Module Code & Module Title
CS4001NI Programming

COURSEWORK-2

Assessment Weightage & Type
30% Individual Coursework

Semester and Year

Spring 2021

Student Name: Ira Lamichhane

Group: N2

London Met ID: 20049152

College ID: NP01NT4S210049
Assignment Due Date: 20th August, 2021
Assignment Submission Date: 20th August, 2021

Table of Contents

Table of Figure

Table of Tables

1. Introduction

Java is an island of the Sunda Islands, south of the Java Sea to the Greater Sunda Islands over 152 million, or 56.1% of the Indonesian population. Java's name "Java" originally came from the java coffee from that island. Java is the number 1 platform for programming and development. It reduces cost, reduces time frames for growth, stimulates creativity and enhances appliance services. Java continues to be the programming platform of choice for businesses and developers, as millions of developers run over 51 billion virtual Java machines worldwide. (Oracle, 2021)

This is the 2nd coursework of module "Programming". This coursework is done using Blue J, MS Word, etc. The aim of this coursework is to implement a scenario of real problems with the Object-oriented Java concept. for the first part of the coursework to make a graphical user interface (GUI) for a system that stores details of Course that includes both academic and non-academic course. We had to create a new class called INGCollege. BlueJ is a Java Development Kit platform for Windows (JDK). In 1999, Michael Kolling and John Rosenberg of Monash University, Australia, developed a free Java environment as a successor to Blue. Before installing BlueJ, you must have JDK version 1.3 or above installed. A free copy of BlueJ can be obtained from the company's official website. To enhance the learning and teaching of OOPs, this tool was built (object-oriented programming). The items can be produced and tested in real time using an interactive interface. BlueJ features a simpler user interface than other professional IDEs. It offers a wide range of teaching materials that are tailored to its objectives. Standard development tools, such as an editor, compiler, and runtime environment, are also available to developers. BlueJ has a number of features, including:

- Work can be done more efficiently.
- A sample program is shown to give you a taste of programming.
- It's easy to debug because it shows faults at the bottom of the screen.
- Simpler than any other IDE in terms of user interface
- Other IDEs do not have some of these new features.
- Any platform that supports java may run BlueJ. This includes Windows, MacOS, and Linux.
- It gives you the ability to interact with various objects.

The accessor and mutator method are used in order to return the values and assign new terms and values. The constructors of the classes are also assigned with parameters which are to be accepted. The attributes are also assigned with different values. Each of the attributes of all classes have the accessor method or the get method and the mutator method is also used in some attributes to assign or set new values.

2. Class Diagram

Introduction:
A class diagram is a static representation. It reflects an application's static view. Not only is a class diagram used to visualize, describe, and log various aspects of a system, but it is often used to construct executable code for the software application. The class diagram depicts the properties and operations of a class, as well as the constraints that the system faces. A class diagram is a visual representation of a set of groups, interfaces, relationships, partnerships, and constraints. The aim of a class diagram is to represent an application's static view. As a coder, you're probably familiar with the Class Diagram (UML). As the static view of an application, it consists of a set of classes and interfaces as well as associations, collaborations, and restrictions. As a result, class diagrams are also utilized to create executable code for forward and reverse engineering. An application's class diagrams are used to represent distinct components of an application and to model static views. Object-oriented languages can directly map UML diagrams. Class diagrams are the only diagrams that can be explicitly mapped to object-oriented languages, making them extremely useful during the construction process. (TutorialsPoint, 2021)

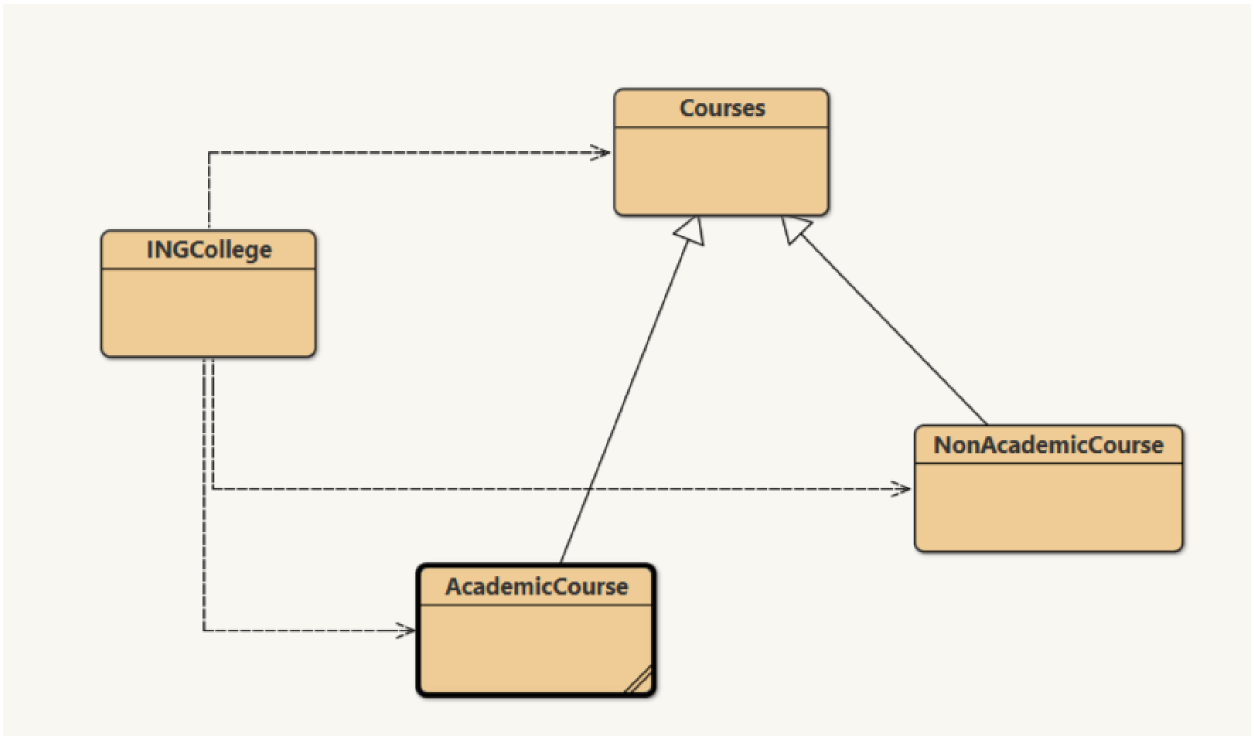


Figure 1: Class Diagram of Classes in BlueJ

INGCollege

INGCollege
-frame, Displayframe: Frame -PMenu, P1, P2, P3: Panel -lblForm, lblHello, lblProcess, lblRead, lblX, lbli, lblii, lbliii, lbliv, lblv, lblCopy, lblStartingDate, lblCompletionDate, lblNumberofAssessments, lblLecturerName, lblCourseLeader, lblCoursesID, lblLevel, lblCredit, lblCourseName, lblAcaDuration, lblForm1, lblForm2, lblDuration, lblPrequisites, lblCourseID, lblNonCourseID, lblNonCourseName, lblNonCoursesID, lblNonCourseLeader, lblInstructorName, lblStartDate, lblNCompletionDate, lblExamDate: Label -txtCourseID, txtCourseName, txtAcaDuration, txtLevel, txtCompletionDate, txtLecturerName, txtStartingDate, txtCredit, txtCourseLeader, txtCoursesID, txtNumberofAssessment, txtNonCourseID,txtNCompletionDate, txtNonCourseName, txtDuration, txtPrerequisites, txtNonCoursesID, txtNonCourseLeader, txtInstructorName, txtStartDate, txtExamDate: TextField -btnHome, btnAca, btnNonAca, btnBack, btnAcaADD, btnAcaRegister, btnAcaDisplay, btnAcaClear, btnNonAcaADD, btnNonAcaRegister, btnRemove, btnNonAcaDisplay, btnNonAcaClear: Button -AcademicList: Courses (ArrayList) -NonAcademicList: Courses (ArrayList) -obj1: Courses -obj2: Courses -x: int -y: int
-INGCollege () +AcaClear () +NonAcaClear () +actionPerformed (actionEvent e) +new INGCollege()

Table 1: Class diagram of INGCollege

3. Pseudocode

Introduction:
Pseudocode is an unofficial way of describing programming which does not need any strict syntax or technological considerations underlying programming language. It is used to create a program outline or rough draft. Phrase code is a plain-language explanation of the stages in an algorithm or another system in computer science and mathematics. Many of the principles of a normal programming language are used in pseudocode, however it is designed for human reading rather than machine reading Variable declarations and language-specific code are often left out.

There are plain language descriptive details or compact mathematical notation added to the programming language. Pseudocode summarizes the flow of a program but excludes underlying data. System designers write pseudocode to ensure that programmers recognize the specifications of a software project and align code accordingly. An effective main concept of an algorithm is pseudocode. It is used in the preparation of an algorithm to draw up the framework of the program. Using pseudocode has the advantage of being easier to grasp than standard programming language code, as well as being an efficient and environment-independent explanation of an algorithm's fundamentals. Textbooks and scientific journals utilize it to document algorithms, as well as for software design, among other things. To a certain extent, pseudocode is similar to skeleton programs, which can be compiled with no errors at all It is possible to consider flowcharts, drakon charts, and Unified Modeling Language (UML) charts as graphical alternatives to pseudocode, but they require more space on the page to accommodate. Haggis, for example, is a language that bridges the gap between pseudocode and the code written in programming languages.

As a pseudocode program is not an executable program, there is no comprehensive standard for pseudocode syntax. However, there are certain limited standards (such as for academic assessment). At pseudo code level, capturing errors or incorrect program flow is helpful for creation because it's less expensive to capture than later. It uses short terms or basic syntax of English to write program code before it is translated into a particular programming language. This works to recognize flow errors at the highest level and understand the flow of data programming. (Bennett, 2021)

INGCollege Class

```
IMPORT packages in program
CREATE class INGCollege
        DEFINE UI components
        DEFINE arraylist named AcademicList
        DEFINE arraylist named NonAcademicList
        DEFINE instance variables of AcademicCourse & NonAcademicCourse
        DEFINE x & y


        CREATE constructor for the class
        CREATE components

                DO
                        DEFINE frame
                                SET size
                                SET Visible
                                SET Layout
                                SET Resizable

                        DEFINE Panels
                                SET Background color
                                SET Bounds
                                SET Layout
                                SET Visible
                                ADD in frame
                        DEFINE Labels AND TextFields

                                SET Bounds

                                SET Font

                                SET Foreground color

                                ADD in panel
                        DEFINE Buttons

                                SET Font

                                SET Background color

                                SET Bounds

                                SET Visible
                **SET** Focusable
                **ADD** in panel
            **REGISTER** buttons in ActionListener
        **END DO**

        **IF**
            **SET** P1 visible
            **SET** P2 & P3 not-visible in frame
        **END IF**

        **IF**
            **SET** P2 visible
            **SET** P1 & P3 not-visible in frame
        **END IF**

        **IF**
            **SET** P3 visible
            **SET** P1 & P2 not-visible in frame
        **END IF**

        **IF** (back button is pressed)
                **SHOW** P1 panel
            **END IF**
    **IF** (Add button is pressed)
        **IF** (Text fields are empty)
            **SHOW** dialog message
        **END IF**
        **IF** (Text Fields are not empty)
            **GET** values from Aca TextFields
            FOR (Courses in AcademicList)
                IF (CourseID is stored in array then text is entered in
    GUI)
                    CALL AcaClear ()
                    SHOW Dialog message
                END IF
            END FOR
    TRY (checking if integer is entered in NOD and duration)
        PARSE NOA and Duration into integer
    END TRY

    CATCH (String is entered in NOA and Duration)
        SHOW dialog message
    END CATCH
    IF
        CREATE object of Academic Course and passing values
        ADD object in Array List
        SHOW dialog message
    END IF

    IF (register button is pressed)
        IF (Text fields are empty)
            SHOW dialog message
        END IF
        IF (Text fields are not empty)
            GET   values from Academic Text fields
            FOR
                IF (CourseID is stored in array)
                    CREAT object of Academic Course
                    ADD +1 on count
                END IF
                IF (isRegistered is false)
                    REGISTER courses
                    SHOW Dialog message
                END IF
            IF (isRegistered is True)
                SHOW dialog message
            END IF
            END FOR
        IF
            DO
                CREAT method btnAcaClear
                    SET TextFields empty
            END DO
            DO
                DEFINE method actionPerformed ActionEvent (e)
                    IF (add button is clicked)
                        CALL AcaClear
                        SHOW Dialog message
                    END IF
            END DO
        IF (Display button is clicked)
            DEFINE default table mode
            DEFINE table
            DEFINE columns
            FOR (length of column)
                ADD column in table mode
```

```
                END FOR
                FOR (Academic list size)
                    CREATE object of Academic Course
                    CONVERT integer into String
                    INSERT row count
                END FOR
                DEFINE
                    SET size
                    SET visibility
                    ADD Scroll pane

        IF (Add button is pressed)
            IF (Text fields are empty)
                SHOW dialog message
            END IF
            IF (Text Fields are not empty)
                GET values from NonAca TextFields
                FOR (Courses in NonAcademicList)
                    IF (CourseID is stored in array then text is entered in
        GUI)
                        CALL NonAcaClear ()
                        SHOW Dialog message
                    END IF
                END FOR
        TRY (checking if integer is entered in duration)
            PARSE Duration into integer
        END TRY

        CATCH (String is entered in Duration)
            SHOW dialog message
        END CATCH
        IF
            CREATE object of Academic Course and passing values
            ADD object in Array List
            SHOW dialog message
        END IF

        IF (register button is pressed)
            IF (Text fields are empty)
                SHOW dialog message
            END IF
            IF (Text fields are not empty)
                GET   values from Academic Text fields
                FOR
                    IF (CourseID is stored in array)
                        CREAT object of Academic Course
                        ADD +1 on count
                    END IF
                    IF (isRegistered is false)
                        REGISTER courses
                        SHOW Dialog message
                    END IF
                IF (isRegistered is True)
                    SHOW dialog message
                END IF
                END FOR
```

```
IF (register button is pressed)
    IF (Text fields are empty)
        SHOW dialog message
    END IF
    IF (Text fields are not empty)
        GET   values from NonAcademic Text fields
        FOR
            IF (CourseID is stored in array)
                CREAT object of NonAcademic Course
                ADD +1 on count
            END IF
            IF (isRegistered is false)
                REGISTER courses
                SHOW Dialog message
            END IF
        IF (isRegistered is True)
            SHOW dialog message
        END IF
        END FOR
    IF (Display button is clicked)
        DEFINE default table mode
        DEFINE table
        DEFINE columns
        FOR (length of column)
            ADD column in table mode
        END FOR
        FOR (NonAcademic list size)
            CREATE object of NonAcademic Course
            CONVERT integer into String
            INSERT row count
        END FOR
        DEFINE
            SET size
            SET visibility
            ADD Scroll pane


    IF
        DO
            CREAT method btnNonAcaClear
                SET TextFields empty
        END DO
        DO
            DEFINE method actionPerformed ActionEvent (e)
                IF (add button is clicked)
                    CALL NonAcaClear
                    SHOW Dialog message
                END IF
        END DO
```

4. Method Description

INGCollege Class:

Import ()

- ✓ Import java package is used to access package and its classes into the java program.

Public ()

- ✓ It is a constructor used to create object in class INGCollege.

ActionListener ()

- ✓ It is responsible for handling all action events such as clicking on components etc.

actionPerformed()

- ✓ It invokes when an action occurs. The action performed message is sent to all action listeners that are registered on the relevant component.

setSize ()

- ✓ The setSize(x, y) method makes a rectangular area of x pixels wide by y pixels high.

setVisible ()

- ✓ The setVisible method makes the frame appear on the screen if it's set on true but if it's set on false the frame doesn't appear on the screen.

setLayout

- ✓ The setLayout method allows us to set the layout of the container to whatever form or design we desire.

setResizeable ()

- ✓ The setResizeable parameter ensures that the GUI looks the way we intend from maximizing it to minimizing or not being able to resize it.

setBounds ()

- ✓ The setBounds method is used to define the bounding of a certain container. It includes position and size.

setBackground (Color)

- ✓ The setBackground (Color) sets the background color of its component.

void add ()

- ✓ The add method in java is used to add a specific element into a set collection.

setForeground(Color)
- ✓ The setForeground(Color) is used to add color to the written text.

setFocusable ()
- ✓ The setFocusable is used to set automatically keypad function on edittext box so after activity starts it will automatically select or deselect the text.

setFont ()
- ✓ The setFont is used to set the theme font, the text's size and its design.

setText ()
- ✓ The setText is used to set the current text that is ready by this String character.

getSource ()
- ✓ The getSource method is used to determine which button was clicked on actionPerformed.

getText ()
- ✓ The getText is used to get the current text that is ready by its String character.

getcourseID ()
- ✓ It is an accessor method which is used to get value of courseID.

getcoursename ()
- ✓ It is an accessor method which is used to get value of course_name.

getcourseduration ()
- ✓ It is an accessor method used to get value of course_duration.

getLevel ()
- ✓ It is an accessor method used to get value of Level.

getCredit ()
- ✓ It is an accessor method used to get value of Credit.

getLecturername ()
- ✓ It is an accessor method used to get value of Lecturername.

getNumberofassessments ()
- ✓ It is an accessor method used to get the value of NumberofAssessments.

getStartingDate()
- ✓ It is an accessor method used to get the value of StartingDate.

getCompletionDate();
- ✓ It is an accessor method used to get the value of CompletionDate.

getprerequisite ()
- ✓ It is an accessor method used to get the value of prerequisite.

getcourseleader()
- ✓ It is an accessor methos used to get the value of courseleader.

getInstructorname ()
- ✓ It is an accessor method used to get the value of instructor name.

getStartDate ()
- ✓ It is an accessor method used to get the value of StartDate.

getCompletionDate ()
- ✓ It is an accessor method used to get the value of Completiondate.

getExamdate ()
- ✓ This is an accessor method which is used to get the value of Examdate.

void display ()
- ✓ It is a display method used to display output of class.

void Register ()
- ✓ It is a register method used to register the input of class.

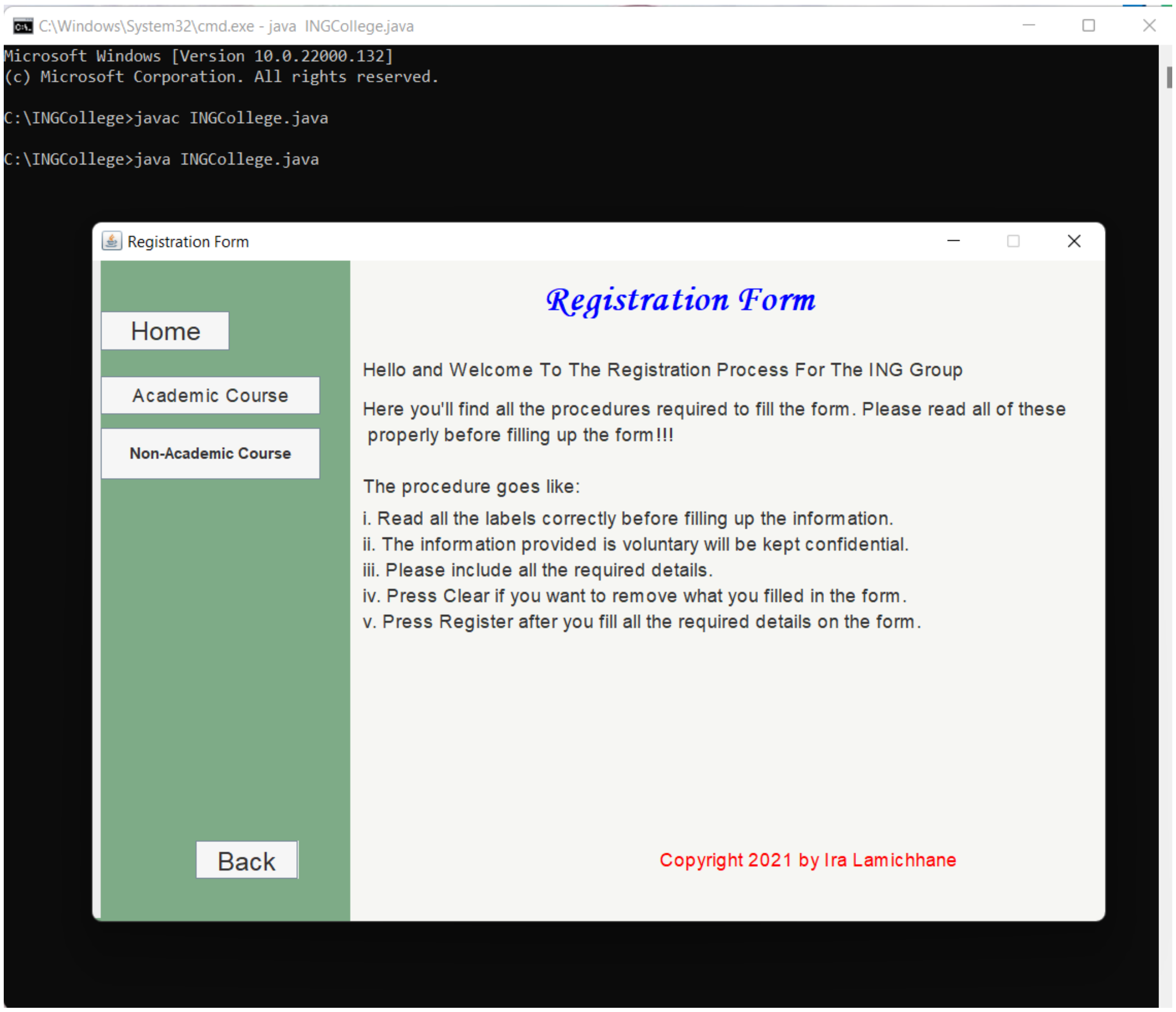
void Remove ()
- ✓ It is a remove method used to remove the input of the class.

showMessageDialog ()
- ✓ This method tells the user that something has happened.

isEmpty ()
- ✓ This method checks weather the String is empty or not.

5. Testing

I. Test 1:



```
C:\Windows\System32\cmd.exe - java  INGCollege.java
Microsoft Windows [Version 10.0.22000.132]
(c) Microsoft Corporation. All rights reserved.

C:\INGCollege>javac INGCollege.java

C:\INGCollege>java INGCollege.java
```

Registration Form

Registration Form

Hello and Welcome To The Registration Process For The ING Group

Here you'll find all the procedures required to fill the form. Please read all of these properly before filling up the form!!!

The procedure goes like:

i. Read all the labels correctly before filling up the information.
ii. The information provided is voluntary will be kept confidential.
iii. Please include all the required details.
iv. Press Clear if you want to remove what you filled in the form.
v. Press Register after you fill all the required details on the form.

Copyright 2021 by Ira Lamichhane

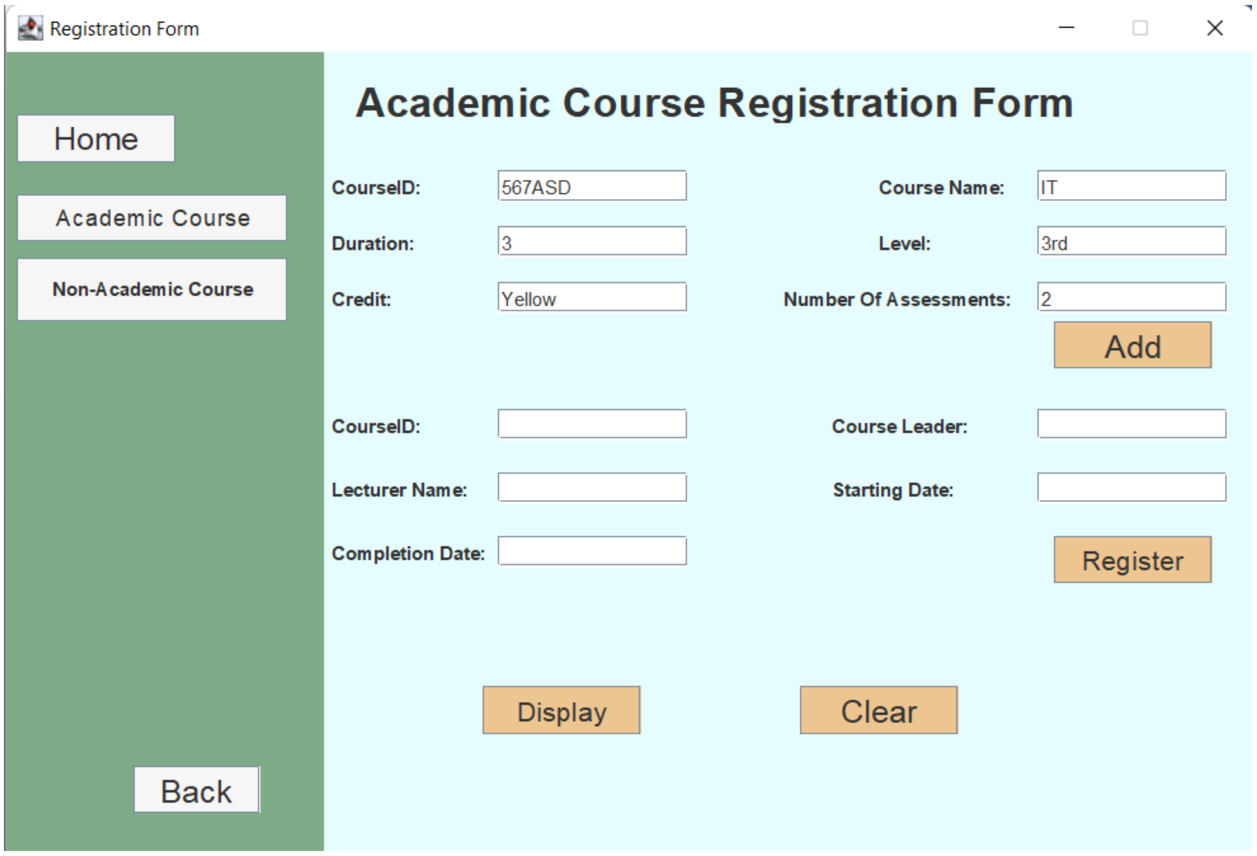
Figure 2: Testing of class through command prompt

II. Test 2:

a) Add course for Academic course:

Test No:	i
Objective:	To Add Course for Academic Class
Action:	>> Add Course is called with the following argument: CourseID: 567ASD Course Name: IT Duration: 3 Level: 3rd Credit: Yellow Number of Assessment: 2 >> Press the Add button
Expected Result:	Academic Course will be added
Actual Result:	Academic Course was added
Conclusion:	The Test was successful

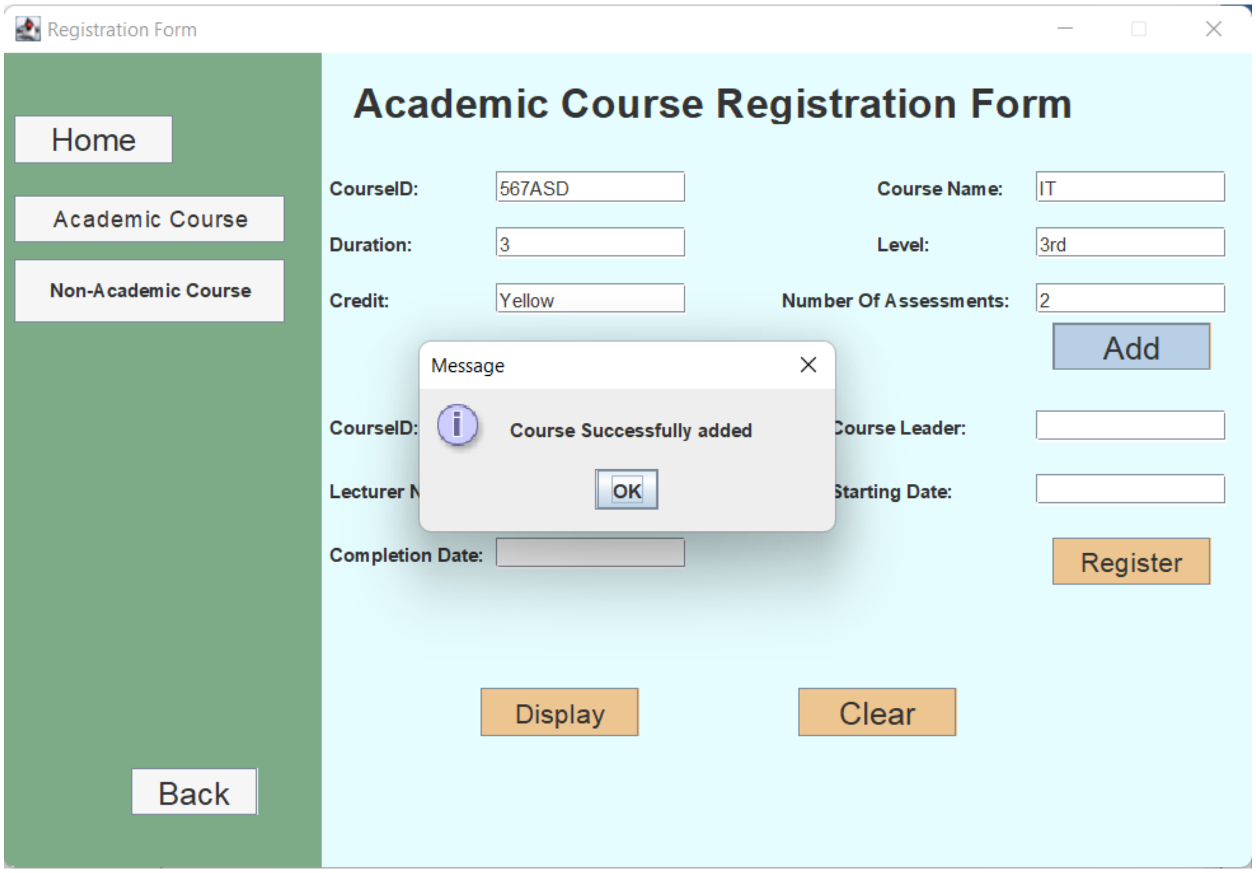
Table 2: Test of ADD button for Academic Course



Academic Course Registration Form

CourseID:	567ASD	Course Name:	IT
Duration:	3	Level:	3rd
Credit:	Yellow	Number Of Assessments:	2
CourseID:		Course Leader:	
Lecturer Name:		Starting Date:	
Completion Date:			

Figure 3: Screenshot of Filled textfield of ADD on Aca course



Academic Course Registration Form

CourseID: 567ASD
Course Name: IT
Duration: 3
Level: 3rd
Credit: Yellow
Number Of Assessments: 2
CourseID:
Course Leader:
Lecturer N
Starting Date:
Completion Date:

Message

Course Successfully added

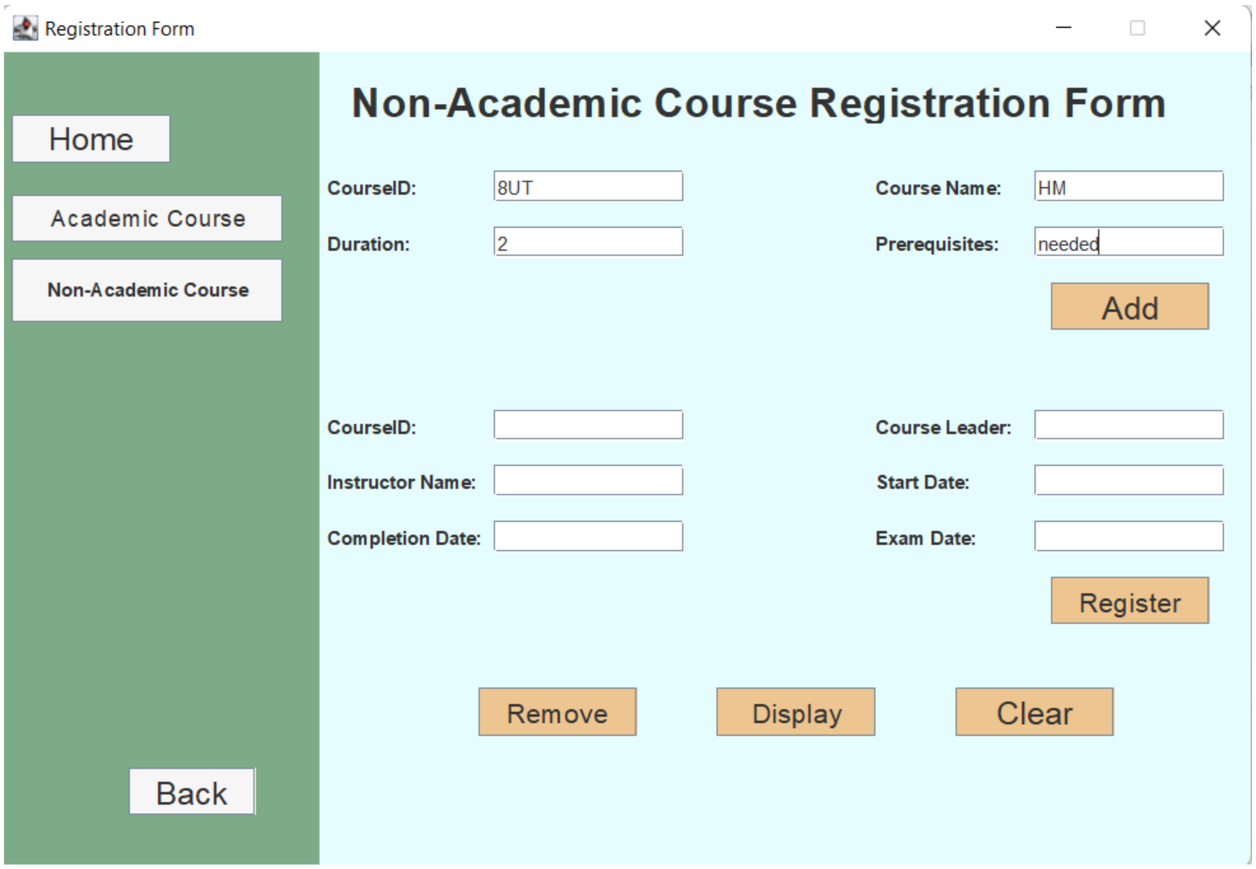
Figure 4: Screenshot of the Course being successfully added on ACA Course

And the course finally got successfully added.

b) ADD course for Non-academic course

Test No:	ii
Objective	To ADD course for Non-Academic Course
Action:	>> Non-Academic Course is called with the following argument: CourseID: 8UT Course Name: HM Duration: 2 Prerequisite: needed >>Press the Add button
Expected Result:	Non-Academic Course would be added
Actual Result:	Course was added.
Conclusion:	Test was successful.

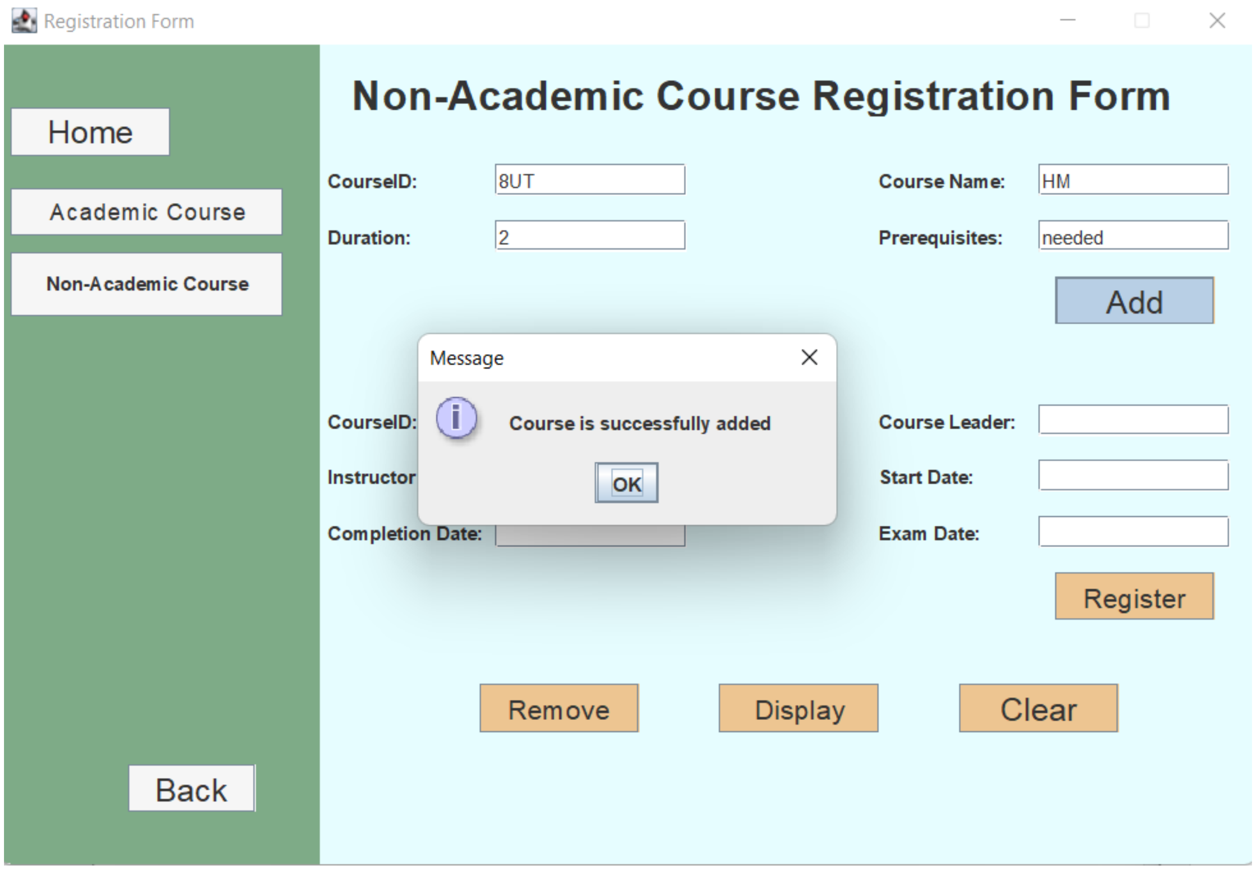
Table 3: Testing of ADD button for NON-ACA Course



CourseID:	8UT	Course Name:	HM
Duration:	2	Prerequisites:	needed
CourseID:		Course Leader:	
Instructor Name:		Start Date:	
Completion Date:		Exam Date:	

Figure 5: Screenshot of filled text fields of NON-ACA Course

Now, pressing the Add button.


Registration Form

Non-Academic Course Registration Form

CourseID: 8UT
Course Name: HM
Duration: 2
Prerequisites: needed

Message
Course is successfully added

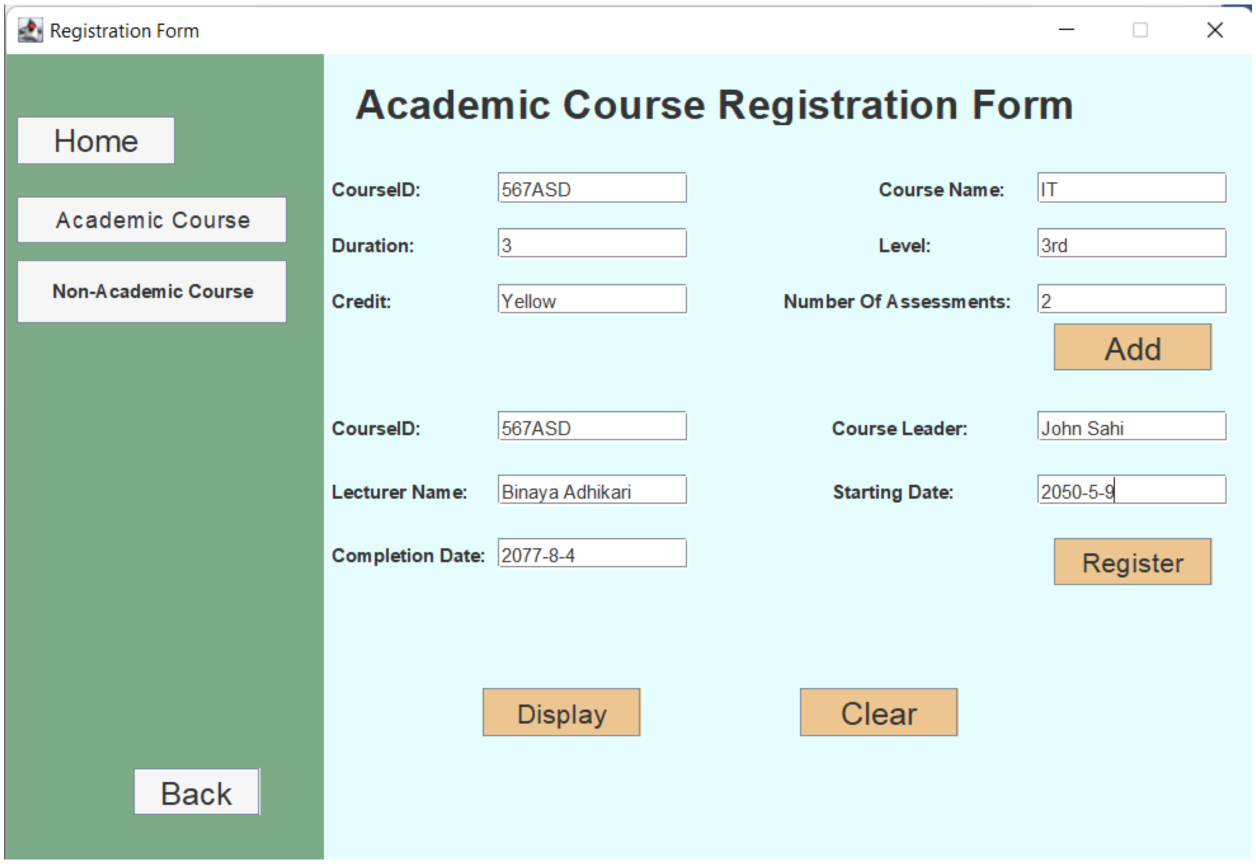
Figure 6: Screenshot of ADD button working successfully on NON-ACA Course

The course got added on Non-Academic Course Registration Form.

c) REGISTER academic course

Test No:	iii
Objective	To REGISTER course for Academic Course
Action:	>> Non-Academic Course is called with the following argument: CourseID: 567ASD Course Leader: John Sahi Lecturer: Name: Binaya Adhikari Starting Date: 2050-5-9 Completion Date: 2077-8-4 >>Press the Register button
Expected Result:	Academic Course would be registered
Actual Result:	Course was registered
Conclusion:	Test was successful.

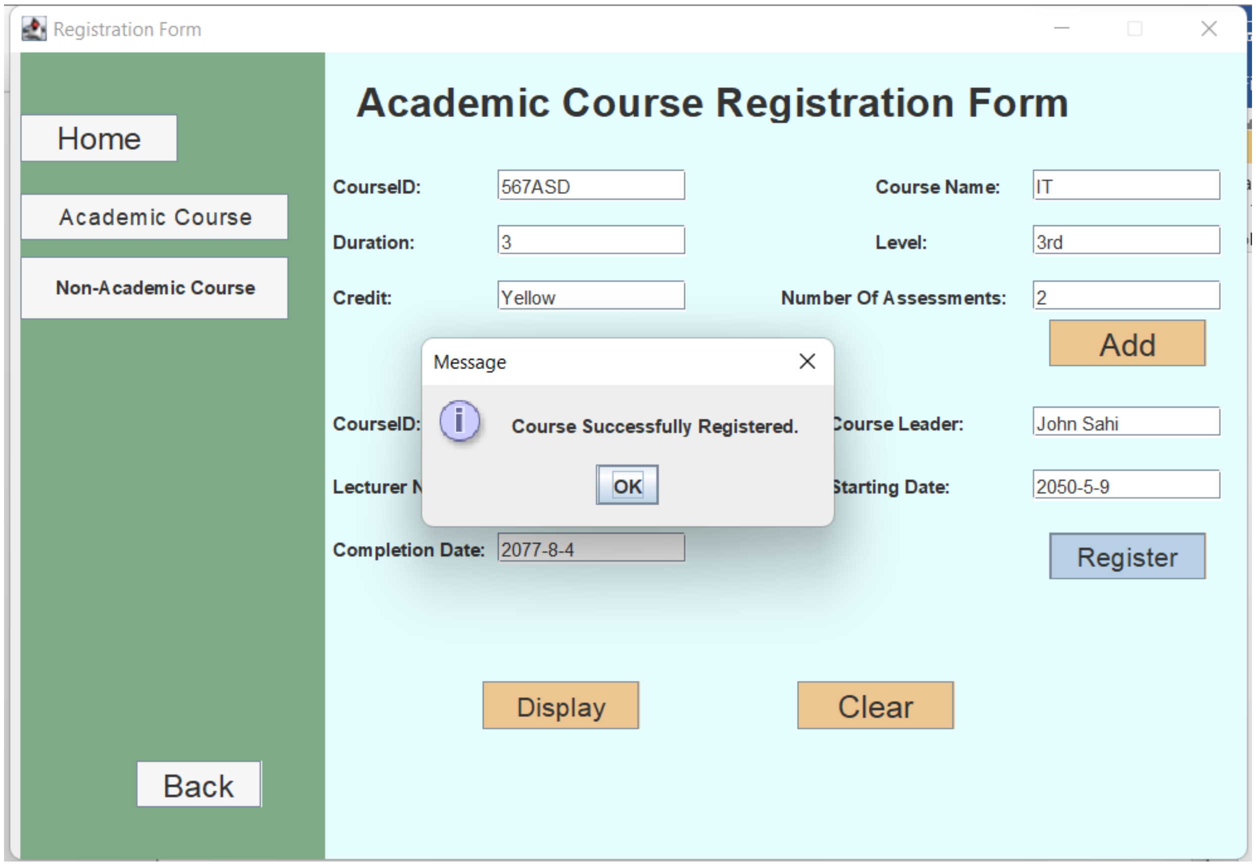
Table 4: Test for REGISTER button on ACA Course



CourseID:	567ASD	Course Name:	IT
Duration:	3	Level:	3rd
Credit:	Yellow	Number Of Assessments:	2
CourseID:	567ASD	Course Leader:	John Sahi
Lecturer Name:	Binaya Adhikari	Starting Date:	2050-5-9
Completion Date:	2077-8-4		

Figure 7: Filled text Fields of ACA Course on Register

Now, I shall press the Register button.



Academic Course Registration Form

CourseID:	567ASD	Course Name:	IT
Duration:	3	Level:	3rd
Credit:	Yellow	Number Of Assessments:	2
CourseID:		Course Leader:	John Sahi
Lecturer N		Starting Date:	2050-5-9
Completion Date:	2077-8-4		

Message

Course Successfully Registered.

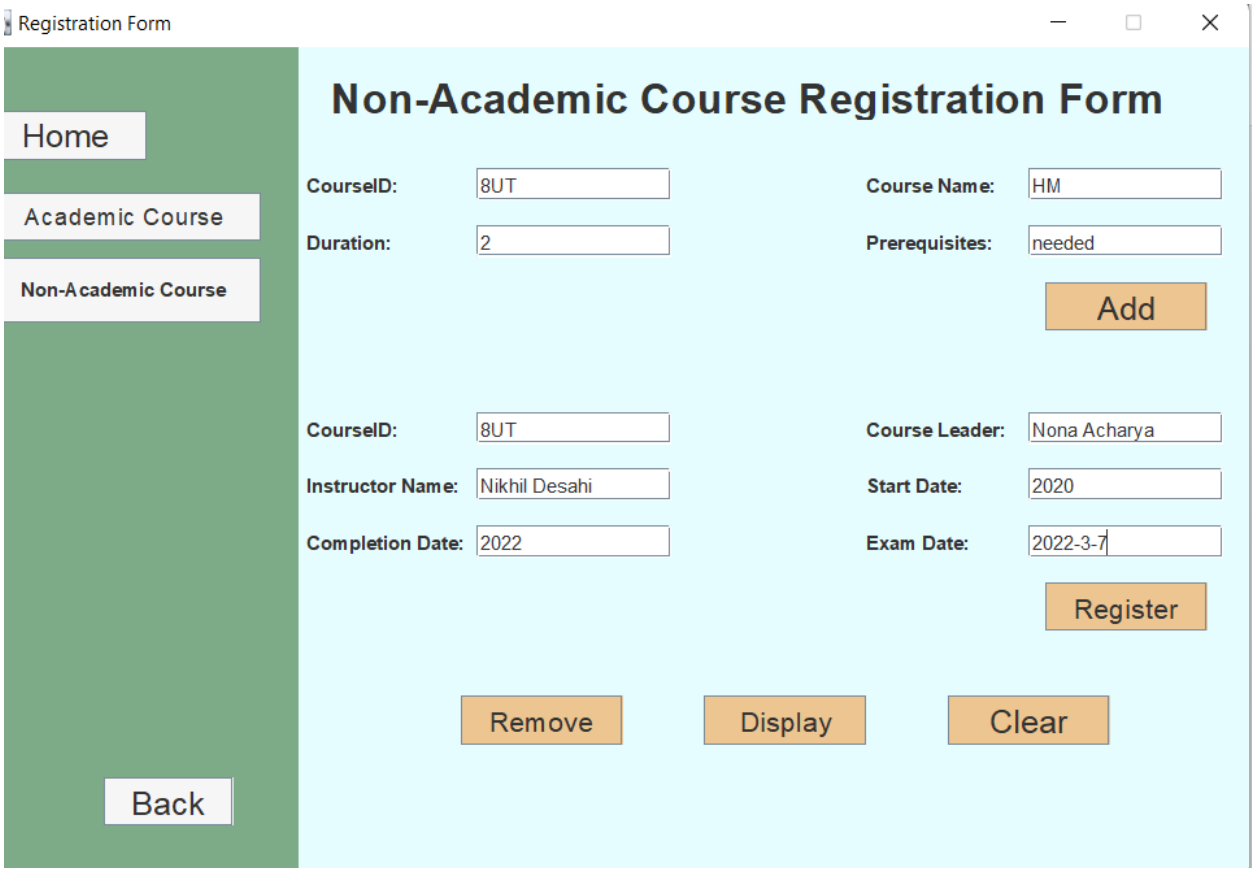
Figure 8: Screenshot of Register button successfully working on ACA Course

The Course got successfully registered.

d) REGISTER Non-Academic Course

Test No:	iv
Objective	To REGISTER course for Non-Academic Course
Action:	>> Non-Academic Course is called with the following argument: CourseID: 8UT Course Leader: Nona Acharya Istructor: Name: Nikhil Deshai Start Date: 2020 Completion Date: 2022 Exam Date: 2022-3-7 >>Press the Register button
Expected Result:	Non-Academic Course would be registered
Actual Result:	Course was registered
Conclusion:	Test was successful.

Table 5: Testing of Register button on ACA Course



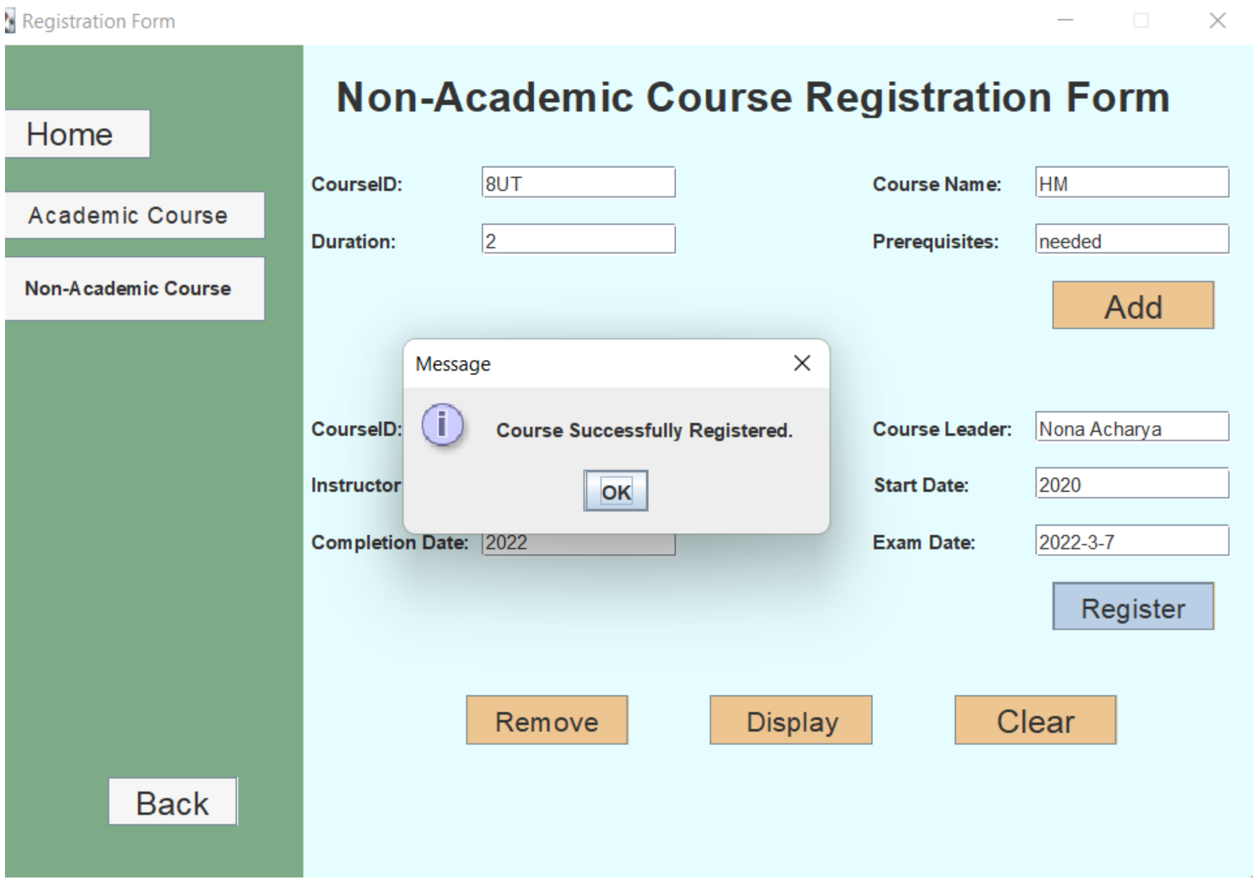
Non-Academic Course Registration Form

CourseID:	8UT	Course Name:	HM
Duration:	2	Prerequisites:	needed

CourseID:	8UT	Course Leader:	Nona Acharya
Instructor Name:	Nikhil Desahi	Start Date:	2020
Completion Date:	2022	Exam Date:	2022-3-7

Figure 9: Filled TextFields of Register field in NON-ACA Course

This is how the course look's before clicking the button Register.



Non-Academic Course Registration Form

CourseID: 8UT
Course Name: HM
Duration: 2
Prerequisites: needed
CourseID:
Course Leader: Nona Acharya
Instructor
Start Date: 2020
Completion Date: 2022
Exam Date: 2022-3-7

Message
Course Successfully Registered.

Figure 10: NON-ACA Course successfully registered

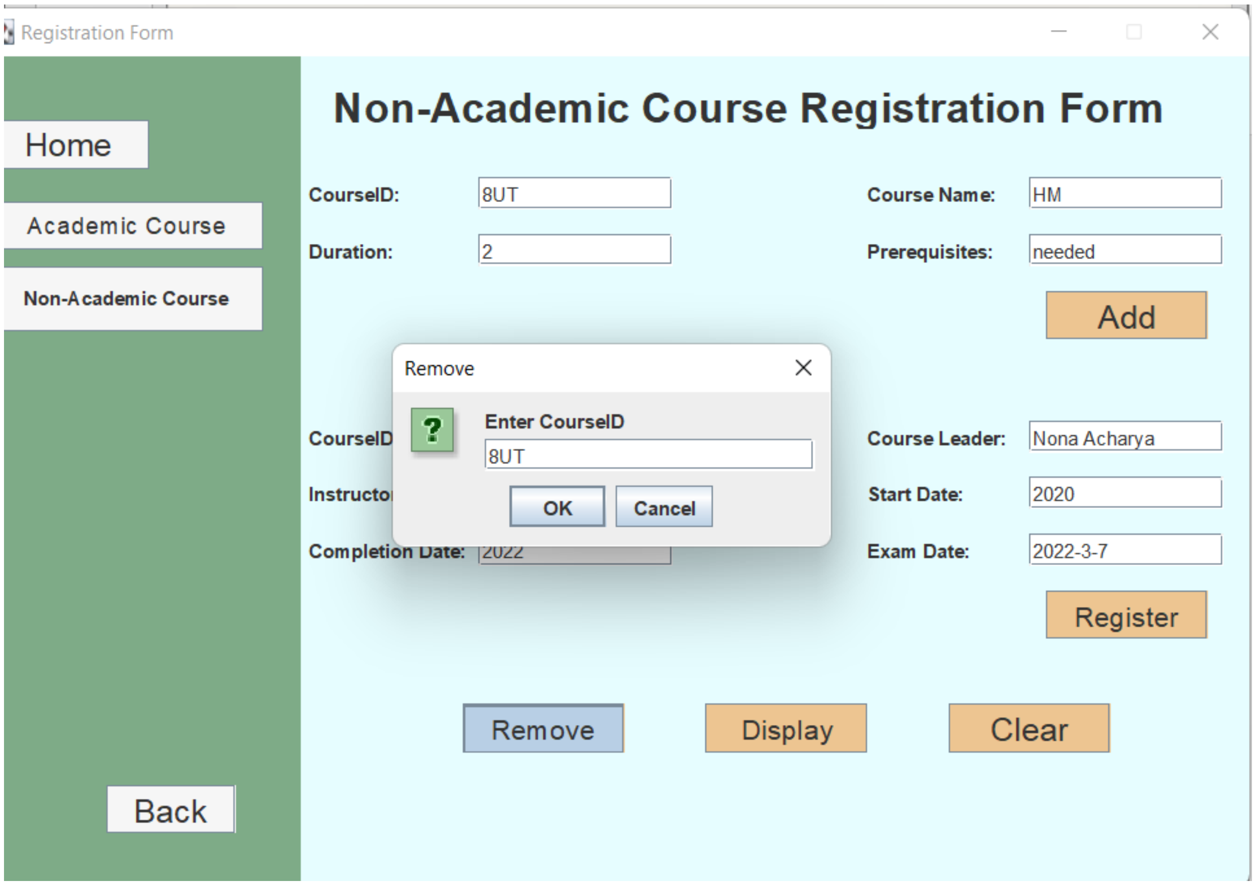
The course get's Registered successfully.

e) REMOVE Non-Academic Course

Test No:	v
Objective:	To Remove the Non-Academic CourseID
Action:	>>Non-Academic Course remove button was called and the registered CourseID was filled: CourseID: 8UT >> OK button was pressed
Expected Result:	Non-Academic Course would be removed.
Actual Result:	Course was removed.
Conclusion:	The Test was successful.

Table 6: Testing Remove button

Now Removing the non-academic course, we shall fill up the CourseID that we're supposed to remove and then press the button Remove.



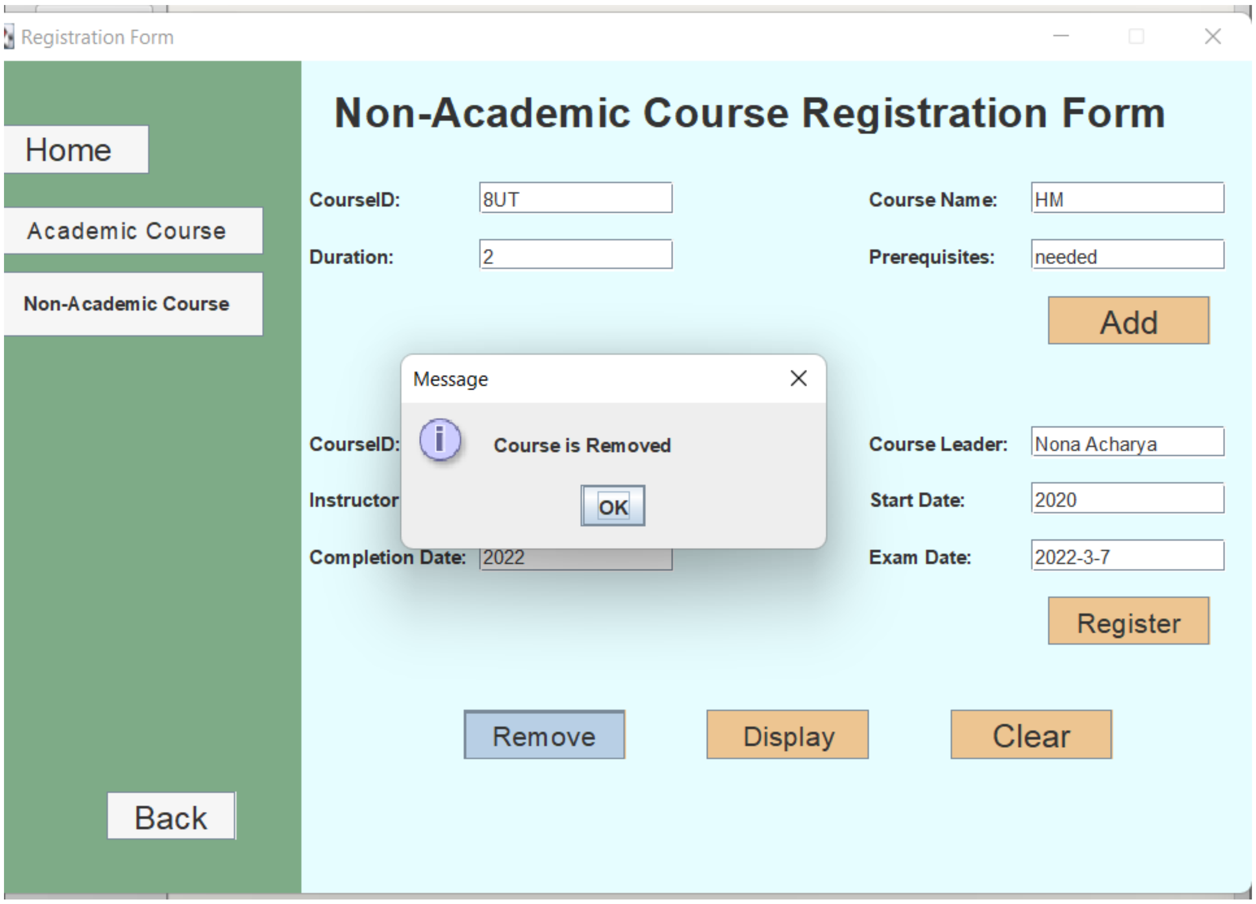
Non-Academic Course Registration Form

CourseID:	8UT	Course Name:	HM
Duration:	2	Prerequisites:	needed
CourseID		Course Leader:	Nona Acharya
Instructor		Start Date:	2020
Completion Date:	2022	Exam Date:	2022-3-7

Enter CourseID

8UT

Figure 11: Filling up the CourseID



Non-Academic Course Registration Form

CourseID: 8UT
Course Name: HM
Duration: 2
Prerequisites: needed

CourseID:
Course Leader: Nona Acharya
Instructor
Start Date: 2020
Completion Date: 2022
Exam Date: 2022-3-7

Course is Removed

Figure 12: Course gets Removed successfully

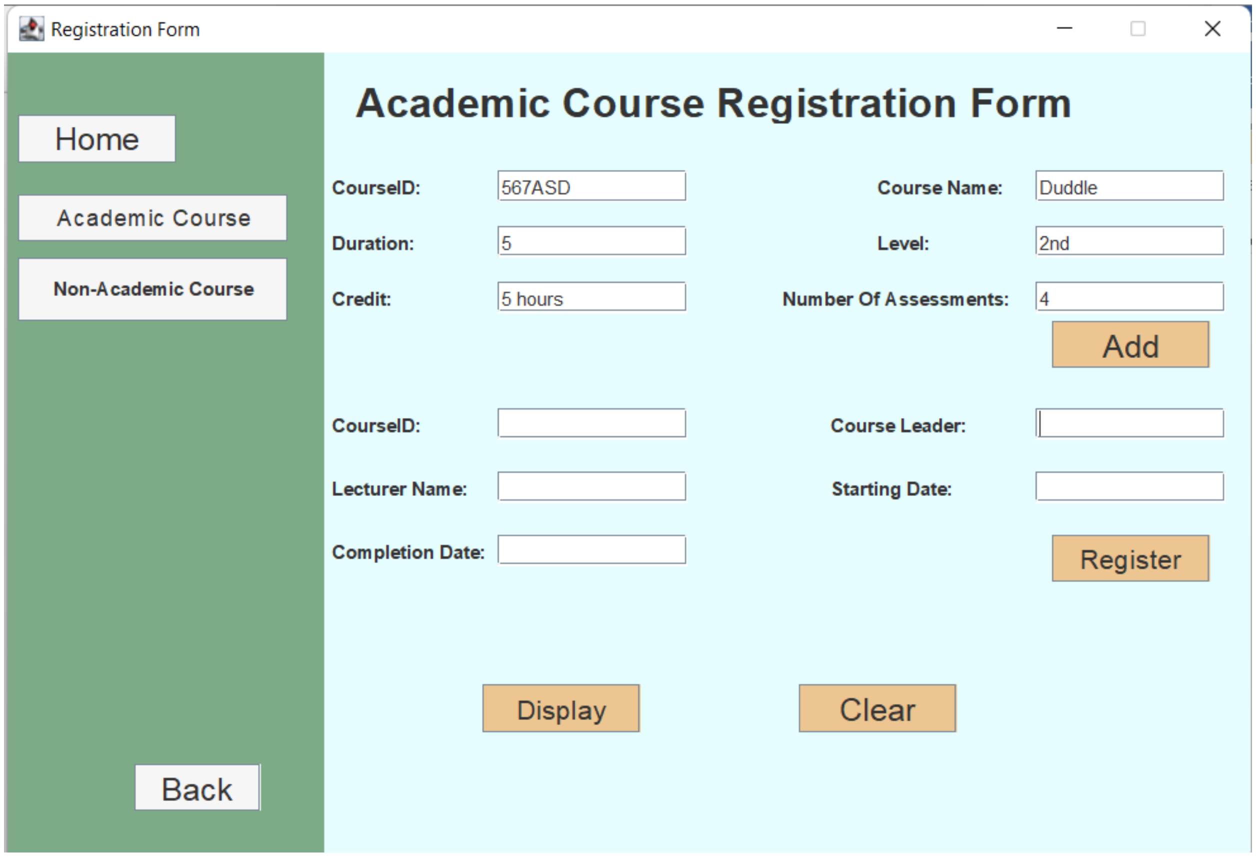
The course gets removed.

III. Test 3:

I. Add Duplicate CourseID

Test No:	i
Objective:	To add the same CourseID that was added before.
Action:	>> Academic Course is called with the following argument: CourseID: 567ASD Course Name: Duddle Duration: 5 Level: 2nd Credit: 5 hours Number of Assessment: 4 >> Press the Add button
Expected Result:	Dialog message will appear saying “CourseID is already added”
Actual Result:	Dialog box appeared saying CourseID was already added.
Conclusion	Test was successful.

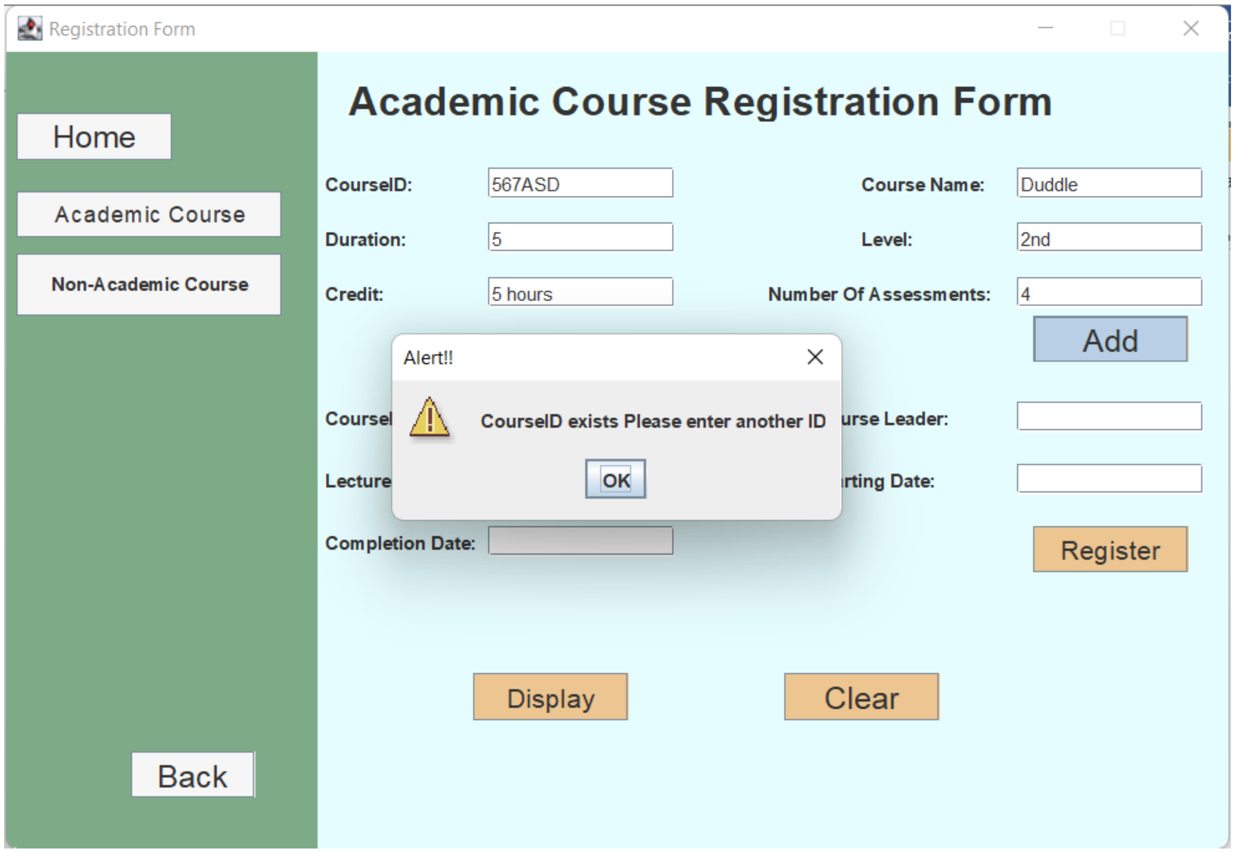
Table 7: Test by Adding DUplicate CourseID



Academic Course Registration Form

CourseID:	567ASD	Course Name:	Duddle
Duration:	5	Level:	2nd
Credit:	5 hours	Number Of Assessments:	4
CourseID:		Course Leader:	
Lecturer Name:		Starting Date:	
Completion Date:			

Figure 13:Filling up ta same CourseID



Alert!!

CourseID exists Please enter another ID

CourseID: 567ASD

Course Name: Duddle

Duration: 5

Level: 2nd

Credit: 5 hours

Number Of Assessments: 4

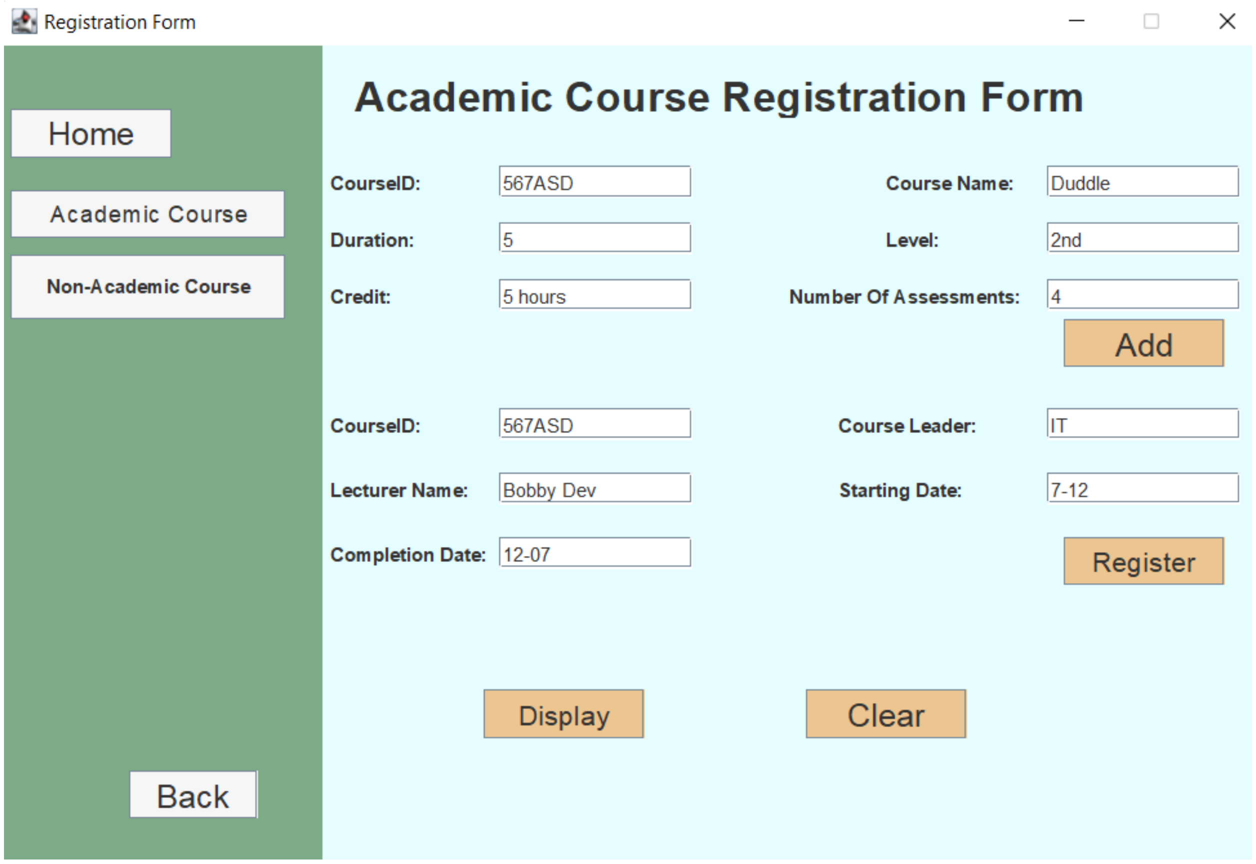
Completion Date:

Figure 14: Course doesn't get added

II. Registering Already registered Course

Test No:	ii
Objective:	To Register the CourseID that is already registered in Academic Course:
Action	>>Academic Course Register is called with the following argument: CourseID: 567ASD Course Name: IT Lecturer Name: Bobby Dev Starting Date: 7-12 Completion Date: 12-07 >>Register Button was pressed
Expected Result:	Dialog box will appear stating "Course is already registered"
Actual Result:	Dialog box appeared with the statement that CourseID was already Registered.
Conclusion:	Test was successful.

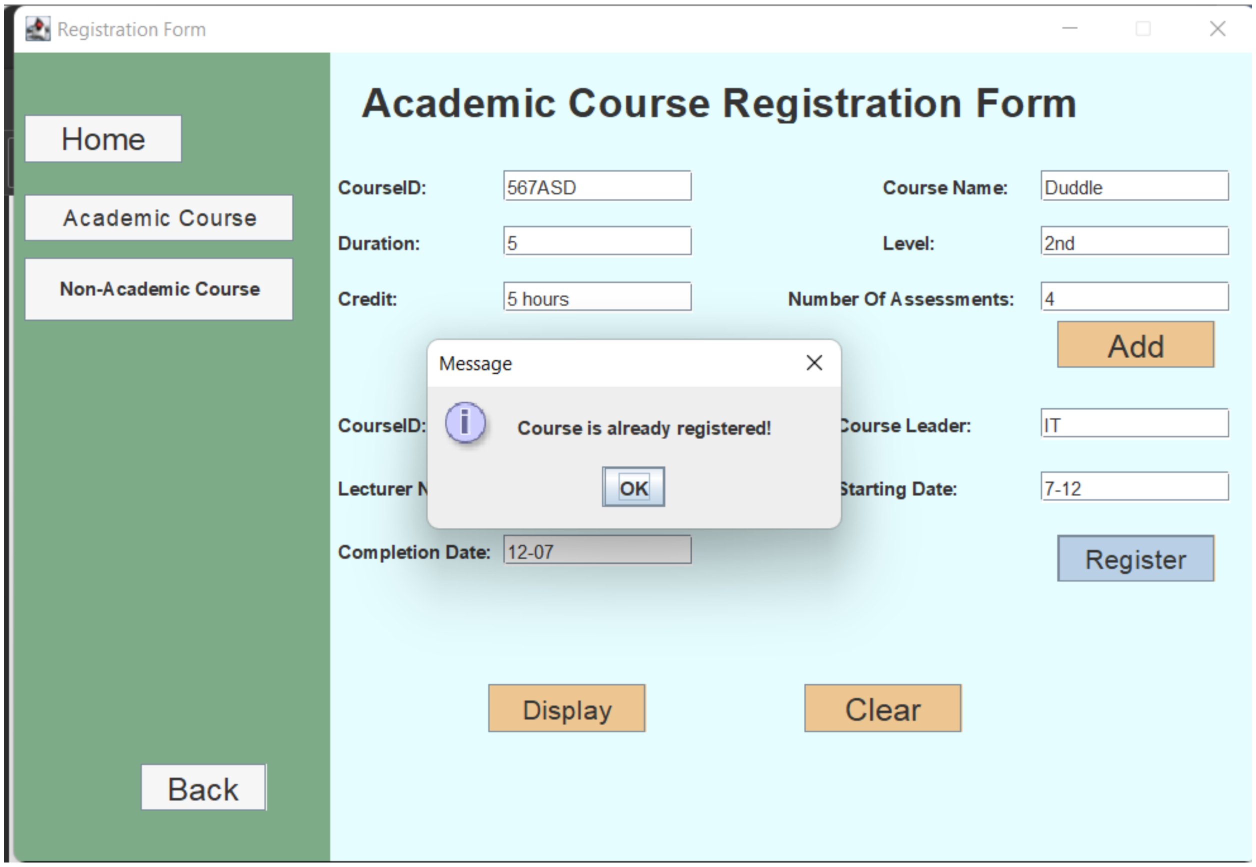
Table 8: Registering already registered Course



Academic Course Registration Form

CourseID:	567ASD	Course Name:	Duddle
Duration:	5	Level:	2nd
Credit:	5 hours	Number Of Assessments:	4
CourseID:	567ASD	Course Leader:	IT
Lecturer Name:	Bobby Dev	Starting Date:	7-12
Completion Date:	12-07		

Figure 15: Filling up the same courseID to register



Academic Course Registration Form

CourseID: 567ASD
Course Name: Duddle
Duration: 5
Level: 2nd
Credit: 5 hours
Number Of Assessments: 4
CourseID:
Course Leader: IT
Lecturer N
Starting Date: 7-12
Completion Date: 12-07

Message
Course is already registered!

Figure 16:Course doesn't get registered since it's already registered

III. Removing the Non-Academic Course that is already removed

Test No:	iii
Objective:	To remove the non-academic course that has been already removed
Action	>>
Expected Result:	
Actual Result:	
Conclusion:	

Table 9: Removing course ID that's already been removed

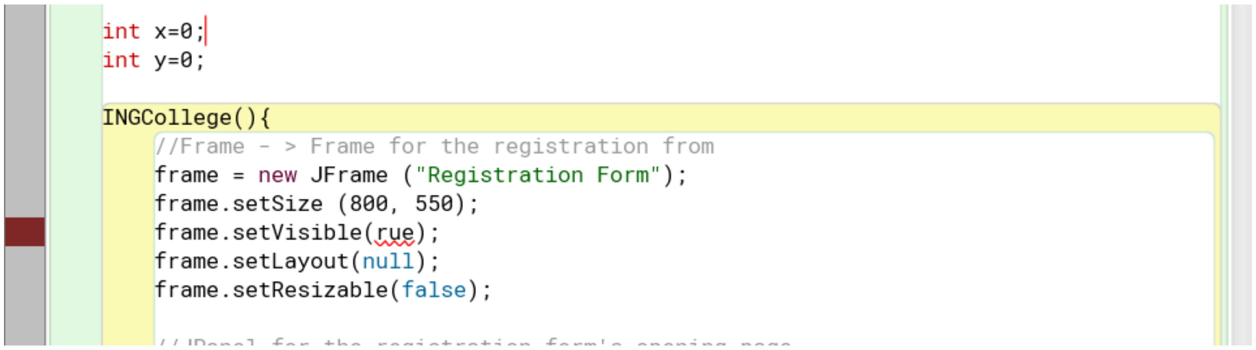
6. Errors Encountered during coding

i. Syntax Error:

When we write a statement that is not true according to the Java programming language's guidelines, we have made a syntax error. Many of them are like things that have semicolons missing, curly braces missing, and a myriad of other things. Java code syntactical errors occur when the language used to construct your code is improper. Even if the condition is on the same line as the if statement, if you try to build an if statement that doesn't include parentheses, you'll get an error. So be careful. Most of these errors will be caught by the compiler for you. For example, if you have an erroneous code syntax, the compiler will be unable to convert it into JRE byte code. Some of the most common syntax problems are listed below:

Detection of error:

The first error I encountered was while making the very first frame look visible. The frame.setVisible (true) was mistaken for frame.setVisible (rue) due to which the error happen.

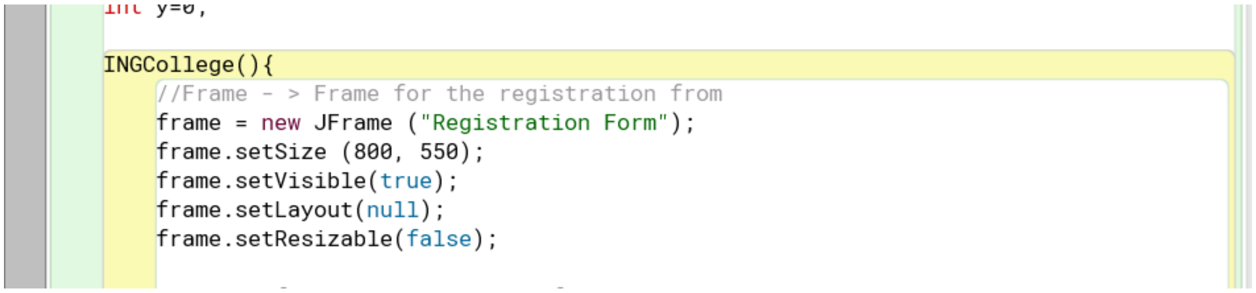


```
int x=0;
int y=0;

INGCollege(){
    //Frame - > Frame for the registration from
    frame = new JFrame ("Registration Form");
    frame.setSize (800, 550);
    frame.setVisible(rue);
    frame.setLayout(null);
    frame.setResizable(false);
```

Figure 17: Syntax error detection

Correction of error:

After making the frame.setVisible (rue) to frame.setVisible (true) the error was eliminated and the program got compiled easily.



```
INGCollege(){
    //Frame - > Frame for the registration from
    frame = new JFrame ("Registration Form");
    frame.setSize (800, 550);
    frame.setVisible(true);
    frame.setLayout(null);
    frame.setResizable(false);
```

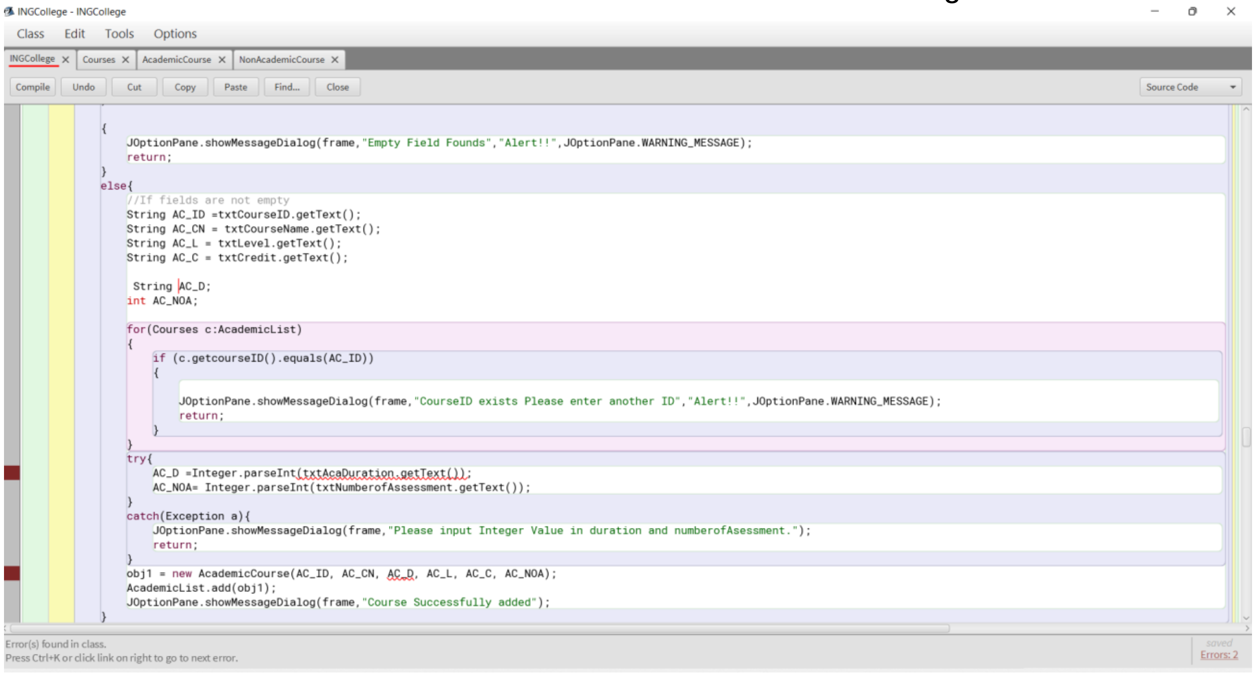
Figure 18: Syntax error correction

ii. Semantic error:

These errors appear during the semantic analysis step of the software. These types of errors are noticed at the time of compilation and can be corrected. When a compiler encounters a mistake, it is usually a scope or declaration error. Defined and undeclared identifiers, for example Another compile-time mistake is type mismatch. Use of the improper variable, operator, or order of operations can lead to a semantic error. Creating faulty program logic that creates wrong outcomes when run. Source code syntax may be correct, but algorithms may not. In some circles, a semantic error is also known as a "logic error"; nevertheless, some programmers think that logic errors result in incorrect information while semantic errors do not.

Detection of error:

The semantic error was detected when I denoted int value as String.


INGCollege - INGCollege
Class Edit Tools Options
INGCollege Courses AcademicCourse NonAcademicCourse
Compile Undo Cut Copy Paste Find... Close
Source Code
saved
Errors: 2

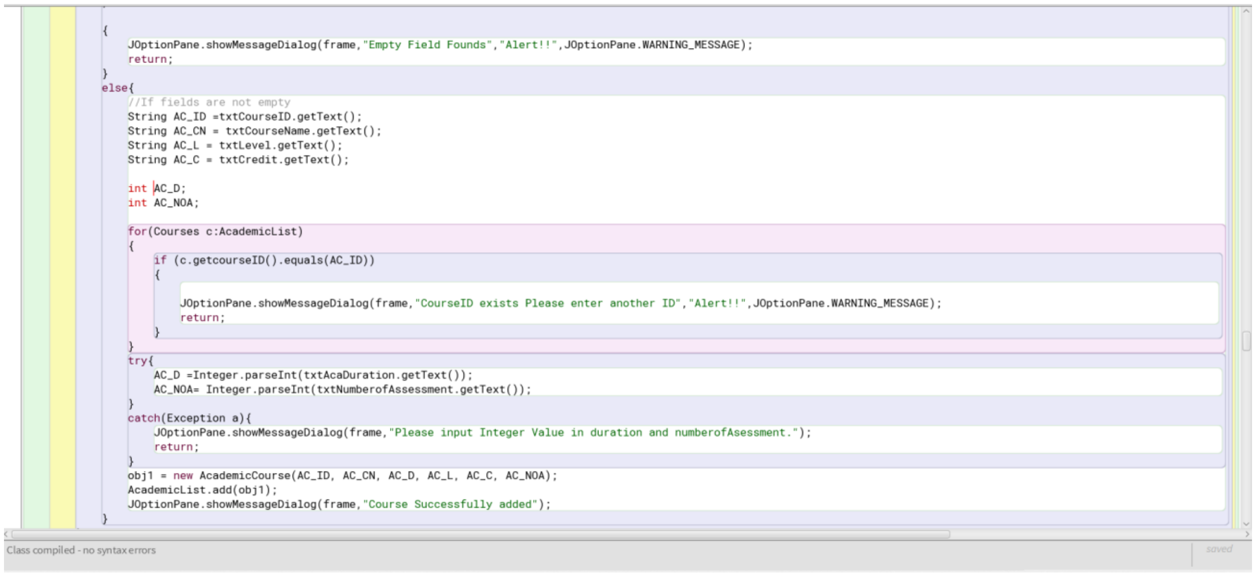
```
{
    JOptionPane.showMessageDialog(frame,"Empty Field Founds","Alert!!",JOptionPane.WARNING_MESSAGE);
    return;
}
else{
    //If fields are not empty
    String AC_ID =txtCourseID.getText();
    String AC_CN = txtCourseName.getText();
    String AC_L = txtLevel.getText();
    String AC_C = txtCredit.getText();

     String AC_D;
    int AC_NOA;

    for(Courses c:AcademicList)
    {
        if (c.getcourseID().equals(AC_ID))
        {

            JOptionPane.showMessageDialog(frame,"CourseID exists Please enter another ID","Alert!!",JOptionPane.WARNING_MESSAGE);
            return;
        }
    }
    try{
        AC_D =Integer.parseInt(txtAcaDuration.getText());
        AC_NOA= Integer.parseInt(txtNumberofAssessment.getText());
    }
    catch(Exception a){
        JOptionPane.showMessageDialog(frame,"Please input Integer Value in duration and numberofAsessment.");
        return;
    }
    obj1 = new AcademicCourse(AC_ID, AC_CN, AC_D, AC_L, AC_C, AC_NOA);
    AcademicList.add(obj1);
    JOptionPane.showMessageDialog(frame,"Course Successfully added");
}
```

Error(s) found in class.
Press Ctrl+K or click link on right to go to next error.

Figure 19: Detection of Semantic error

Correction of error:

After I corrected the String written in front of AC_D into int the error was corrected and the code complied.



```
{
    JOptionPane.showMessageDialog(frame,"Empty Field Founds","Alert!!",JOptionPane.WARNING_MESSAGE);
    return;
}
else{
    //If fields are not empty
    String AC_ID =txtCourseID.getText();
    String AC_CN = txtCourseName.getText();
    String AC_L = txtLevel.getText();
    String AC_C = txtCredit.getText();

    int AC_D;
    int AC_NOA;

    for(Courses c:AcademicList)
    {
        if (c.getcourseID().equals(AC_ID))
        {

            JOptionPane.showMessageDialog(frame,"CourseID exists Please enter another ID","Alert!!",JOptionPane.WARNING_MESSAGE);
            return;
        }
    }
    try{
        AC_D =Integer.parseInt(txtAcaDuration.getText());
        AC_NOA= Integer.parseInt(txtNumberofAssessment.getText());
    }
    catch(Exception a){
        JOptionPane.showMessageDialog(frame,"Please input Integer Value in duration and numberofAsessment.");
        return;
    }
    obj1 = new AcademicCourse(AC_ID, AC_CN, AC_D, AC_L, AC_C, AC_NOA);
    AcademicList.add(obj1);
    JOptionPane.showMessageDialog(frame,"Course Successfully added");
}
```

```
Class compiled - no syntax errors
```

Figure 20: Correction of Semantic Error

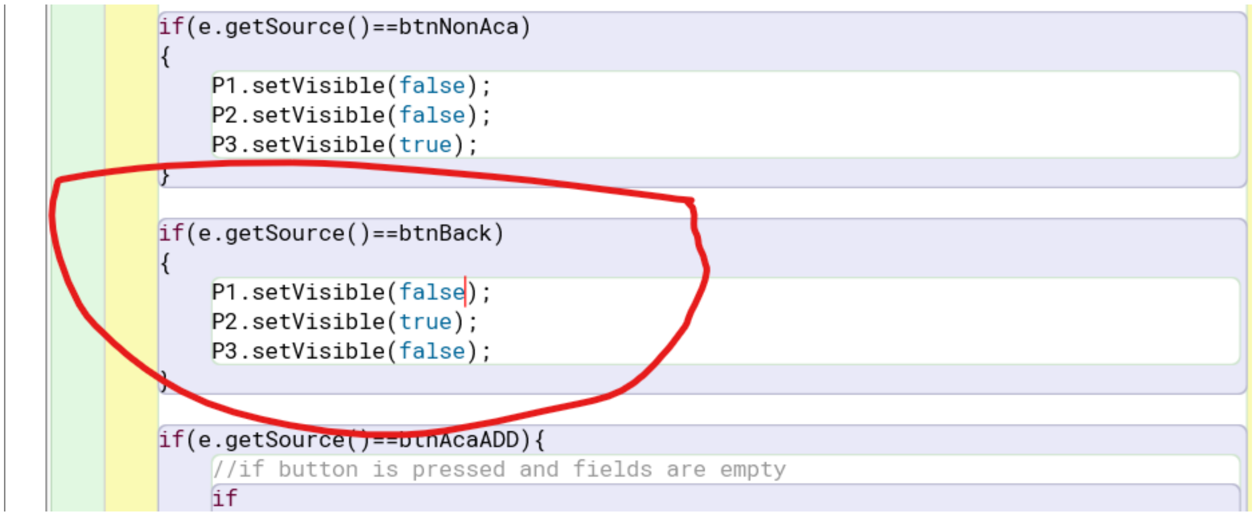
iii. Logical error:

The programmer commits a logical error while writing the software source code. Logical errors generally produce unpredictable logical results. Both the translator, the parser and the language of the script contain logical mistakes. Logical errors cannot be detected by the syntax nor by runtime errors compiler or translator. This occurs when your software compiles and runs, but does the wrong thing or gives an inaccurate result (or no output at all) when it should. Neither the compiler nor the JVM can detect these mistakes. In order to help you discover the mistake, the Java system does not disclose any extra information. Semantic errors are also known as logical errors. These problems occur when a programmer uses an improper idea or concept when coding. The difference between syntax and logical errors is that syntax errors are grammatical faults, while logic errors are errors coming from a faulty understanding.

Detection of Error:

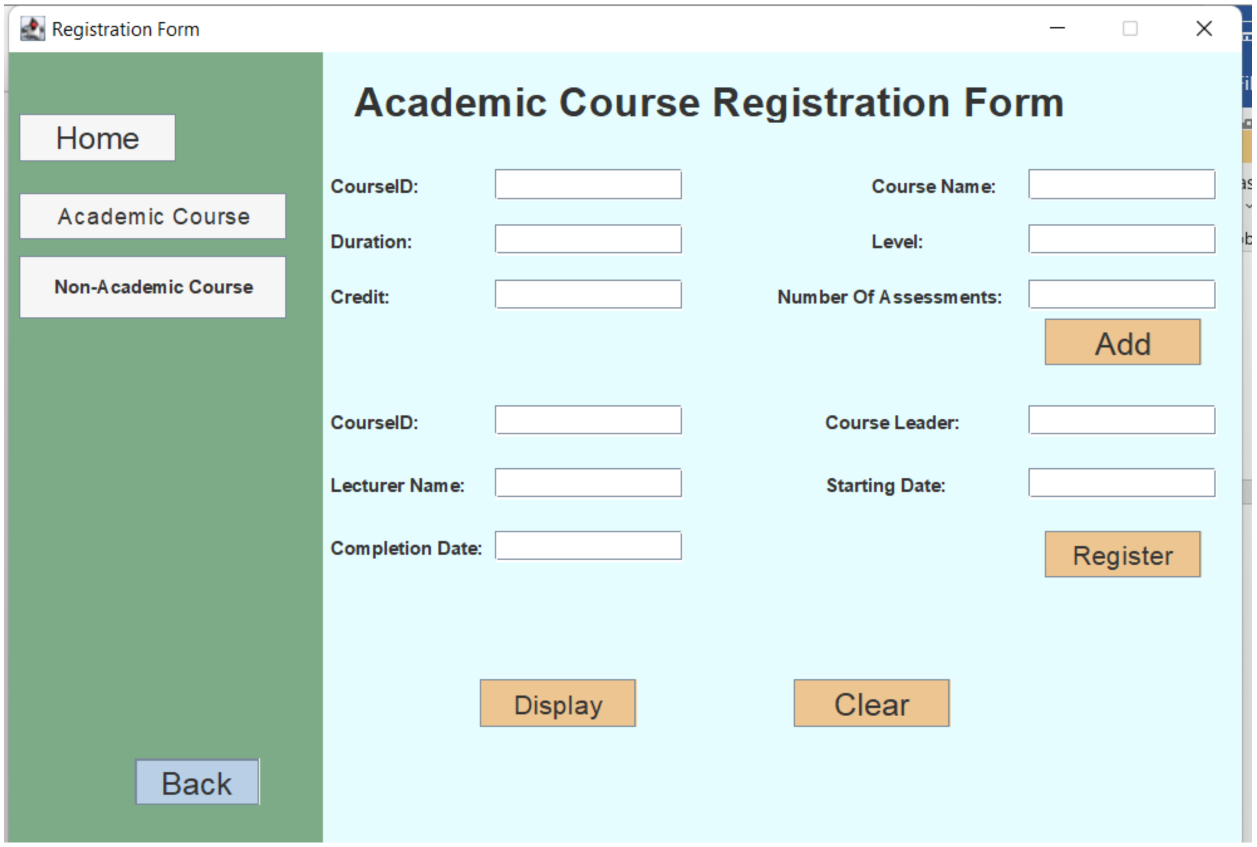
The logical error that I got into was while switching the panel. I accidentally wrote visibility false on the required panel and true on the menu panel and the other panel which was not supposed to be shown up got into the way.

When the back button was pressed instead of panel P1 the panel P2 opened



```
if(e.getSource()==btnNonAca)
{
    P1.setVisible(false);
    P2.setVisible(false);
    P3.setVisible(true);
}

if(e.getSource()==btnBack)
{
    P1.setVisible(false);
    P2.setVisible(true);
    P3.setVisible(false);
}

if(e.getSource()==btnAcaADD){
    //if button is pressed and fields are empty
    if
```

Figure 21: Detection of LOgical Error on code



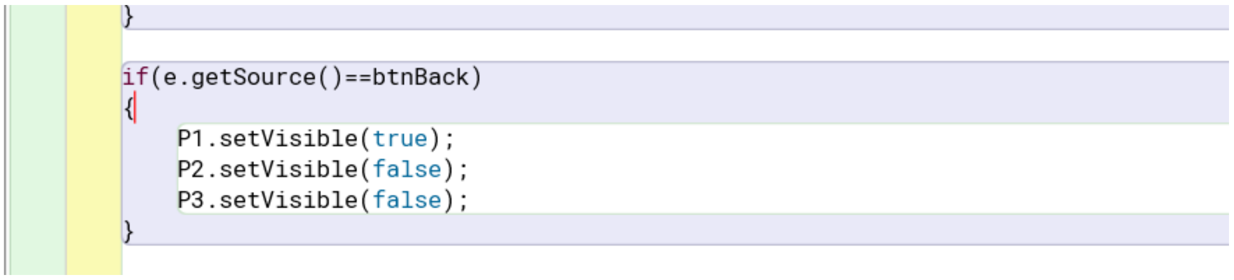
Registration Form

Academic Course Registration Form

CourseID:		Course Name:	
Duration:		Level:	
Credit:		Number Of Assessments:	
CourseID:		Course Leader:	
Lecturer Name:		Starting Date:	
Completion Date:			

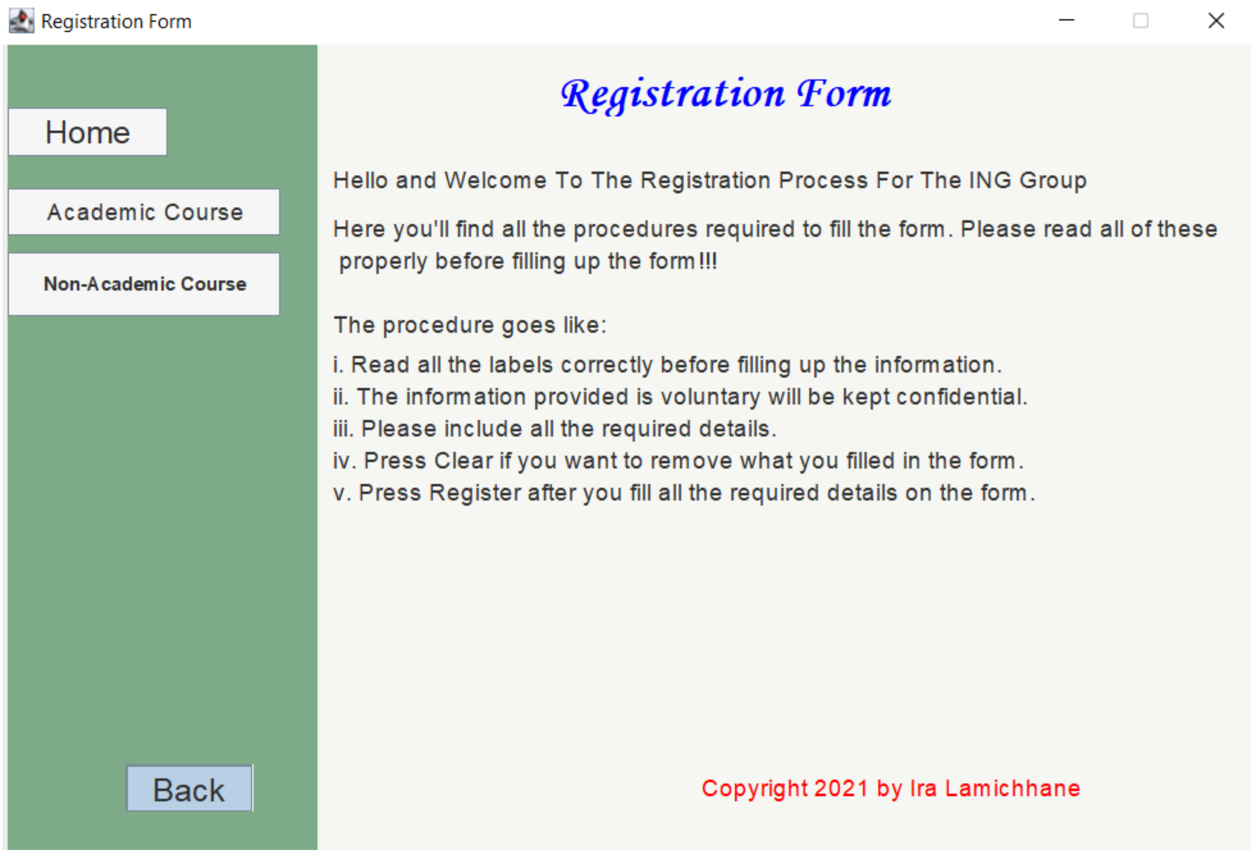
Figure 22: Logical error on GUI

Correction of Error:
I corrected the panel visibility then the error was corrected and the required panel was shown.



```
}

if(e.getSource()==btnBack)
{
    P1.setVisible(true);
    P2.setVisible(false);
    P3.setVisible(false);
}
```

Figure 23: Correction of Logical error on Code



Registration Form

Hello and Welcome To The Registration Process For The ING Group

Here you'll find all the procedures required to fill the form. Please read all of these properly before filling up the form!!!

The procedure goes like:

i. Read all the labels correctly before filling up the information.
ii. The information provided is voluntary will be kept confidential.
iii. Please include all the required details.
iv. Press Clear if you want to remove what you filled in the form.
v. Press Register after you fill all the required details on the form.

Copyright 2021 by Ira Lamichhane

Figure 24: Correction of Logical error on GUI

7. Conclusion:

Java gives us the real opportunity to write most of our programs in a secure language. It extends Java with a parametric polymorphical mechanism that allows generic abstractions to be defined and implemented. The paper gives the extensive language a full design. Computers and other electronic devices may be intuitively operated by direct manipulation of graphical icons such as buttons, scroll bars, windows, tabs, menus, cursors and the mouse pointing device thanks to graphical user interfaces. Interactive capabilities such as voice commands and touchscreens can be found in many modern graphical interfaces. When the Palo Alto research lab at Xerox came up with the graphical user interface in the late 1970s, it was a response to the inefficient usability of early text-based command-line interfaces for average users.

This was the Second coursework of Programming and to be honest it was not easy. This coursework has taught me how not to give up on things even if they get tangled hard. Programming sure is a hard subject to learn, we get many errors and a simple error can cause the program to not run. The GUI I made makes me feel satisfied to know I'm now capable of making good GUI which are user friendly and not hard to work with. This GUI is connected with the very first coursework that we submitted on our first semester. At that very time Java was very new to me and I was not that satisfied with my work, but this coursework was fun and I've now learned a lot about java and how things work. I can now proudly say I'm evolving within. This coursework has two registration form one is Academic Course Registration Form and the other is Non-Academic Course Registration Form. I have made a menu which contains buttons through which we can easily go into the respective registration form and a back button that leads our way back to the home panel.

Thanks to Binaya sir who guided me throughout the coursework. Having helpful teacher and friends made me solve most of my confusions. I learned about different methods and coding styles and also learned how to research in a proper way. Many methods like getter, setter, constructor, actionListner, etc. was used while making this coursework. Completing this coursework was like a rollercoaster ride, it included joy, fear and a lot of tears. Looking forward to making myself even better day by day.

8. Reference

Bennett, C. &. C. L., 2021. The Economic Times. [Online]

Available at: https://economictimes.indiatimes.com/definition/pseudocode [Accessed 12 8 2021]. Oracle, 2021. Oracle Java. [Online]

Available at: https://www.oracle.com/java/ [Accessed 18 8 2021].

ParaCoding, 2021. ParaCoding. [Online]

Available at: https://www.progracoding.com/logical-error/ [Accessed 15 08 2021].

Stakify, 2021. Top JAva errors. [Online]

Available at: https://stackify.com/top-java-software-errors/ [Accessed 13 8 2021].

Tutorialspoint, 2021. UML class diagaram. [Online]

Available at: https://www.tutorialspoint.com/uml/uml_class_diagram.htm [Accessed 18 8 2021].

9. Appendix

- **INGCollege Class**

```
 import javax.swing.*;
import java.awt.*;
import java.util.*;
import javax.swing.table.DefaultTableModel;
import java.awt.event.*;

public class INGCollege implements ActionListener
{
    JFrame frame, Displayframe ;

    JPanel P1,PMenu,P2,P3;

    JLabel
lblForm,lblHello,lblProcess,lblRead,lblX,lbli,lblii,lbliii,lbliv,lblv,lblCopy,lblStartingDat
e,lblCompletionDate,lblNumberofAssessments,lblLecturerName,lblCourseLeader,l
blCoursesID,lblLevel,lblCredit,lblCourseName,lblAcaDuration,

lblForm1,lblForm2,lblDuration,lblPrequisites,lblCourseID,lblNonCourseID,lblNonCo
urseName,lblNonCoursesID,lblNonCourseLeader,lblInstructorName,lblStartDate,lb
lNCompletionDate,lblExamDate;

    JTextField
txtCourseID,txtCourseName,txtAcaDuration,txtLevel,txtCompletionDate,txtLecturer
Name,txtStartingDate,txtCredit,txtCourseLeader,txtCoursesID,txtNumberofAssess
ment,txtNonCourseID,txtNCompletionDate

,txtNonCourseName,txtDuration,txtPrerequisites,txtNonCoursesID,txtNonCourseLe
ader,txtInstructorName,txtStartDate,txtExamDate;

    JButton
btnHome,btnAca,btnNonAca,btnBack,btnAcaADD,btnAcaRegister,btnAcaDisplay,b
tnAcaClear,btnNonAcaADD,btnNonAcaRegister,btnRemove,btnNonAcaDisplay,bt
nNonAcaClear;

    ArrayList<Courses>AcademicList = new ArrayList<Courses>();

    ArrayList<Courses>NonAcademicList = new ArrayList<Courses>();

    Courses obj1 ; Courses obj2;
    int x=0;
    int y=0;

    INGCollege(){
        //Frame - > Frame for the registration from
        frame = new JFrame ("Registration Form");
        frame.setSize (800, 550);
        frame.setVisible(true);
        frame.setLayout(null);
        frame.setResizable(false);

        //JPanel for the registration form's opening page
        P1 = new JPanel ();
        P1.setBounds(200,0,650,550);
        P1.setBackground(Color.decode("#F6F6F3"));
        P1.setLayout(null);
        P1.setVisible(true);
        frame.add(P1);

        // Welcome paragraph for Home page or pannel no 1

        lblForm = new JLabel ("Registration Form");
        lblForm.setBounds (150, 20, 500, 25);
        Font f1 = new Font ("Monotype Corsiva", Font.BOLD, 30);
        lblForm.setForeground(Color.BLUE);
        lblForm.setFont(f1);
        P1.add(lblForm);

        lblHello = new JLabel ("Hello and Welcome To The Registration Process For
The ING Group");
        lblHello.setBounds (10, 70, 5000, 30);
        Font Hell = new Font ("Arial", Font.PLAIN, 15);
        lblHello.setFont(Hell);
        P1.add(lblHello);

        lblProcess = new JLabel ("Here you'll find all the procedures required to fill
the form. Please read all of these ");
        lblProcess.setBounds (10, 100, 5000, 30);
        lblProcess.setFont(Hell);
        P1.add(lblProcess);

        lblRead = new JLabel (" properly before filling up the form!!!");
        lblRead.setBounds (10, 120, 5000, 30);
        lblRead.setFont(Hell);
        P1.add(lblRead);

        lblX = new JLabel ("The procedure goes like:  ");
        lblX.setBounds (10, 160, 5000, 30);
        lblX.setFont(Hell);
        P1.add(lblX);

        lbli = new JLabel ("i. Read all the labels correctly before filling up the
information.");
        lbli.setBounds (10, 185, 5000, 30);
        lbli.setFont(Hell);
        P1.add(lbli);

        lblii = new JLabel ("ii. The information provided is voluntary will be kept
confidential.");
        lblii.setBounds (10, 205, 5000, 30);
        lblii.setFont(Hell);
        P1.add(lblii);

        lbliii = new JLabel ("iii. Please include all the required details.");
        lbliii.setBounds (10, 225, 5000, 30);
        lbliii.setFont(Hell);
        P1.add(lbliii);

        lbliv = new JLabel ("iv. Press Clear if you want to remove what you filled in the
form.");
        lbliv.setBounds (10, 245, 5000, 30);
        lbliv.setFont(Hell);
        P1.add(lbliv);

        lblv = new JLabel ("v. Press Register after you fill all the required details on
the form.");
        lblv.setBounds (10, 265, 5000, 30);
        lblv.setFont(Hell);
        P1.add(lblv);

        lblCopy = new JLabel ("Copyright 2021 by Ira Lamichhane ");
        lblCopy.setBounds (240, 450, 5000, 30);
        lblCopy.setFont(Hell);
        lblCopy.setForeground(Color.RED);
        P1.add(lblCopy);
```

```
//JPanel for the menu bar
PMenu = new JPanel ();
PMenu.setBounds(0, 0, 200, 550);
PMenu.setBackground(Color.decode("#7eab87"));
PMenu.setLayout(null);
PMenu.setVisible(true);
frame.add(PMenu);

//BUttons for JPanel number 1

btnHome = new JButton ("Home");
btnHome.setFont (new Font ("Arial", Font.PLAIN, 20));
btnHome.setBackground(Color.decode("#F6F6F6"));
btnHome.setBounds (7, 40, 100, 30);
btnHome.setVisible(true);
btnHome.setFocusable (false);
PMenu.add(btnHome);

btnAca = new JButton ("Academic Course");
btnAca.setFont (new Font ("Arial", Font.PLAIN, 15));
btnAca.setBackground(Color.decode("#F6F6F6"));
btnAca.setBounds (7, 90, 170, 30);
btnAca.setFocusable (false);
btnAca.setVisible(true);
PMenu.add(btnAca);

btnNonAca = new JButton ("Non-Academic Course");
btnNonAca.setFont(new Font ("Arial",Font.BOLD,12));
btnNonAca.setBackground(Color.decode("#F6F6F6"));
btnNonAca.setBounds (7, 130, 170, 40);
btnNonAca.setFocusable (false);
btnNonAca.setVisible(true);
PMenu.add(btnNonAca);

btnBack = new JButton ("Back");
btnBack.setFont (new Font ("Arail", Font.PLAIN, 20));
btnBack.setBackground(Color.decode("#F6F6F6"));
btnBack.setBounds(80, 450, 80, 30 );
btnBack.setVisible(true);
btnBack.setFocusable (false);
PMenu.add(btnBack);
```

```
// JPannel for Academic course

P2 = new JPanel ();
P2.setBounds(200,0,650,550);
P2.setBackground(Color.decode("#e6fdff"));
P2.setLayout(null);
P2.setVisible(true);
frame.add(P2);

//JLabel for heading of ACA course
lblForm1 = new JLabel ("Academic Course Registration Form");
lblForm1.setBounds (20, 20, 500, 25);
Font ff = new Font ("Arial", Font.BOLD, 26);
lblForm1.setFont(ff);
P2.add(lblForm1);

// JPannel for Non Academic Course
P3 = new JPanel ();
P3.setBounds(200,0,650,550);
P3.setBackground(Color.decode("#e6fdff"));
P3.setLayout(null);
P3.setVisible(true);
frame.add(P3);

//JLabel for heading of NON ACA Course
lblForm2 = new JLabel ("Non-Academic Course Registration Form");
lblForm2.setBounds (20, 20, 550, 25);
Font fff = new Font ("Arial", Font.BOLD, 26);
lblForm2.setFont(fff);
P3.add(lblForm2);

// using JLabel component of swing package for Academic Course

//JLabel and TextField for CourseID
lblCourseID = new JLabel ("CourseID: ");
lblCourseID.setBounds (5, 70, 120, 30);
P2.add(lblCourseID);

txtCourseID = new JTextField();
txtCourseID.setBounds(110, 75, 120, 20);
P2.add(txtCourseID);
```

```
//JLabel and TextField for Course Name
lblCourseName = new JLabel ("Course Name:");
lblCourseName.setBounds (350, 70, 120, 30);
P2.add(lblCourseName);

txtCourseName = new JTextField();
txtCourseName.setBounds (450, 75, 120, 20);
P2.add(txtCourseName);

//JLabel and TextField for Duration

lblAcaDuration = new JLabel ("Duration: ");
lblAcaDuration.setBounds (5, 105, 120, 30 );
P2.add(lblAcaDuration);

txtAcaDuration = new JTextField ();
txtAcaDuration.setBounds (110, 110, 120, 20);
P2.add(txtAcaDuration);

//JLabel and TextField for Level
lblLevel = new JLabel ("Level: ");
lblLevel.setBounds (350, 105, 120, 30);
P2.add(lblLevel);

txtLevel = new JTextField ();
txtLevel.setBounds (450, 110, 120, 20);
P2.add(txtLevel);

//JLabel and TextField for Credit
lblCredit = new JLabel ("Credit: ");
lblCredit.setBounds (5, 140, 120, 30);
P2.add(lblCredit);

txtCredit = new JTextField ();
txtCredit.setBounds (110, 145, 120, 20);
P2.add(txtCredit);

//JLabel and TextField for Number of Assessments

lblNumberofAssessments = new JLabel ("Number Of Assessments: ");
lblNumberofAssessments.setBounds (290, 140, 150, 30);
P2.add(lblNumberofAssessments);
```

```
txtNumberofAssessment = new JTextField ();
txtNumberofAssessment.setBounds (450, 145, 120, 20);
P2.add(txtNumberofAssessment);

//ADD button for ACA form
btnAcaADD = new JButton ("Add");
btnAcaADD.setFont (new Font ("Arail", Font.PLAIN, 20));
btnAcaADD.setBackground(Color.decode("#edc591"));
btnAcaADD.setBounds(460, 170, 100, 30 );
btnAcaADD.setFocusable (false);
btnAcaADD.setVisible(true);
P2.add(btnAcaADD);

//JLabel and TextField for CourseID
lblCoursesID = new JLabel ("CourseID: ");
lblCoursesID.setBounds (5, 220, 120, 30);
P2.add(lblCoursesID);

txtCoursesID = new JTextField ();
txtCoursesID.setBounds (110, 225, 120, 20);
P2.add(txtCoursesID);

//JLabel and TextField for Course Leader
lblCourseLeader = new JLabel ("Course Leader: ");
lblCourseLeader.setBounds (320, 220, 120, 30);
P2.add(lblCourseLeader);

txtCourseLeader = new JTextField();
txtCourseLeader.setBounds (450, 225, 120, 20);
P2.add(txtCourseLeader);

//JLabel and TextField for Lecturer Name
lblLecturerName = new JLabel ("Lecturer Name: ");
lblLecturerName.setBounds (5, 260, 120, 30);
P2.add(lblLecturerName);

txtLecturerName = new JTextField ();
txtLecturerName.setBounds (110, 265, 120, 20);
P2.add(txtLecturerName);

//JLabel and TextField for Starting Date
lblStartingDate = new JLabel ("Starting Date: ");
lblStartingDate.setBounds (320, 260, 120, 30);
```

```
P2.add(lblStartingDate);

txtStartingDate = new JTextField ();
txtStartingDate.setBounds (450, 265, 120, 20);
P2.add(txtStartingDate);

//JLabel and TextField for Completion Date
lblCompletionDate = new JLabel ("Completion Date: ");
lblCompletionDate.setBounds (5, 300, 120, 30);
P2.add(lblCompletionDate);

txtCompletionDate = new JTextField ();
txtCompletionDate.setBounds (110, 305, 120, 20);
P2.add(txtCompletionDate);

// Register button for ACA Course
btnAcaRegister = new JButton ("Register");
btnAcaRegister.setFont (new Font ("Arail", Font.PLAIN, 17));
btnAcaRegister.setBackground(Color.decode("#edc591"));
btnAcaRegister.setBounds (460, 305, 100, 30);
btnAcaRegister.setVisible (true);
btnAcaRegister.setFocusable (false);
P2.add(btnAcaRegister);

//Display Button for ACA Course
btnAcaDisplay = new JButton ("Display");
btnAcaDisplay.setFont (new Font ("Arail", Font.PLAIN, 17));
btnAcaDisplay.setBackground(Color.decode("#edc591"));
btnAcaDisplay.setBounds (100, 400, 100, 30);
btnAcaDisplay.setFocusable (false);
btnAcaDisplay.setVisible (true);
P2.add(btnAcaDisplay);

//Clear buttoon for ACA Course
btnAcaClear = new JButton ("Clear");
btnAcaClear.setFont (new Font ("Arail", Font.PLAIN, 20));
btnAcaClear.setBackground(Color.decode("#edc591"));
btnAcaClear.setBounds (300, 400, 100, 30);
btnAcaClear.setFocusable (false);
btnAcaClear.setVisible (true);
P2.add(btnAcaClear);

// using JLabel component of swing package for Non-Academic Course
```

```
//JLabel and TextField for CourseID
lblNonCourseID = new JLabel ("CourseID: ");
lblNonCourseID.setBounds (5, 70, 120, 30);
P3.add(lblNonCourseID);

txtNonCourseID = new JTextField();
txtNonCourseID.setBounds(110, 75, 120, 20);
P3.add(txtNonCourseID);

//JLabel and TextField for Course Name
lblNonCourseName = new JLabel ("Course Name:");
lblNonCourseName.setBounds (350, 70, 120, 30);
P3.add(lblNonCourseName);

txtNonCourseName = new JTextField();
txtNonCourseName.setBounds (450, 75, 120, 20);
P3.add(txtNonCourseName);

//JLabel and TextField for Duration
lblDuration = new JLabel ("Duration: ");
lblDuration.setBounds (5, 105, 120, 30 );
P3.add(lblDuration);

txtDuration = new JTextField ();
txtDuration.setBounds (110, 110, 120, 20);
P3.add(txtDuration);

//JLabel and TextField for Prerequiste
lblPrequisites = new JLabel ("Prerequisites: ");
lblPrequisites.setBounds (350, 105, 120, 30);
P3.add(lblPrequisites);

txtPrerequisites = new JTextField ();
txtPrerequisites.setBounds (450, 110, 120, 20);
P3.add(txtPrerequisites);

// ADD button for NON ACA Course
btnNonAcaADD = new JButton ("Add");
btnNonAcaADD.setFont (new Font ("Arail", Font.PLAIN, 20));
btnNonAcaADD.setBackground(Color.decode("#edc591"));
btnNonAcaADD.setBounds(460, 145, 100, 30 );
btnNonAcaADD.setVisible(true);
btnNonAcaADD.setFocusable (false);
P3.add(btnNonAcaADD);

//JLabel and TextField for CorseID
lblNonCoursesID = new JLabel ("CourseID: ");
lblNonCoursesID.setBounds (5, 220, 120, 30);
P3.add(lblNonCoursesID);

txtNonCoursesID = new JTextField ();
txtNonCoursesID.setBounds (110, 225, 120, 20);
P3.add(txtNonCoursesID);

//JLabel and TextField for Course Leader
lblNonCourseLeader = new JLabel ("Course Leader: ");
lblNonCourseLeader.setBounds (350, 220, 120, 30);
P3.add(lblNonCourseLeader);

txtNonCourseLeader = new JTextField();
txtNonCourseLeader.setBounds (450, 225, 120, 20);
P3.add(txtNonCourseLeader);

//JLabel and TextField for Instructor Name
lblInstructorName = new JLabel ("Instructor Name: ");
lblInstructorName.setBounds (5, 255, 120, 30);
P3.add(lblInstructorName);

txtInstructorName = new JTextField ();
txtInstructorName.setBounds (110, 260, 120, 20);
P3.add(txtInstructorName);

//JLabel and TextField for Start Date
lblStartDate = new JLabel ("Start Date: ");
lblStartDate.setBounds (350, 255, 120, 30);
P3.add(lblStartDate);

txtStartDate = new JTextField ();
txtStartDate.setBounds (450, 260, 120, 20);
P3.add(txtStartDate);

//JLabel and TextField for Completion Date
lblNCompletionDate = new JLabel ("Completion Date: ");
lblNCompletionDate.setBounds (5, 290, 120, 30);
P3.add(lblNCompletionDate);
```

```
txtNCompletionDate = new JTextField ();
txtNCompletionDate.setBounds (110, 295, 120, 20);
P3.add(txtNCompletionDate);

//JLabel and TextField for Exam Date
lblExamDate = new JLabel ("Exam Date: ");
lblExamDate.setBounds (350, 290, 120, 30);
P3.add(lblExamDate);

txtExamDate = new JTextField ();
txtExamDate.setBounds (450, 295, 120, 20);
P3.add(txtExamDate);

//Buttons for NON-Academic Courses

// Register button for NON ACA Course
btnNonAcaRegister = new JButton ("Register");
btnNonAcaRegister.setFont (new Font ("Arail", Font.PLAIN, 17));
btnNonAcaRegister.setBackground(Color.decode("#edc591"));
btnNonAcaRegister.setBounds (460, 330, 100, 30);
btnNonAcaRegister.setVisible (true);
btnNonAcaRegister.setFocusable (false);
P3.add(btnNonAcaRegister);

//Remove button for NON ACA Course
btnRemove = new JButton ("Remove");
btnRemove.setFont (new Font ("Arail", Font.PLAIN, 17));
btnRemove.setBackground(Color.decode("#edc591"));
btnRemove.setBounds (100, 400, 100, 30);
btnRemove.setVisible (true);
btnRemove.setFocusable (false);
P3.add(btnRemove);

//Display button for NON ACA COurse
btnNonAcaDisplay = new JButton ("Display");
btnNonAcaDisplay.setFont (new Font ("Arail", Font.PLAIN, 17));
btnNonAcaDisplay.setBackground(Color.decode("#edc591"));
btnNonAcaDisplay.setBounds (250, 400, 100, 30);
btnNonAcaDisplay.setVisible (true);
btnNonAcaDisplay.setFocusable (false);
P3.add(btnNonAcaDisplay);
```

```
    //CLear Button for NON ACA Course
    btnNonAcaClear = new JButton ("Clear");
    btnNonAcaClear.setFont (new Font ("Arail", Font.PLAIN, 20));
    btnNonAcaClear.setBackground(Color.decode("#edc591"));
    btnNonAcaClear.setBounds (400, 400, 100, 30);
    btnNonAcaClear.setVisible (true);
    btnNonAcaClear.setFocusable (false);
    P3.add(btnNonAcaClear);

    // Adding buttons to action Listner

    btnAca.addActionListener(this);
    btnHome.addActionListener(this);
    btnNonAca.addActionListener(this);
    btnBack.addActionListener(this);

    btnAcaADD.addActionListener(this);
    btnAcaRegister.addActionListener(this);
    btnAcaDisplay.addActionListener(this);
    btnAcaClear.addActionListener(this);
    btnNonAcaADD.addActionListener(this);
    btnNonAcaRegister.addActionListener(this);
    btnRemove.addActionListener(this);
    btnNonAcaDisplay.addActionListener(this);
    btnNonAcaClear.addActionListener(this);

}


// Making the buttons work by switching panels
public void actionPerformed (ActionEvent e ) {
    if (e.getSource ()== btnAca)
    {
        P1.setVisible(false);
        P2.setVisible(true);
        P3.setVisible(false);
    }

    if(e.getSource()==btnHome)
    {
        P1.setVisible(true);
        P2.setVisible(false);
        P3.setVisible(false);
        }

        if(e.getSource()==btnNonAca)
        {
            P1.setVisible(false);
            P2.setVisible(false);
            P3.setVisible(true);
        }

        if(e.getSource()==btnBack)
        {
            P1.setVisible(true);
            P2.setVisible(false);
            P3.setVisible(false);
        }


        // Making the Add button work
        if(e.getSource()==btnAcaADD){
            //if button is pressed and fields are empty
            if
            (txtCourseID.getText().isEmpty()||
            txtCourseName.getText().isEmpty()||
            txtAcaDuration.getText().isEmpty()||
            txtLevel.getText().isEmpty()||
            txtCredit.getText().isEmpty()||
            txtNumberofAssessment.getText().isEmpty()
            )

            {
                JOptionPane.showMessageDialog(frame,"Empty Field
Founds","Alert!!",JOptionPane.WARNING_MESSAGE);
                return;
            }
            else{
                //If fields are not empty
                String AC_ID =txtCourseID.getText();
                String AC_CN = txtCourseName.getText();
                String AC_L = txtLevel.getText();
                String AC_C = txtCredit.getText();

                int AC_D;
                int AC_NOA;
                for(Courses c:AcademicList)
                {
                    if (c.getcourseID().equals(AC_ID))
                    {

                        JOptionPane.showMessageDialog(frame,"CourseID exists Please
        enter another ID","Alert!!",JOptionPane.WARNING_MESSAGE);
                        return;
                    }
                }
                try{
                    AC_D =Integer.parseInt(txtAcaDuration.getText());
                    AC_NOA= Integer.parseInt(txtNumberofAssessment.getText());
                }
                catch(Exception a){
                    JOptionPane.showMessageDialog(frame,"Please input Integer Value
        in duration and numberofAsessment.");
                    return;
                }
                obj1 = new AcademicCourse(AC_ID, AC_CN, AC_D, AC_L, AC_C,
        AC_NOA);
                AcademicList.add(obj1);
                JOptionPane.showMessageDialog(frame,"Course Successfully added");
            }
        }

        // Making the Register button work
        if(e.getSource()==btnAcaRegister){
            //if button is pressed ansd fields are empty
            if
            (txtCoursesID.getText().isEmpty()||
            txtCourseLeader.getText().isEmpty()||
            txtLecturerName.getText().isEmpty()||
            txtStartingDate.getText().isEmpty()||
            txtCompletionDate.getText().isEmpty()
            )

            {
                JOptionPane.showMessageDialog(frame,"Empty Field
        Founds","Alert!!",JOptionPane.WARNING_MESSAGE);
                return;
            }
```

```
          else{
             //If fields are not empty
             String AC_IDs =txtCoursesID.getText();
             String AC_CL = txtCourseLeader.getText();
             String AC_LN = txtLecturerName.getText();
             String AC_SD = txtStartingDate.getText();
             String AC_CD = txtCompletionDate.getText();

             for (int i = 0; i<AcademicList.size(); i++){
                if (AcademicList.get(i).getcourseID().equals(txtCoursesID.getText()))
                {
                   AcademicCourse AC = (AcademicCourse)(AcademicList.get(i));
                   x = x+1;

                   if (!AC.getIsRegistered()){
                      AC.Register(AC_CL, AC_LN, AC_SD, AC_CD);
                      JOptionPane.showMessageDialog(frame,"Course Successfully
Registered.");
                      return;
                   }

                   if (AC.getIsRegistered()){

                      JOptionPane.showMessageDialog(frame,"Course is already
registered!");
                      return;
                   }
                }

             }

             if(x==0){
                JOptionPane.showMessageDialog(frame,"CourseID doesn't exist");
                return;
             }

          }
       }

       //Making the clear button work
       if (e.getSource()==btnAcaClear){
          txtCourseID.setText("");
          txtCourseName.setText("");
```

```
            txtAcaDuration.setText("");
            txtLevel.setText("");
            txtCredit.setText("");
            txtNumberofAssessment.setText("");
            txtCoursesID.setText("");
            txtCourseLeader.setText("");
            txtLecturerName.setText("");
            txtStartingDate.setText("");
            txtCompletionDate.setText("");

        }


        //For Non Academic Course

        // Making the Add button work
        if(e.getSource()==btnNonAcaADD){
            //if button is pressed and fields are empty
            if
            (txtNonCourseID.getText().isEmpty()||
            txtNonCourseName.getText().isEmpty()||
            txtDuration.getText().isEmpty()||
            txtPrerequisites.getText().isEmpty()
            )

            {
                JOptionPane.showMessageDialog(frame,"Empty Field
    Founds","Alert!!",JOptionPane.WARNING_MESSAGE);
                return;
            }
            else{
                //If fields are not empty
                String NAC_ID =txtNonCourseID.getText();
                String NAC_CN = txtNonCourseName.getText();
                String NAC_P = txtPrerequisites.getText();
                int NAC_D;


                for(Courses c:NonAcademicList)
                {
                    if (c.getcourseID().equals(NAC_ID))
                    {
```

```
                    JOptionPane.showMessageDialog(frame,"CourseID exists Please
enter another ID","Alert!!",JOptionPane.WARNING_MESSAGE);
                    return;
                 }
              }
              try{
                 NAC_D =Integer.parseInt(txtDuration.getText());
              }
              catch(Exception b){
                 JOptionPane.showMessageDialog(frame,"Please Input Integer Value
in Duration! ");
                 return;
              }
              obj2 = new NonAcademicCourse(NAC_ID, NAC_CN, NAC_D, NAC_P );
              NonAcademicList.add(obj2);
              JOptionPane.showMessageDialog(frame,"Course is successfully
added");
           }
        }

        //Making the Display button work

        if(e.getSource()==btnAcaDisplay){
           {
              DefaultTableModel tableModelACA = new DefaultTableModel();
              JTable table = new JTable(tableModelACA);

              //adding columns
              String
column[]={"CourseID","CourseName","Duration","Level","Credit","NumberOfAsses
sment","CourseLeader","LecturerName","StartingDate","CompletionDate"};
              for(int i=0;i<10;i++)
                 tableModelACA.addColumn(column[i]);
              for(int i=0; i<AcademicList.size();i++)
              {
                 AcademicCourse AC = (AcademicCourse)(AcademicList.get(i));
                 String S1 =String.valueOf(AC.getcourseduration());
                 String S2 =String.valueOf(AC.getNumberofAssessments());

                 String
S3[]={AC.getcourseID(),AC.getcoursename(),S1,AC.getLevel(),AC.getCredit(),S2,
AC.getcourseleader(),AC.getLecturername(),AC.getStartingDate(),AC.getCompleti
onDate()};
```

```
                tableModelACA.insertRow(tableModelACA.getRowCount(), S3);
            }

            Displayframe = new JFrame();
            Displayframe.setSize(800, 800);
            Displayframe.add(new JScrollPane(table));
            Displayframe.setVisible(true);
        }
    }


    // Making the Register Button work

    if(e.getSource()==btnNonAcaRegister){
        //if button is pressed ansd fields are empty
        if
        (txtNonCoursesID.getText().isEmpty()||
        txtNonCourseLeader.getText().isEmpty()||
        txtInstructorName.getText().isEmpty()||
        txtStartDate.getText().isEmpty()||
        txtNCompletionDate.getText().isEmpty()||
        txtExamDate.getText().isEmpty()
        )

        {
            JOptionPane.showMessageDialog(frame,"Empty Field
Founds","Alert!!",JOptionPane.WARNING_MESSAGE);
            return;
        }
        else{
            //If fields are not empty
            String NAC_IDs =txtNonCoursesID.getText();
            String NAC_CL = txtNonCourseLeader.getText();
            String NAC_IN = txtInstructorName.getText();
            String NAC_SD = txtStartDate.getText();
            String NAC_CD = txtCompletionDate.getText();
            String NAC_ED = txtExamDate.getText();

            for (int i = 0; i<NonAcademicList.size(); i++){
                if
(NonAcademicList.get(i).getcourseID().equals(txtNonCourseID.getText()))
                {
                        NonAcademicCourse NAC =
(NonAcademicCourse)(NonAcademicList.get(i));
                        y = y+1;

                        if (!NAC.getisRegistered()){
                            NAC.register(NAC_CL, NAC_IN, NAC_SD, NAC_CD, NAC_ED);
                            JOptionPane.showMessageDialog(frame,"Course Successfully
Registered.");
                            return;
                        }

                        if (NAC.getisRegistered()){

                            JOptionPane.showMessageDialog(frame,"Course is already
Registered!");
                            return;
                        }
                    }

                }

                if(y==0){
                    JOptionPane.showMessageDialog(frame,"CourseID doesn't exist");
                    return;
                }

            }
        }

        // Making the Remove button work

        if(e.getSource()==btnRemove){
            String R=JOptionPane.showInputDialog(frame,"Enter
CourseID","Remove",JOptionPane.QUESTION_MESSAGE);
            for(int i=0; i<NonAcademicList.size();i++){
                if(NonAcademicList.get(i).getcourseID().equals(R))
                {
                    NonAcademicCourse NAC =
(NonAcademicCourse)(NonAcademicList.get(i));
                    if(!NAC.getisRemoved())
                    {
                        NAC.remove();
                        JOptionPane.showMessageDialog(frame,"Course is Removed");
                        return;
                    }
                }
                else
                {
                    JOptionPane.showMessageDialog(frame,"CourseID doesn't
Exists","!!!",JOptionPane.WARNING_MESSAGE);
                    return;
                }
            }
        }

        //Making the Display button work
        if(e.getSource()==btnNonAcaDisplay){
            {
                DefaultTableModel tableModelNAC = new DefaultTableModel();
                JTable table = new JTable(tableModelNAC);

                //adding columns
                String
column[]={"CourseID","CourseName","Duration","Prerequisite","CourseLeader","In
structorName","StartingDate","CompletionDate","ExamDate"};
                for(int i=0;i<9;i++)
                    tableModelNAC.addColumn(column[i]);
                for(int i=0; i<NonAcademicList.size();i++)
                {
                    NonAcademicCourse NAC =
(NonAcademicCourse)(NonAcademicList.get(i));
                    String S4 =String.valueOf(NAC.getcourseduration());

                    String
S5[]={NAC.getcourseID(),NAC.getcoursename(),S4,NAC.getprerequisite(),NAC.ge
tcourseleader(),NAC.getInstructorname(),NAC.getStartdate(),NAC.getCompletiond
ate(), NAC.getExamdate()};
                    tableModelNAC.insertRow(tableModelNAC.getRowCount(), S5);
                }

                Displayframe = new JFrame();
                Displayframe.setSize(800, 800);
                Displayframe.add(new JScrollPane(table));
                Displayframe.setVisible(true);
            }
        }
```

```
        // Making the Clear buttons work
        if(e.getSource()==btnNonAcaClear){
            txtNonCourseID.setText("");
            txtNonCourseName.setText("");
            txtDuration.setText("");
            txtPrerequisites.setText("");
            txtNonCoursesID.setText("");
            txtNonCourseLeader.setText("");
            txtInstructorName.setText("");
            txtStartDate.setText("");
            txtNCompletionDate.setText("");
            txtExamDate.setText("");
        }




    }

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

    }

}
```

Course Class

```
public class Courses
  {
    // Attributes of class Courses//
  private String courseID;
  private String coursename;
  public String courseleader;
  private int courseduration;

  // Constructors//
      public Courses ( String courseID, String coursename, int courseduration)
  {
```

```
  this.courseID = courseID;
  this.coursename = coursename;
  this.courseduration = courseduration;
  this.courseleader = ""; // Here, setting course_leader as empty//
}


// Getter for courseID//
public String getcourseID ()
{
   return this.courseID;
}

// Getter for course_name//
public String getcoursename ()
{
   return this.coursename;
}

// Getter for course_duration//
public int getcourseduration ()
{
   return this.courseduration;
}


// Setter for course_duration//
public void setcourseleader (String course_leader)
{
   this.courseleader = courseleader;
}

//Getter for courseLeader//
public String getcourseleader()
{
   return courseleader;
  }

// Method of displaying data //
public void display()
{
  // Displaying the Header//
  System.out.println("\t\t COURSES DETAIL");
```

```
      System.out.println( "Course ID=" + courseID);
      System.out.println( "Course name=" + coursename);

      // Display course_leader if it's not empty//
      if (this.courseleader!="")
      {
         System.out.println( "Course leader=" + courseleader);
        }

   }
 }
```

Academic Course Class

```
// AcademicCourse Class is sub-class of Class Courses//
public class AcademicCourse extends Courses
 {
   // Attributes of class AcademicCourse//
   private String Lecturername;
   private String Level;
   private String Credit;
   private String StartingDate;
   private String CompletionDate;
   private int NumberofAssessments;
   private boolean isRegistered;

   //Constructors//
   public AcademicCourse( String courseID, String coursename, int courseduration,
String Level, String Credit, int NumberofAssessments)
    {
      super(courseID, coursename, courseduration); // Calling parent class
constructor//
      this.Level = Level;
      this.Credit = Credit;
      this.NumberofAssessments = NumberofAssessments;
      // Setting Lecturer_name as empty//
      this.Lecturername = "";
      // Setting Starting_Date as empty//
      this.StartingDate= "";
      // Setting Completion_Date as empty//
      this.CompletionDate= "";
      // Setting isRegistered as false//
      this.isRegistered= false;
   }

    //Accessor Methods//

    public boolean getIsRegistered()
    {
       return isRegistered;
    }

    //Getter for Level//
     public String getLevel()
    {
       return Level;
    }

    //Getter for Credit//
    public String getCredit()
    {
       return Credit;
    }

    public String getStartingDate()
    {
       return StartingDate;
    }

    public String getCompletionDate()
    {
       return CompletionDate;
    }


    // Getter for Number_of_Assessments//
    public int getNumberofAssessments ()
    {
       return NumberofAssessments;
    }

    // Setter for Lecturer_name//
    public void setLecturername( String Lecturername)
    {
       this.Lecturername = Lecturername;
    }
```

```
    //Getter for LecturerName//
    public String getLecturername()
    {
        return Lecturername;
    }

    // Setter for Number_of_Assessments//
    public void setNumberofAssessments(int NumberofAssessments)
    {
        this.NumberofAssessments = NumberofAssessments;
    }

    // New method Register//
    public void Register( String Lecturername, String courseleader, String
StartingDate, String CompletionDate)
    {
        // Display if AcademicCourse is Registered//
        if (this.isRegistered == true)
        {
        System.out.println( "Academic Course has been registered.");
        System.out.println("The Course Leader's name is:" +super.courseleader);
        System.out.println( "The name of Lecturer is:" +this.Lecturername);
        System.out.println( "The Starting date is:" +this.StartingDate);
        System.out.println("The Completion date is:" +this.CompletionDate);
        }

        // Updating Vales//
       else
       {
        super.courseleader=courseleader;// Calling course_leader from parent class//
        this.Lecturername=Lecturername;
        this.StartingDate=StartingDate;
        this.CompletionDate=CompletionDate;
        this.isRegistered=true;
       }
    }

    // Method of Displaying Data//
    public void display()
    {
        super.display(); // Calling parent class for displaying data//
        if (isRegistered == true)
        {
        // Display Header//
        System.out.println("\n\t\t ACADEMIC COURSE DETAIL");
        // Display Lecturer_name//
        System.out.println("The Lecturer's name is :" +this.Lecturername);
        // Display Level//
        System.out.println("The Level is :" +this.Level);
        // Display Credit//
        System.out.println("The Credit is :" +Credit);
        // Display Starting_Date//
        System.out.println("The Starting Date is :" +this.StartingDate);
        // Display Completion_Date//
        System.out.println("The Completion Date is :" +this.CompletionDate);
        // Display Number_of_Assessments//
        System.out.println("The Number of Assessments is:"
+this.NumberofAssessments);
        }
    }

}
```

- **Non-Academic Course Class**

```
 // NonAcademicCourse Class is Sub-class of class Course//
public class NonAcademicCourse extends Courses
{
    // Attributes of NonAcademicCourse Class//
    private String Instructorname;
    private String Startdate;
    private String Completiondate;
    private String Examdate;
    private String prerequisite;
    private boolean isRegistered;
    private boolean isRemoved;

    public NonAcademicCourse(String courseID, String coursename, int
courseduration, String prerequisite)
    {
        super(courseID, coursename, courseduration);// Calling parent class
constructor//
        //Setting Start_date as empty//
        this.Startdate = "";
        // Setting Completion_date as empty//
        this.Completiondate = "";
        // Setting Exam_date as empty//
        this.Examdate = "";
        // Setting isRegistered as false//
        this.isRegistered= false;
        // Setting isRemoved as false//
        this.isRemoved= false;
        Instructorname = Instructorname;

        this.prerequisite = prerequisite;
    }

    //Accessor method//

    // Getter for Start_date//

    public String getStartdate()
    {
        return this.Startdate;
    }

    // Getter for Completion_date//
    public String getCompletiondate()
    {
        return this.Completiondate;
    }

    // Getter for Exam_date//
    public String getExamdate()
    {
        return this.Examdate;
    }

    // Getter for prerequisite//
    public String getprerequisite()
    {
        return this.prerequisite;
    }

    //Getter for isRegistered//
    public boolean getisRegistered()
    {
        return this.isRegistered;
    }
```

```
    //Getter for isRemoved//
    public boolean getisRemoved()
    {
        return this.isRemoved;
    }

    public String getInstructorname()
    {
        return Instructorname;
    }

    //Setter for Instructor_name//
    public void setInstructorname(String Instructorname)
    {
        if(isRegistered == false)
        {
            this.Instructorname=Instructorname;
             System.out.println("Ihe updated Instructor name is: " +this.Instructorname);
        }

        else
        {
           System.out.println("The Instructor's name cannot be changed.");
        }
    }

    // New method Register//
    public void register(String courseleader, String Instructorname, String Startdate,
String Examdate, String Completiondate)
    {
        if(isRegistered == false)
        {
            super.courseleader = courseleader;
            this.Instructorname=Instructorname;
            this.Startdate = Startdate;
            this.Completiondate = Completiondate;
            this.Examdate = Examdate;
            this.isRegistered = true;
        }

        else
        {
            System.out.println("Course is already Registered.");
        }
    }

    // Method for Removal//
    public void remove()
    {
        if(isRemoved == true)
        {
            System.out.println("Course is already Removed.");
        }

        else
        {
            super.setcourseleader ("");
            this.Instructorname = "";
            this.Startdate = "";
            this.Completiondate = "";
            this.Examdate = "";
            //Setting isRegistered false//
            this.isRegistered = false;
            // Setting idRemoved true//
            this.isRemoved = true;
        }
    }

    // Displaying Method//
    public void display ()
    {
        super.display ();

        // Display if isRegistered is true//
        if(isRegistered == true)
        {
            System.out.println("\n\t\t NON-ACADEMIC COURSE DEATIL");
            System.out.println("Instructor Name:" +Instructorname);
            System.out.println("Starting Date:" +Startdate);
            System.out.println("Completion Date:" +Completiondate);
            System.out.println("Exam Date:" +Examdate);
        }
    }
}
```