non Academic Course

    public String getNACourseRegisterID()
    {
        return this.field_NACregisterID.getText();
    }

    public String getNACourseLeader()
    {
        return this.field_NACLeader.getText();
    }

    public String getInstructorName()
    {
        return this.field_IName.getText();
    }

    public String getNACStartDate()
    {
        NACStartingDate = box_NACYear1.getSelectedItem()+ " " +
box_NACMonth1.getSelectedItem()+ " " + box_NACDay1.getSelectedItem();
        return NACStartingDate;
    }

    public String getNACCompleteDate()
```

```
    {
        NACCompletionDate = box_NACYear2.getSelectedItem().toString()+ " " +
box_NACMonth2.getSelectedItem()+ " " + box_NACDay2.getSelectedItem();
        return NACCompletionDate;
    }

    public String getExamDate()
    {
        ExamDate = box_NACYear3.getSelectedItem()+ " " +
box_NACMonth3.getSelectedItem()+ " " + box_NACDay3.getSelectedItem();
        return ExamDate;
    }

    //Implementing actionPerformed method.
    public void actionPerformed(ActionEvent e)
    {
        /* Changes to Academic Course panel when the button is clicked.
           1. Set Visible as true for Academic Components.
           2. Set Visible as false for Non Academic Components. */
        if(e.getSource() == button_clickAC)
        {
            button_Display.setVisible(true);
            panel_addAC.setVisible(true);
            panel_registerAC.setVisible(true);
            panel_Academic.setVisible(true);

            button_NADisplay.setVisible(false);
            panel_addNAC.setVisible(false);
            panel_registerNAC.setVisible(false);
            panel_NonAcademic.setVisible(false);
```

```
        }

        /* Changes to Non-Academic Course panel when the button is clicked.
           1. Set Visibile as true for Non Academic Components.
           2. Set Visible as false for Academic Components. */
        else if(e.getSource() == button_clickNAC)
        {
            button_NADisplay.setVisible(true);
            panel_addNAC.setVisible(true);
            panel_registerNAC.setVisible(true);
            panel_NonAcademic.setVisible(true);

            button_Display.setVisible(false);
            panel_addAC.setVisible(false);
            panel_registerAC.setVisible(false);
            panel_Academic.setVisible(false);
        }

        /* ACADEMIC COURSE BUTTONS.
           Adds Academic Course when the button is clicked.
           1. Makes sure that none of the fields are left empty.
           2. If the Course ID has already been added an error message pops up.
           3. Checks if the user has entered correct integer value in Duration and
Assessment field.
           4. Adds the Academic Course after checking. */
        else if(e.getSource() == button_addAC)
        {
            int count = 0;
            int count2 = 0;
```

```
        if (getACourseID().equals("") || getACourseName().equals("") ||
getLevel().equals("") || getCredit().equals(""))
        {
            JOptionPane.showMessageDialog(frame, "Please ensure that all fields are
filled.","ATTENTION!", JOptionPane.ERROR_MESSAGE);
            count = 1;
        }
        if (count == 0)
        {
            for (Course alist: list_Course)
            {
                if (getACourseID().equals(alist.getCourseID()))
                {
                    JOptionPane.showMessageDialog(frame, "The Course ID: " +
getACourseID() +"\n"+ " with Course Name: " + getACourseName() +
                                        "\n"+" has already been added.","ATTENTION!",
JOptionPane.ERROR_MESSAGE);
                    count2 = 1;
                    break;
                }
            }
            try
            {
                Integer.parseInt(this.field_Duration.getText());
                try
                {
                    Integer.parseInt(this.field_NoOfAssessment.getText());
                }
                catch (NumberFormatException ex)
                {
```

```
                    JOptionPane.showMessageDialog(frame, "The Assessment must be an
integer number.","ATTENTION!", JOptionPane.ERROR_MESSAGE);
                    count2 = 1;
                }
            }
            catch (NumberFormatException ex)
            {
                JOptionPane.showMessageDialog(frame, "The Duration must be an integer
number.","ATTENTION!", JOptionPane.ERROR_MESSAGE);
                count2 = 1;
            }
            if (count2 == 0)
            {
                JOptionPane.showMessageDialog(frame, "The following values has been
added \n"+ "Course ID: " + getACourseID() + "\n" +
                                        "Course Name: " + getACourseName() + "\n"+
"Duration: " + getACDuration() + "\n" +
                                        "Level: " + getLevel() + "\n"+ "Credit: " + getCredit() +
"\n"+"No. Of Assessment: " +
                                         getNoOfAssessment()+"\n" + "to Academic Course.");
                AcademicObject = new AcademicCourse(getACourseID(),
getACourseName(), getACDuration(), getLevel(), getCredit(), getNoOfAssessment());
                list_Course.add(AcademicObject);
            }
        }
    }

    /* Register Academic Course when the Button is clicked.
       1. Makes sure that none of the fields are left empty.
       2. Checks if the Course is already registered.
```

```
        3. Registers the Academic Course after checking.
        4. If the Course ID doesn't match an error message pops up. */
    else if(e.getSource() == button_registerAC)
    {
        int count = 0;
        if (getACourseRegisterID().equals("") || getACourseLeader().equals("") ||
getACLecturerName().equals(""))
        {
            JOptionPane.showMessageDialog(frame, "Please ensure that all fields are
filled.","ATTENTION!", JOptionPane.ERROR_MESSAGE);
            count = 1;
        }
        if (count == 0)
        {
            for (Course alist: list_Course)
            {
                if (getACourseRegisterID().equals(alist.getCourseID()) && alist instanceof
AcademicCourse)
                {
                    AcademicCourse AcademicObject = (AcademicCourse) alist;
                    count = 1;
                    if (AcademicObject.getisRegistered() == true)
                    {
                        JOptionPane.showMessageDialog(frame, "This course is already
registered.");
                        count = 1;
                    }
                    else if (AcademicObject.getisRegistered() == false)
                    {
```

```
                    AcademicObject.register(getACourseLeader(), getACLecturerName(),
getACStartDate() ,getACCompleteDate());
                    JOptionPane.showMessageDialog(frame, "The Academic Course has
been registered. \n"
                                        + "Course ID: " + getACourseRegisterID() + "\n"
                                        + "Course Leader: " + getACourseLeader() + "\n"
                                        + "Lecturer Name: " + getACLecturerName() + "\n"
                                        + "Starting Date: " + getACStartDate() + "\n"
                                        + "Completion Date: " + getACCompleteDate());
                    count = 1;
                }
            }
        }
    }
    if (count == 0)
    {
        JOptionPane.showMessageDialog(frame, "Course ID does not match. Please
enter the correct value.", "ATTENTION!", JOptionPane.WARNING_MESSAGE);
    }
}

//Display Academic Course details when the Button is clicked.
else if(e.getSource() == button_Display)
{
    for (Course alist: list_Course)
    {
        if (alist instanceof AcademicCourse)
        {
            AcademicCourse AcademicObject = (AcademicCourse) alist;
            System.out.println("");
```

```
                AcademicObject.display();
            }
        }
    }

    /* NON-ACADEMIC COURSE BUTTONS.
       Add Non Academic Course when the button is clicked.
       1. Makes sure that none of the fields are left empty.
       2. If the Course ID has already been added an error message pops up.
       3. Checks if the user has entered correct integer value in Duration field.
       4. Adds the Non Academic Course after checking.*/
    else if(e.getSource() == button_addNAC)
    {
        int count = 0;
        int count2 = 0;
        if (getNACourseID().equals("") || getNACourseName().equals("") ||
getPrerequisite().equals(""))
        {
            JOptionPane.showMessageDialog(frame, "Please ensure that all fields are
filled.","ATTENTION!", JOptionPane.ERROR_MESSAGE);
            count = 1;
        }
        if (count == 0)
        {
            for (Course alist: list_Course)
            {
                if (getNACourseID().equals(alist.getCourseID()))
                {
                    JOptionPane.showMessageDialog(frame, "The Course ID: " +
getNACourseID() + "\n"+ " with Course Name: " + getNACourseName() +
```

```
                                        "\n"+" has already been added.","ATTENTION!",
JOptionPane.ERROR_MESSAGE);
                    count2 = 1;
                    break;
                }
            }
            try
            {
                Integer.parseInt(this.field_NADuration.getText());
            }
            catch (NumberFormatException ex)
            {
                JOptionPane.showMessageDialog(frame, "The Duration must be an integer
number.","ATTENTION!", JOptionPane.ERROR_MESSAGE);
                count2 = 1;
            }
            if (count2 == 0)
            {
                JOptionPane.showMessageDialog(frame, "The following values has been
added \n"+ "Course ID: " + getNACourseID() + "\n"+
                                        "Course Name: " + getNACourseName() + "\n"+
"Duration: " + getNACDuration() + "\n" +
                                        "Prerequisite: " + getPrerequisite() + "\n" + "to Non-
Academic Course.");
                NonAcademicObject = new NonAcademicCourse(getNACourseID(),
getNACourseName(), getNACDuration(), getPrerequisite());
                list_Course.add(NonAcademicObject);
            }
        }
    }
```

```
        /* Register Non-Academic Course when the Button is clicked.
           1. Makes sure that none of the fields are left empty.
           2. Checks if the Course is already registered.
           3. Registers the Non Academic Course after checking.
           4. If the Course ID doesn't match an error message pops up. */
        else if(e.getSource() == button_registerNAC)
        {
            int count = 0;
            if (getNACourseRegisterID().equals("") || getNACourseLeader().equals("") ||
getInstructorName().equals(""))
            {
                JOptionPane.showMessageDialog(frame, "Please ensure that all fields are
filled.","ATTENTION!", JOptionPane.ERROR_MESSAGE);
                count = 1;
            }
            if (count == 0)
            {
                for (Course alist: list_Course)
                {
                    if (getNACourseRegisterID().equals(alist.getCourseID()) && alist instanceof
NonAcademicCourse)
                    {
                        NonAcademicCourse NonAcademicObject = (NonAcademicCourse)
alist;
                        count = 1;
                        if (NonAcademicObject.getisRegistered() == true)
                        {
                            JOptionPane.showMessageDialog(frame, "This course is already
registered.");
```

```
                    count = 1;
                }
                else if (NonAcademicObject.getisRegistered() == false)
                {
                    NonAcademicObject.register(getNACourseLeader(),
getInstructorName(), getNACStartDate() ,getNACCompleteDate(),
                                        getExamDate());
                    JOptionPane.showMessageDialog(frame, "The Non-Academic Course
has been registered. \n"
                                        + "Course ID: " + getNACourseRegisterID() + "\n"
                                        + "Course Leader: " + getNACourseLeader() + "\n"
                                        + "InstructorName: " + getInstructorName() + "\n"
                                        + "Starting Date: " + getNACStartDate() + "\n"
                                        + "Completion Date: " + getNACCompleteDate() +
"\n"
                                        + "Exam Date: " + getExamDate());
                    count = 1;
                }
            }
        }
    }
    if (count == 0)
    {
        JOptionPane.showMessageDialog(frame, "Course ID does not match. Please
enter the correct value.", "ATTENTION!", JOptionPane.WARNING_MESSAGE);
    }
}

/* Remove Non-Academic Course when the Button is clicked.
  1. Checks the Course ID.
```

```
        2. If isRemoved is false, it removes the course.
        3. If it has already been removed an error message pops up.*/
    else if(e.getSource() == button_RemoveNAC)
    {
        int count = 0;
        if (getNACourseID().equals("") || getNACourseRegisterID().equals(""))
        {
            JOptionPane.showMessageDialog(frame, "Please ensure that all fields are
filled.","ATTENTION!", JOptionPane.ERROR_MESSAGE);
            count = 1;
        }
        if (count == 0)
        {
            for (Course alist: list_Course)
            {
                if (getNACourseID().equals(alist.getCourseID()) && alist instanceof
NonAcademicCourse)
                {
                    NonAcademicCourse NACourse = (NonAcademicCourse) alist;
                    if(NACourse.getisRemoved()==false)
                    {
                        JOptionPane.showMessageDialog(frame, "The Non-Academic Course
with "+ "\n"
                                                + "Course ID: "+ getNACourseID() + " has been
removed.");
                        NACourse.remove();
                        count = 1;
                    }
                    else if(NACourse.getisRemoved()==true)
                    {
```

```
                    JOptionPane.showMessageDialog(frame, "The Non-Academic Course
with "+ "\n"
                                        + "Course ID: "+ getNACourseID() + " has already
been removed.","Attention"
                                        ,JOptionPane.ERROR_MESSAGE);
                    break;
                }
            }
        }
    }
}

//Display Non-Academic Course details when the Button is clicked.
else if(e.getSource() == button_NADisplay)
{
    for (Course alist: list_Course)
    {
        if (alist instanceof NonAcademicCourse)
        {
            NonAcademicCourse NonAcademicObject = (NonAcademicCourse) alist;
            System.out.println("");
            NonAcademicObject.display();
        }
    }
}

//Clear all fields when the Button is clicked.
else if(e.getSource() == button_Clear)
{
    //Clear all the data present
```

```
field_ID.setText("");
field_CName.setText("");
field_Duration.setText("");
field_Level.setText("");
field_Credit.setText("");
field_NoOfAssessment.setText("");
field_ACregisterID.setText("");
field_ACLeader.setText("");
field_LName.setText("");
fieldNA_ID.setText("");
fieldNA_CName.setText("");
field_NADuration.setText("");
field_Prerequisite.setText("");
field_NACregisterID.setText("");
field_NACLeader.setText("");
field_IName.setText("");
box_ACYear1.setSelectedIndex(0);
box_ACMonth1.setSelectedIndex(0);
box_ACDay1.setSelectedIndex(0);
box_ACYear2.setSelectedIndex(0);
box_ACMonth2.setSelectedIndex(0);
box_ACDay2.setSelectedIndex(0);
box_NACYear1.setSelectedIndex(0);
box_NACMonth1.setSelectedIndex(0);
box_NACDay1.setSelectedIndex(0);
box_NACYear2.setSelectedIndex(0);
box_NACMonth2.setSelectedIndex(0);
box_NACDay2.setSelectedIndex(0);
box_NACYear3.setSelectedIndex(0);
box_NACMonth3.setSelectedIndex(0);
```

```
            box_NACDay3.setSelectedIndex(0);
        }
    }
}
```

9. Appendix2.

1. Course Class.

```
/**
 * A course class represents a real world course.
 * Course is used as an abstract superclass of:
 * Academic Course and Non Academic Course.
 * @author (Rabina Shrestha)
 * @version (5.0.0)
 */
public class Course
{
    //Four Attributes / Instance Variables of Course Class.

    private String CourseID;
    private String CourseName;
    private String CourseLeader;
    private int Duration;

    /*
     Creating Constructor of Course with CourseID, Name, and Duration.
     */

    Course(String CourseID, String CourseName, int Duration)
```

```
    {
        this.CourseID = CourseID;
        this.CourseName = CourseName;
        this.CourseLeader = "";
        this.Duration = Duration;
    }

    // Using Accessor Method / Getters Method for each attributes.

    /*
     Return the Course ID of the Course.
     */

    public String getCourseID()
    {
        return this.CourseID;
    }

    /*
     Return the Course Name of the Course.
     */

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

    /*
     Return the Course Leader of the Course.
     */
```

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

/*
 Return the Duration of the Course.
 */

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

// Parameter passed to method in order to set a new name as the Course Leader.

/*
 Set a new Course Leader for the Course.
 */

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

/*
 Display the Course details and Course Leader if assigned.
 */
```

```
    public void display()
    {
        System.out.println("Course ID: " + getCourseID());
        System.out.println("Course Name: " + getCourseName());
        System.out.println("Duration: " + getDuration());
        if (CourseLeader != "")
        {
            System.out.println("Course Leader: " + getCourseLeader());
        }
    }
}
```

2. Academic Course Class

```
/**
 * A class representing Academic Course.
 * Academic Course is used as a subclass of Course.
 * @author (Rabina Shrestha)
 * @version (5.0.0)
 */
public class AcademicCourse extends Course
{
    // Seven Attributes / Instance Variables of Academic Course.

    private String LecturerName;
    private String Level;
    private String Credit;
    private String StartingDate;
    private String CompletionDate;
```

```
    private int NumberOfAssessments;
    private boolean isRegistered;

    /*
     Creating Constructor of Academic Course which has six parameters.
     */

    AcademicCourse(String CourseID,String CourseName, int Duration,String Level,
String Credit, int NumberOfAssessments)
    {
        super(CourseID, CourseName, Duration);
        this.LecturerName = "";
        this.StartingDate = "";
        this.CompletionDate = "";
        this.Level = Level;
        this.Credit = Credit;
        this.NumberOfAssessments = NumberOfAssessments;
        this.isRegistered = false;
    }

    // Using Accessor Method / Getters Method for each attributes.

    /*
     Return the Lecturer Name of the Academic Course.
     */

    public String getLecturerName()
    {
        return this.LecturerName;
    }
```

```
/*
 Return the Level of the Academic Course.
 */

public String getLevel()
{
    return this.Level;
}

/*
 Return the Credit of the Academic Course.
 */

public String getCredit()
{
    return this.Credit;
}

/*
 Return the Starting Date of the Academic Course.
 */

public String getStartingDate()
{
    return this.StartingDate;
}

/*
 Return the Completion Date of the Academic Course.
```

```
     */

    public String getCompletionDate()
    {
        return CompletionDate;
    }

    /*
     Return the Number of Assessments of the Academic Course.
     */

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

    /*
     Return the Registered Status of the Academic Course.
     */

    public boolean getisRegistered()
    {
        return this.isRegistered;
    }

    // Parameter passed to method in order to change and set new Lecturer Name and
new Number of Assessments.

    /*
     Set a new Lecturer Name for the Academic Course.
```

```
     */

    public void setLecturerName(String newLecturerName)
    {
        this.LecturerName = newLecturerName;
    }

    /*
     Set a new Number of Assessments for the Academic Course.
     */

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



    /* Method used to register any particular academic course.
       Method has four parameters. */

    /*
     Register the Academic Course.
     */

    public void register(String CourseLeader, String LecturerName , String StartingDate,
String CompletionDate)
    {
        if (this.isRegistered == true)
        {
```

```
            System.out.println("This course is already registered.The details of the course:
");
            System.out.println("Lecturer Name: " + this.LecturerName);
            System.out.println("Starting Date: " + this.StartingDate);
            System.out.println("Completion Date: " + this.CompletionDate);
        }
        else
        {
            super.setCourseLeader(CourseLeader);
            /* CourseLeader is called from the parent class with course leader name
               as a parameter. */
            this.LecturerName = LecturerName;
            this.StartingDate = StartingDate;
            this.CompletionDate = CompletionDate;
            this.isRegistered = true;
        }
    }

    /*
     Display the Academic Course details.
     */

    public void display()
    {
        super.display();
        /* Calling method in Course class to display
           CourseID, CourseName, Duration, and CourseLeader. */
        System.out.println("Level: " + getLevel());
        System.out.println("Credit: " + getCredit());
        System.out.println("Number of Assessments: " + getNumberOfAssessments());
```

```
        if (this.isRegistered == true)
        {
            System.out.println("Lecturer Name: " + getLecturerName());
            System.out.println("Starting Date: " + getStartingDate());
            System.out.println("Completion Date: " + getCompletionDate());
        }
    }
}
```

3. Non-Academic Course Class.

```
/**
 * A class representing Non Academic Course.
 * Non Academic Course is used as a subclass of Course.
 * @author (Rabina Shrestha)
 * @version (5.0.0)
 */
public class NonAcademicCourse extends Course
{
    // Seven Attributes / Instance Variables of Non Academic Course.

    private String InstructorName;
    private int Duration;
    private String StartingDate;
    private String CompletionDate;
    private String ExamDate;
    private String Prerequisite;
    private boolean isRegistered;
    private boolean isRemoved;
```

```
    /*
     Creating Constructor of Non Academic Course which has seven parameters.
     */

    NonAcademicCourse(String CourseID, String CourseName, int Duration, String
Prerequisite)
    {
        super(CourseID, CourseName, Duration);
        this.InstructorName = "";
        this.Prerequisite = Prerequisite;
        this.Duration = Duration;
        this.StartingDate = "";
        this.CompletionDate = "";
        this.ExamDate = "";
        this.isRegistered = false;
        this.isRemoved = false;
    }

    // Using Accessor Method / Getters Method for each attributes.

    /*
     Return the Instructor Name of the Non Academic Course.
     */

    public String getInstructorName()
    {
        return this.InstructorName;
    }
```

```
/*
 Return the Duration of the Non Academic Course.
 */

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

/*
 Return the Starting Date of the Non Academic Course.
 */

public String getStartingDate()
{
    return this.StartingDate;
}

/*
 Return the Completion Date of the Non Academic Course.
 */

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

/*
 Return the Exam Date of the Non Academic Course.
 */
```

```
    public String getExamDate()
    {
        return this.ExamDate;
    }

    /*
     Return the Prerequisite of the Non Academic Course.
     */

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

    /*
     Return the Registered status of the Non Academic Course.
     */

    public boolean getisRegistered()
    {
        return this.isRegistered;
    }

    /*
     Return the Removed status of the Non Academic Course.
     */

    public boolean getisRemoved()
    {
```

```
        return this.isRemoved;
    }

    // Parameter passed to method in order to change and set new Instructor Name.

    /*
     Set a new Instructor Name for the Non Academic Course.
     */

    public void setInstructorName(String newInstructorName)
    {
        if (this.isRegistered == false)
        {
            this.InstructorName = newInstructorName;
        }
        else
        {
            System.out.println("The Instructor Name is already registered, cannot update
instructor name");
        }
    }

    /* Method used to register the non academic course.
       Method has five parameters. */

    /*
     Register the Non Academic Course.
     */
```

```
    public void register(String CourseLeader, String InstructorName, String StartingDate,
String CompletionDate, String ExamDate)
    {
        if (this.isRegistered == true)
        {
            System.out.println("This course is already registered.The details of the course:
");
            System.out.println("Instructor Name: " + this.InstructorName);
            System.out.println("Starting Date: " + this.StartingDate);
            System.out.println("Completion Date: " + this.CompletionDate);
            System.out.println("Exam Date: " + this.ExamDate);
        }
        else
        {
            super.setCourseLeader(CourseLeader);
            /* CourseLeader is called from the parent class with course leader name
               as a parameter. */
            this.InstructorName = InstructorName;
            this.StartingDate = StartingDate;
            this.CompletionDate = CompletionDate;
            this.ExamDate = ExamDate;
            this.isRegistered = true;
        }
    }

    // Method used to remove the non academic course.

    /*
     Remove the Non Academic Course.
     */
```

```
    public void remove()
    {
        if (this.isRemoved == true)
        {
            System.out.println("The course has already been removed.");
        }
        else
        {
            super.setCourseLeader("");
            this.InstructorName = "";
            this.StartingDate = "";
            this.CompletionDate = "";
            this.ExamDate = "";
            this.isRegistered = false;
            this.isRemoved = true;
        }
    }

    /*
     Display the Non Academic Course details.
     */

    public void display()
    {
        super.display();
        /* Calling method in Course class to display
           CourseID, CourseName, and Duration. */
        System.out.println("Prerequisite: " + getPrerequisite());
        if (this.isRegistered == true)
```

```
        {
            System.out.println("Starting Date: " + getStartingDate());
            System.out.println("Completion Date: " + getCompletionDate());
            System.out.println("Exam Date: " + getExamDate());
        }
    }
}
```

10.Bibliography

BlueJ, 1999. *BlueJ.* [Online]
Available at: https://www.bluej.org/about.html
[Accessed 18 August 2021].

Computer Hope, 2020. *Computer Hope.* [Online]
Available at: https://www.computerhope.com/jargon/d/drawio.htm
[Accessed 18 August 2021].

Java, 1996. *Java.* [Online]
Available at: https://www.java.com/en/java_in_action/bluej.jsp
[Accessed 18 August 2021].

Kölling, M. & Institute, M., 1999. *BlueJ.* [Online]
Available at: https://www.bluej.org/tutorial/tutorial-201.pdf
[Accessed 18 August 2021].

Microsft, 1989. *Microsoft.* [Online]
Available at: https://www.microsoft.com/en-us/microsoft-365/word
[Accessed 18 August 2021].

Shrestha, R., 2021. *Programming Coursework 1,* Kathmandu: Programming Assessment 1.

Somani, D., 2021. *Gadgets Now.* [Online]
Available at: https://www.gadgetsnow.com/faqs/what-is-ms-word-and-its-features/articleshow/84464446.cms
[Accessed 18 August 2021].