LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CS4001NI Programming

COURSEWORK-2

Assessment Weightage & Type

30% Individual Coursework

Semester and Year

Spring 2021

Student Name: Roshan Gautam

Group: N2

London Met ID: 20049215

College ID: NP01NT4S210047

Assignment Due Date: 20th August, 2021

Assignment Submission Date: 20th August, 2021

I confirm that I understand my coursework needs to be submitted online via Google classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submission will be treated as non-submission and a mark of zero will be awarded.

Table of Content

List of Tables

List of Figures

1. Introduction

Java is a high-level, general-purpose, class-based, object-oriented programming language. It is most used programming language for developing Java applications in laptops, data centers, game consoles, cell phones, etc. it is one of the fast, secure, and reliable programming languages preferred by many organizations for building their projects. Once a program is written, java translates the codes into binary codes that computers can understand and execute. (Guru 99 , 2021)

This is the second coursework of Programming Module. The task carried out during the development of coursework is based on with reference to first course work. The aim of this assignment is to develop a graphical user interface (GUI) for a system that stores details of Course that includes both academic and non-academic course. A new class called INGCollege is created. GUI consists of Text Fields which accepts some parameters and stores the value in the Array List when Add button or Register button is pressed. The display method is called from Academic Course Class and Non-Academic Course Class when display button is pressed respectively.

The GUI Text Field accepts the parameter which are to be assigned. getText() method is used for accepting the assigned parameter. Tools that was proven to be useful during the development of Course work is Bluej, MS-Word, and Draw.io. The programming code was completely written in Bluej, whereas the report was prepared in MS-Word and Draw.io was used for development of figures and tables too.

2. Class Diagram

2.1. Introduction to Class Diagram

Class diagram is a static diagram that represents the static view of an application with can also be used for constructing executable code of the software application. The purpose of class diagram is used for analysis and design of static view of an application, describe responsibilities of a system, and forward and reverse engineering. Class diagram empower us to show programming in a significant degree of planning and without taking a glimpse at the source code.

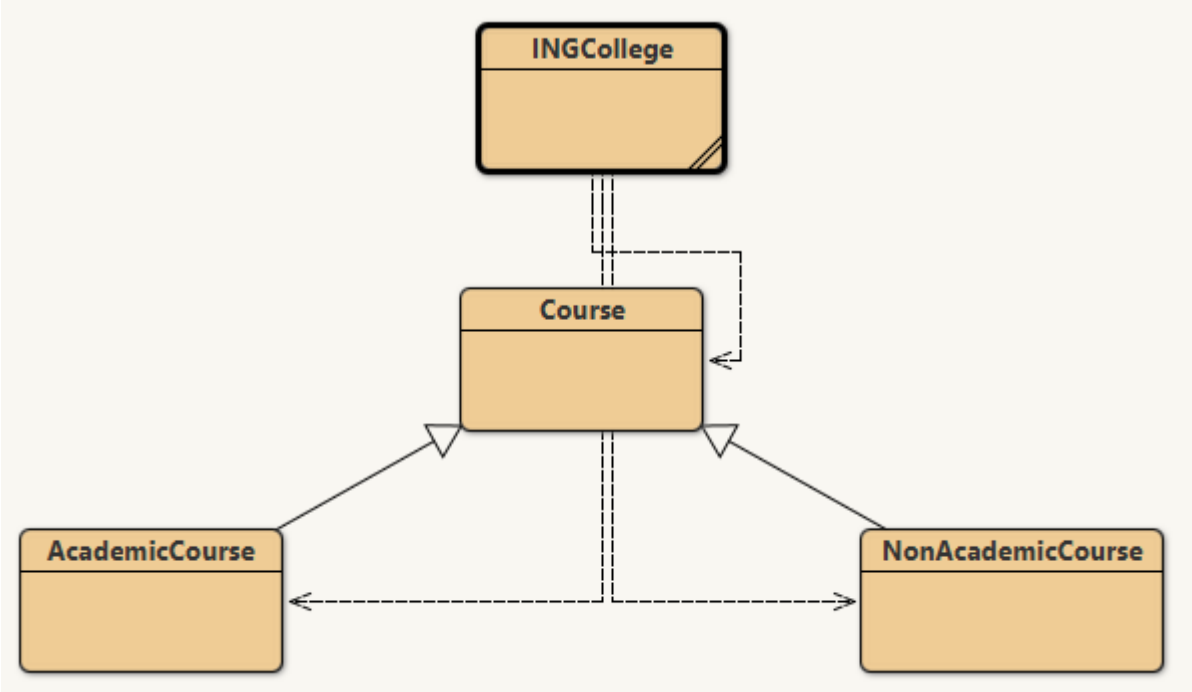

Figure 2-1 : Class Diagram of Classes

2.2. INGCollege Class Diagram

-	INGCollege			
academicCoure : JMenuItem	ff: Font	lblAcompletionDate2 : JLabel	lblNAcourseID2 : JLabel	txtAcredit : JTextField
btnAadd : JButton	file : JMenu	lblAcourseID : JLabel	lblNAcourseLeader : JLabel	txtAduration : JTextField
btnAclear : JButton	fr : Font	lblAcourseID2 : JLabel	lblNAcourseName : JLabel	txtAlecturerName : JTextField
btnAdisplay : JButton	ft: Font	lblAcourseLeader : JLabel	lblNAduration : JLabel	txtAlevel : JTextField
btnAregister : JButton	homeAcademicCourse : JLabel	lblAcourseName : JLabel	lblNAexamDate : JLabel	txtAnumOfAssessment : JTextField
btnHacademicCourse : JButton	homeNonAcademicCourse : JLabel	lblAcourseName : JLabel	lblNAinstructorName : JLabel	txtAstartingDate : JTextField
btnHexit : JButton	homeSelect : JLabel	lblAcredit : JLabel	lblNAinstructorName2 : JLabel	txtNAcompletionDate : JTextField
btnHhelp : JButton	homeWelcome : JLabel	lblAduration : JLabel	lblNAprerequisite : JLabel	txtNAcourseID : JTextField
btnHnonAcademicCourse : JButton	jAcademicCourse : JPanel	lblAlecturerName : JLabel	lblNAstartDate : Jlabel	txtNAcourseID2 : JTextField
btnNAadd : JButton	jf : JFrame	lblAlevel : JLabel	lblNonAcademicCourse : JLabel	txtNAcourseLeader : JTextField
btnNAclear : JButton	jfResult : JFrame	lblAnumOfAssessment : JLabel	nonAcademicCourse : JMenuItem	txtNAcourseName : JTextField
btnNAdisplay : JButton	jmb : JMenuBar	lblAnumOfAssessment2 : JLabel	txtAcompletionDate : JTextField	txtNAduration : JTextField
btnNAregister : JButton	jNonAcademicCourse : JPanel	lblAstartingDate : JLabel	txtAcourseID : JTextField	txtNAexamDate : JTextField
btnNAremove : JButton	jWelcometoCourse : JPanel	lblNAcompletionDate : JLabel	txtAcourseID2 : JTextField	txtNAinstructorName : JTextFleld
exitApp : JMenuItem	lblAcademicCourse : JLabel	lblNAcompletionDate2 : JLabel	txtAcouseLeader : JTextField	txtNAprerequisite : JTextField
fb : Font	lblAcompletionDate : JLabel	lblNAcourseID : JLabel	txtAcourseName : JTextField	txtNAstartDate : JTextField

ArrayList <Course> academicCoruse = new ArrrayList <Course>()
ArrayList <Course> nonAcademicCourse = new ArrayList (Course>()

Table 2-1 : INGCollege Class Diagram

3. Pseudocode

3.1. Introduction

Pseudocode is a method of programming that doesn't need any programming language sentence. Pseudocode is not a genuine programming language and cannot be arranged into an executable program. It utilizes basic English language grammar to compose code for programs before it is changed over into a particular programming language.

Benefits of pseudocode:

- Pseudocode makes programming easy to understand for programmers.
- It empowers the developers to focus on algorithm part of code development.

(The Economic Times, n.d)

3.2. INGCollege Class Pseudocode

IMPORT packages in program
CREATE class INGCollege
    DEFINE UI components
        CREATE frame
        CREATE panel
        CREATE Menu
        CREATE Label, Text Field and Button for Academic and Non-Academic Course
    CREATE constructor for class INGCollege
        DEFINE frame for Course Registration
            SET Close Operation on Exit
        DEFINE frame for Display
        DEFINE font for Label, Text Field and Button
        DEFINE Menu Bar

```
        SET Menu Bar in frame
        DEFINE Menu
        SET Menu in Menu Bar
        DEFINE Menu Item
        SET Menu Item in Menu with Separators
DEFINE panel for Welcome to Course
        DEFINE jWelcometoCourse components
DEFINE panel for Academic Course
        DEFINE jAcademicCourse components
        DEFINE ArrayList  for  AcademicCourseList
        DEFINE actionperformed method for academic add button
                INITIALIZE courseID, courseName, level and credit
                as String type
                INITIALIZE duration and numOfAssessment as int
                type
                INITIALIZE and SET value of checkInt as false
                INITIALIZE and SET value of check as false
                SET duration and numOfAssessment as 0
                IF ( is textFied empty ? )
                        YES : show data required message
                END IF
                ELSE IF
                        GET all values from TextField
                        TRY
                                CONVERT duration and
                                numOfAssessment to number
                                SET value of checkInt
                        END TRY
                        CATCH
                                DISPLAY error message
                        END CATCH
```

```
                    FOR (each data in AcademicCourseList)
                        IF (data.getCourseID() == courseID)
                            SET value of check
                            BREAK loop
                        END IF
                    END FOR
                    IF (check value of check and checkInt)
                        PASS value of TextField to
                            AcademicCourse Class
                        DISPLAY Course successfully added
                    END IF
                    ELSE IF (check value of checkInt)
                        DISPLAY course is already added
                    END IF
DEFINE actionperformed method for academic register
button
    INITIALIZE  instance variable as String type
    INITIALIZE and SET value of isRegistered variable as
        boolean type
    IF (text field is empty ?)
        DISPLAY required all TextField
    END IF
    ELSE IF
        GET all value of TextField
        INITIALIZE and SET value of variable i
        FOR ( each data in AcademicCourseList)
            IF (check courseID for matching )
                SET acGet as Object Casting for
                AcademicCourse
                SET value of isRegistered
```

```
                    IF (acGet.getIsRegistered() ==
                    false)
                        PASS value of TextField
                        to Register Method
                        DISPLAY course
                        successfully registered
                    END IF
                    ELSE IF
                        DISPLAY course already
                        registered
                    END IF
                END IF
                SET value of i to increase by 1
            END FOR
            IF (check isRegistered is false)
                DISPLAY course not registered yet
            END IF
        END IF
    DEFINE actionperformed method for academic display
    button
        IF (AcademicCourseList is empty)
            DSIPLAY No course added yet
        END IF
        ELSE IF
            INITIALIZE and set value of variable i
            FOR (each data in AcademicCourseList)
                SET acGet as object casting for
                AcademicCourse
                CALL display method of
                AcademicCoures
                SET value of i to increase by 1
```

```
                    END FOR
                END IF
            DEFINE actionperformed method for academic clear button
                SET value of all TextField using setText() as (“ ”)
            ADD jAcademicCourse to frame
        DEFINE panel for Non-Academic Course
            DEFINE Non-Academic Course components
            DEFINE ArrayList for NonAcademicCourseList
            DEFINE actionperformed method for nonAcademic add
            button
                INITIALIZE  instance variable courseID, courseName,
                prerequisite as String type
                INITIALIZE duration as int type
                INITIALIZE and SET value of checkInt and check
                SET value of duration
                IF (TextField is empty )
                    DISPLAY all TextField required
                END IF
                ELSE IF
                    GET all value of TextField
                    TRY
                        CONVERT duration to number
                        SET value of checkInt

                    END TRY
                    CATCH
                        DISPLAY error message
                    END CATCH
                    FOR (each data in NonAcademicCourseList)
                        IF ( check courseID for matching)
                            SET value of check
```

```
                    BREAK loop
                END IF
            END FOR
            IF (check values of check and checkInt)
                PASS value of TextField to
                NonAcademicCourse Class
                DISPLAY course successfully added
            END IF
            ELSE IF (check value of checkInt)
                DISPLAY course already added
            END IF
        END IF
    DEFINE actionperformed method for nonAcademic register
    button
        INITIALIZE instance variable as String type
        INITIALIZE and SET value of isRegistered variable as
        boolean type
        IF (text field is empty ?)
            DISPLAY required all TextField
        END IF
        ELSE IF
            GET all value of TextField
            INITIALIZE and SET value of variable i
            FOR ( each data in NonAcademicCourseList)
                IF (check courseID for matching )
                    SET nacGet as Object Casting
                    for NonAcademicCourse
                    SET value of isRegistered
                    IF (acGet.getIsRegistered() ==
                    false)
```

```
                    PASS value of TextField
                    to register Method
                    DISPLAY course
                    successfully registered
                END IF
                ELSE IF
                    DISPLAY course already
                    registered
                END IF
            END IF
            SET value of i to increase by 1
        END FOR
        IF (check isRegistered is false)
            DISPLAY course not registered yet
        END IF
    END IF
DEFINE actionperformed for nonAcademic remove button
    IF (TextField is Empty ?)
        DISPLAY TextField required
    END IF
    ELSE IF
        DISPLAY do you want to remove ?
        IF ( remove is YES )
            GET value of TextField
            SET iterator as itr for
            NonAcademicCourseList
            INITIALIZE and set value of
            recordFound
            WHILE (itr has next data)
                SET value to Course c
```

```
                    IF ( check if value of courseID
                    matches)
                        SET value of recordFound
                        SET nac as object casting
                    for NonAcademicCourse class
                        IF (nac.getIsRemoved ==
                        false)
                            CALL remove
                            method from
                            NonAcademicCours
                            e
                        END IF
                        CALL remove method or
                        iterator
                        DISPLAY course removed
                        successfully
                    END IF
                END WHILE
                IF (check recordFound )
                    DISPLAY course not found
                END IF
            END IF
        END IF
    DEFINE actionperformed method for non-academic display
    button
        INITIALIZE variable prerequisite as String type
        IF (NonAcademicCourseList is empty)
            DISPLAY No course added yet
        END IF
        ELSE IF
            INITIALIZE and set value of variable i
```

```
                    FOR (each data in NonAcademicCourseList)
                        SET acGet as object casting for
                        NonAcademicCourse
                        CALL display method of
                        NonAcademicCoures
                        PRINT prerequisite value
                        SET value of i to increase by 1
                    END FOR
            END IF
        DEFINE actionperformed method for academic clear button
            SET value of all TextField using setText() as (“ ”)
        ADD jNonAcademicCourse to frame
        DEFINE actionperformed method for JavaMenu exit button
            DISPLAY do you want to exit
            IF (exit option is yes)
                EXIT system
            END IF
        DEFINE actionperformed method for JavaMenu
        nonAcademicCourse button
            SET value of NonAcademicCourse visible
            SET value of AcademicCourse visible
        DEFINE actionperformed method for JavaMenu
        AcademicCourse button
            SET value of AcademicCourse visible
            SET value of NonAcademicCourse visible
        DEFINE actionperformed method for WelcometoCourse
        AcademicCourse button
            SET value of AcademicCourse visible
            SET value of WelcometoCourse visible
        DEFINE actionperformed method for WelcometoCourse
        NonAcademicCourse button
```

SET value of NonAcademicCourse visible

SET value of WelcometoCourse visible

DEFINE actionperformed method for WelcometoCourse exit button

DISPLAY do you want to exit

IF (option is Yes)

EXIT system

END IF

DEFINE actionperformed method for WelcometoCourse help button

DSIPLAY help option in case of any inquery

CREATE main class

INITIALIZE and call INGCollege constructor

4. Method Description

Method Description is the explanation of each method in INGCollege class. Different methods used in INGCollege class are :

- INGCollege() : This is a constructor which consists of all the codes that is used for making Graphical User Interface(GUI).
- btnAadd.addActionListener() : This is an action listener which adds the course in Academic Course when add button is pressed.
- btnAregister.addActionListener() : This is an action listener which registers the course in Academic Course after the course has been added successfully register button is pressed.
- btnAdisplay.addActionListener() : This is an action listener which displays the course details when display button is pressed.
- btnAclear.addActionListener() : This is an action listener which clears all the text field in GUI when clear button is pressed.
- btnNAadd.addActionListener: This is an action listener which adds the course detail in Non-Academic Course when add button is pressed.

- btnNAregister.addActionListener(): This is an action listener which registers the course detail in Non-Academic Course when register button is pressed.
- btnNAdisplay.addActionListener() : This is an action listener which displays the course details when display button is pressed.
- btnNAclear.addActionListener() : This is an action listener which clears all the TextField in GUI when clear button is pressed.
- btnNAremove.addActionListener() : This is an action listener which allows users to remove a course from non-academic course when courseID is given for removal.
- exitApp.addActionListener() : This is an action listener which exits the GUI when Exit button is pressed from JavaMenu.
- nonAcademicCourse.addActionListener() : This is an action listener which is used for changing panel between academic course and non-academic course from JavaMenu.
- academicCourse.addActionListener() : This is an action listener which is used for changing panel between academic couse and non-academic course from JavaMenu.
- btnHacademicCourse.addActionListener() : This is an action listener which allows user to go to academic course panel when this button is pressed.
- btnHnonAcademicCourse.addActionListener() : This is an action listener which allows user to go to non-academic course panel when this button is pressed.
- btnHexit.addActionListener() : This is an action listener which allows user to exit the GUI when the Exit button is pressd.
- btnHhelp.addActionListener() : This is an action listener which allows user to view help display for inquiry when help button is pressed.
- Void main() : This is the main class which is called for running the program and constructor INGCollege() is called when executed.

5. Testing

5.1. Test 1

Test Number	1
Objective	To compile and run the program in command prompt
Action	INGCollege.java file is compiled. And INGCollege class file is run in command prompt
Expected Result	To open the GUI made in INGCollege Class
Actual Result	The program is successfully compiled, and GUI was opened using command prompt
Conclusion	The test is successful.

Table 5-1 : Test 1 Testing Table

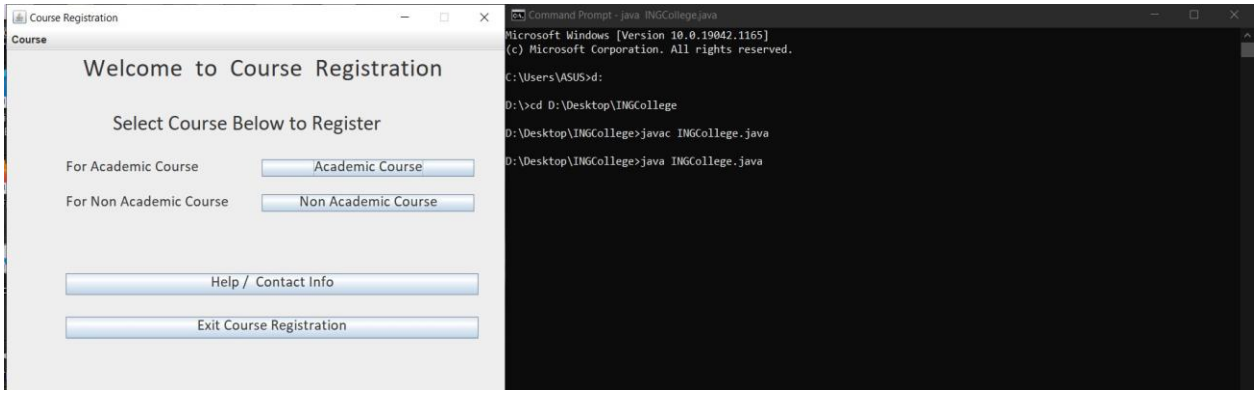

Figure 5-1: Test 1 Display Figure

5.2. Test 2

Test Number	2.a
Objective	To Add and Register course in Academic Course.
Action	All the Text Field is filled properly, and the courses is added through add course button while registered through registered button.

Expected Result	To add and register courses in Academic Course with Suitable Display.
Actual Result	The course was successfully added and registered in Academic Course.
Conclusion	The test is successful.

Table 5-2: Test 2 Testing Table for Academic Course

Output

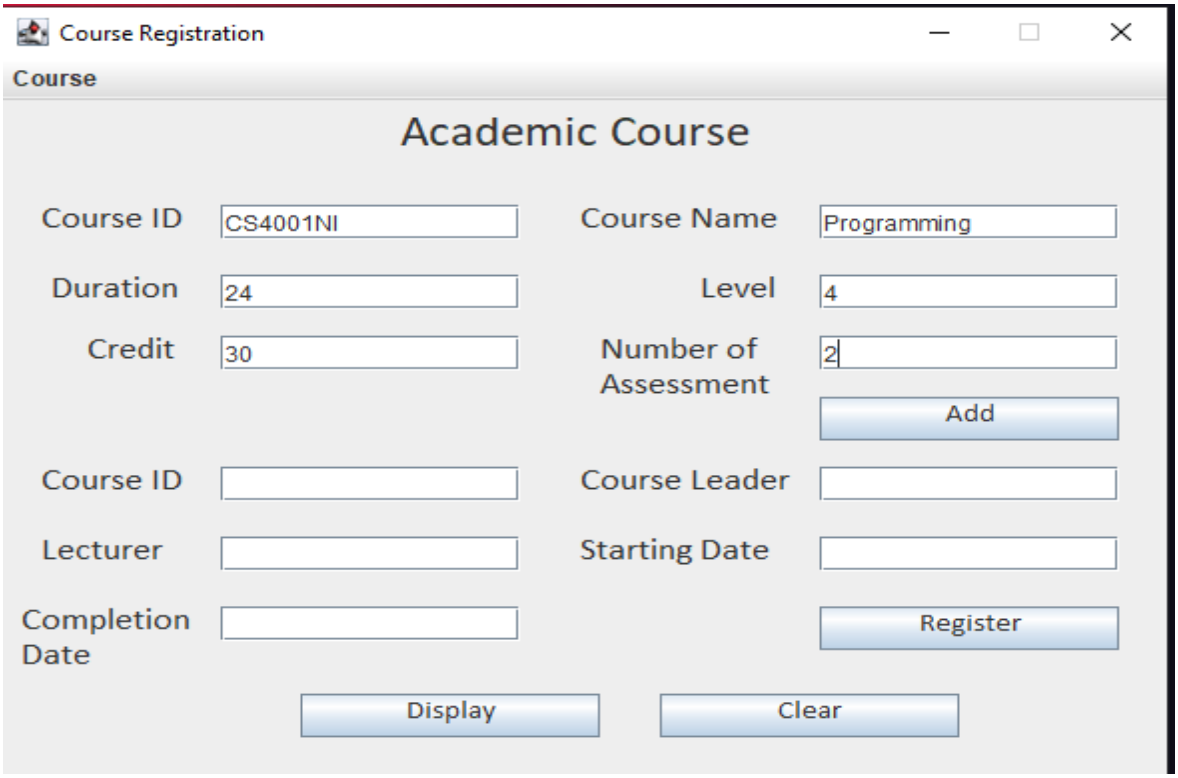

Figure 5-2: Adding course detail in Text Field

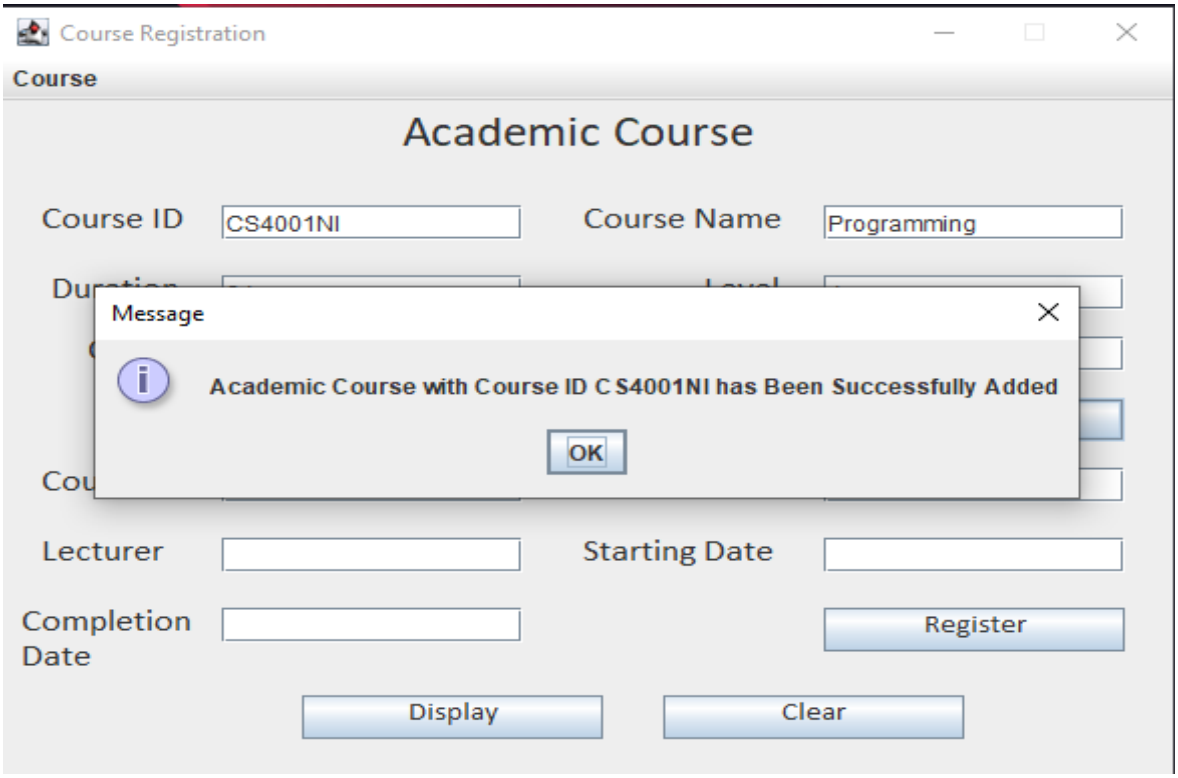

Figure 5-3: Course Successfully Added Display

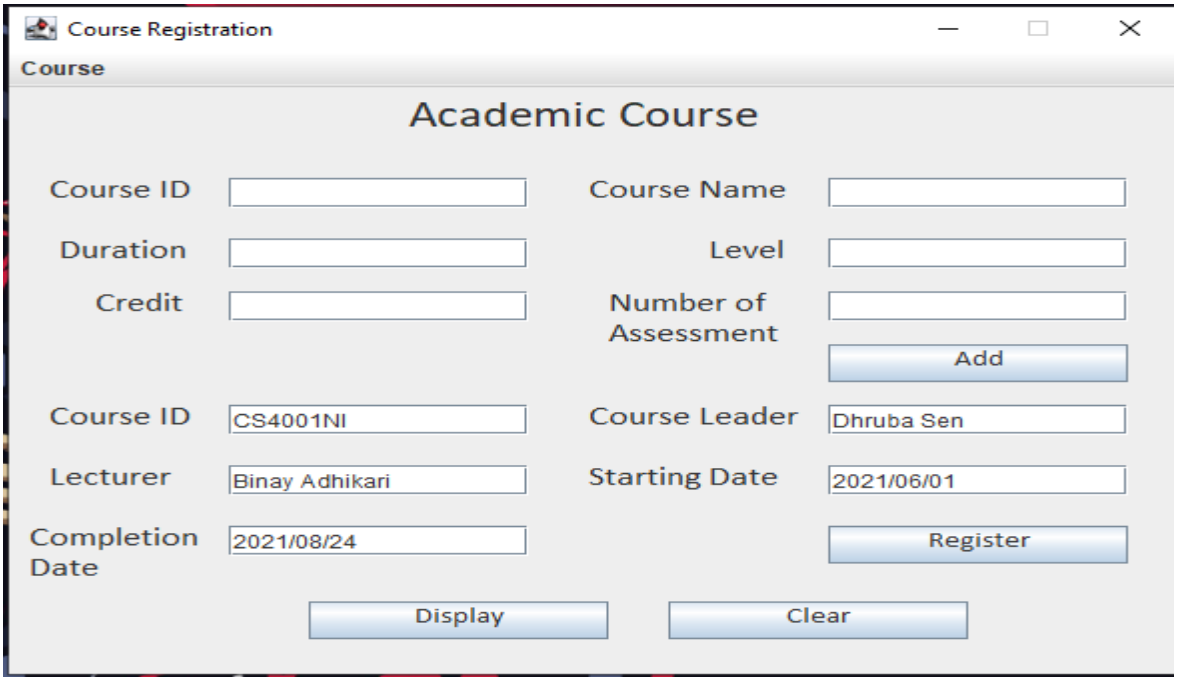

Figure 5-4: Adding Registration Detail in Text Field

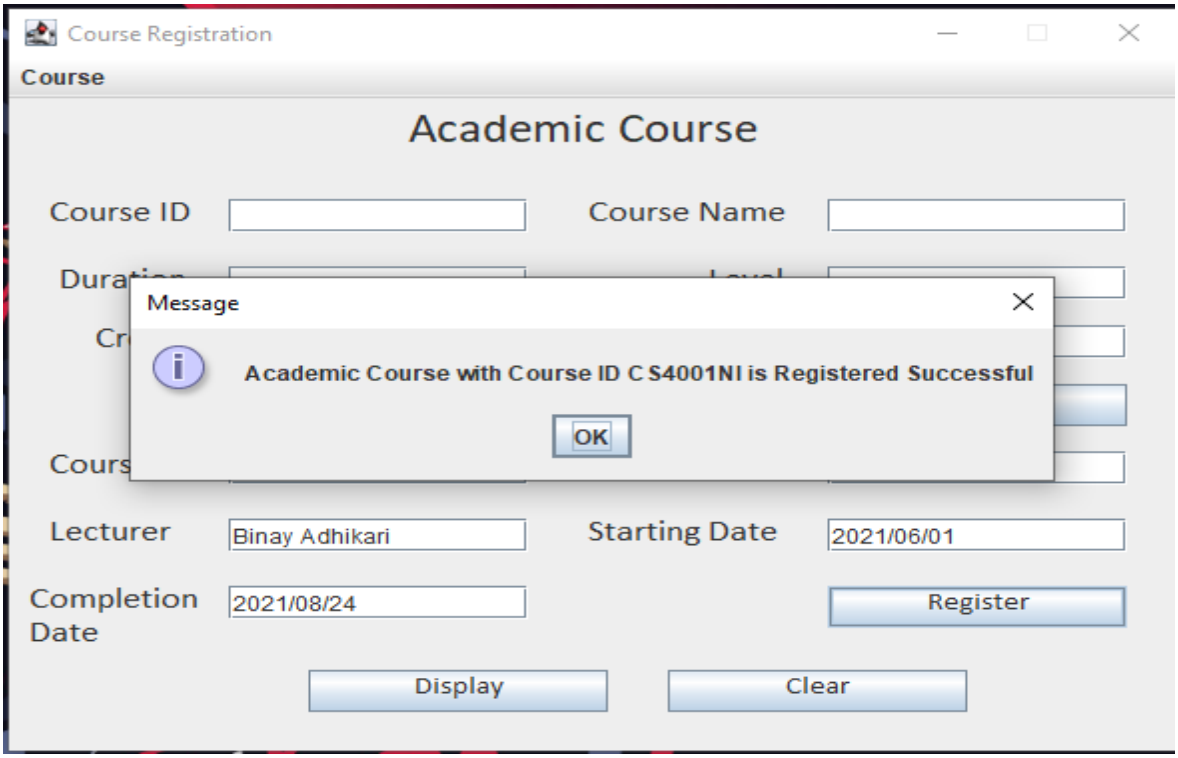

Figure 5-5: Course Successfully Registered Display

Test Number	2.b
Objective	To Add and Register course in Non-Academic Course.
Action	All the Text Field is filled properly, and the courses is added through add button while registered through register button.
Expected Result	To add and register courses in Non-Academic Course with Suitable Display.
Actual Result	The course was successfully added and registered in Non-Academic

	Course.
Conclusion	The test is successful.

Table 5-3: Test 2 Testing Table for Non-Academic Course

OUTPUT:

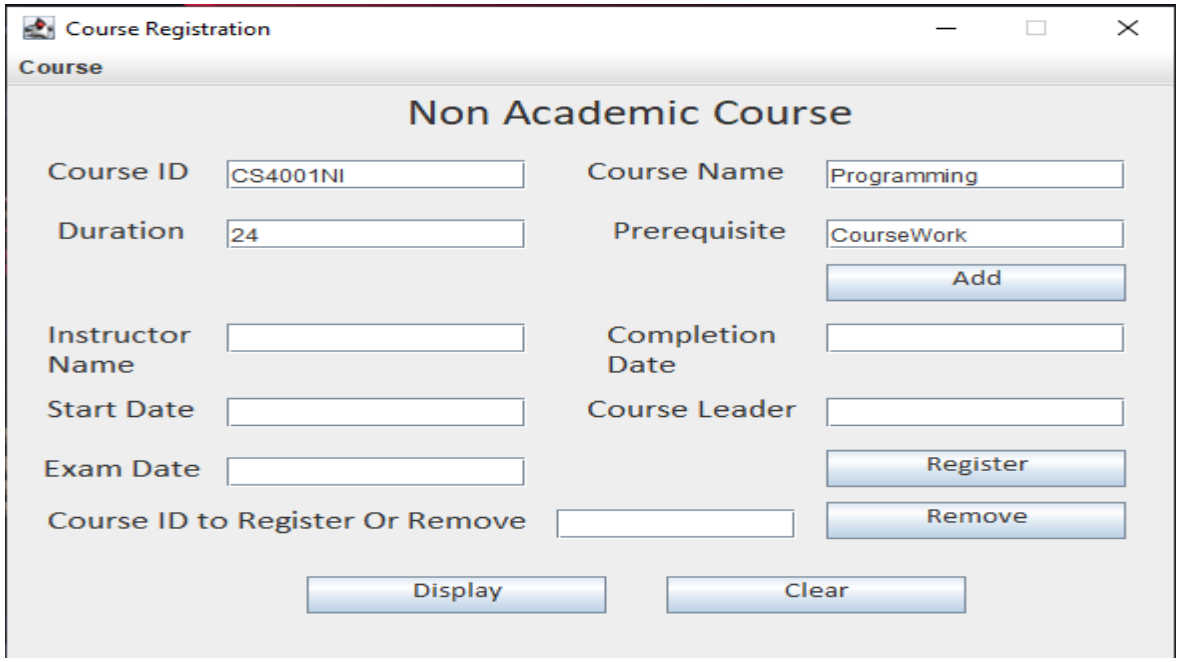

Figure 5-6: Adding course detail in Text Field

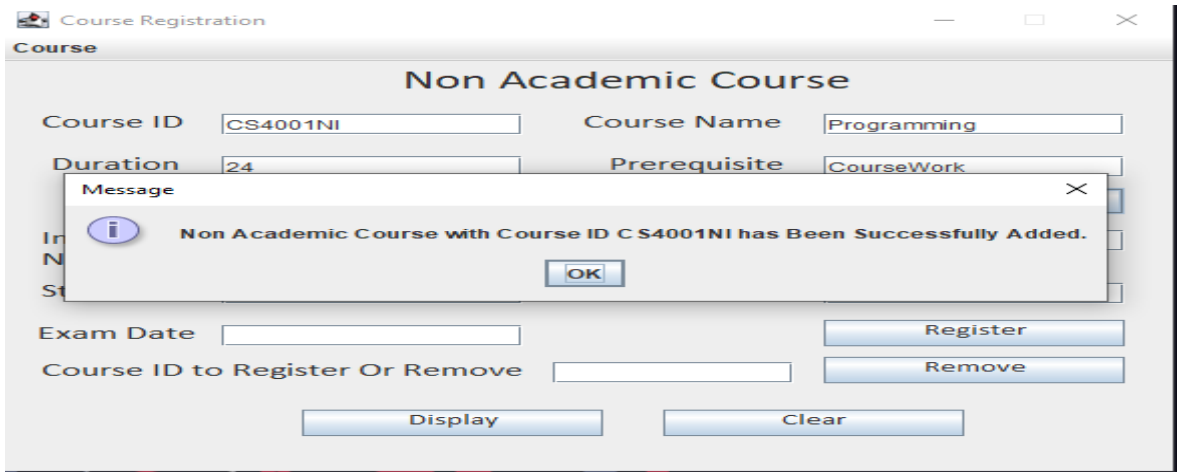

Figure 5-7: Course Successfully Added Display

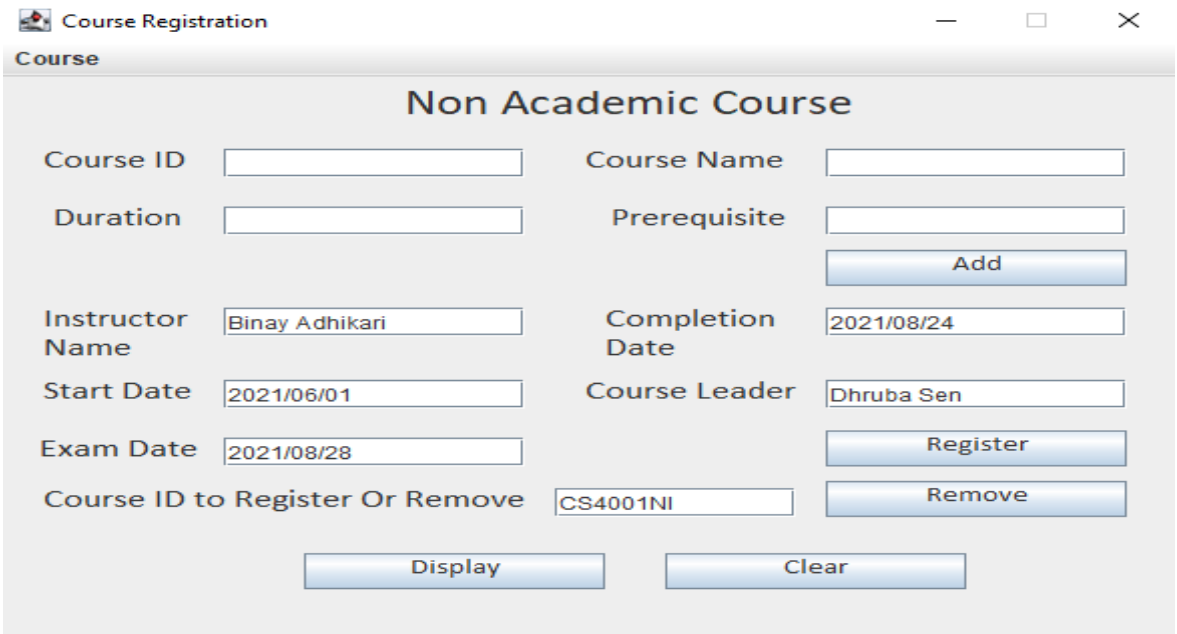

Figure 5-8: Adding course detail in Text Field

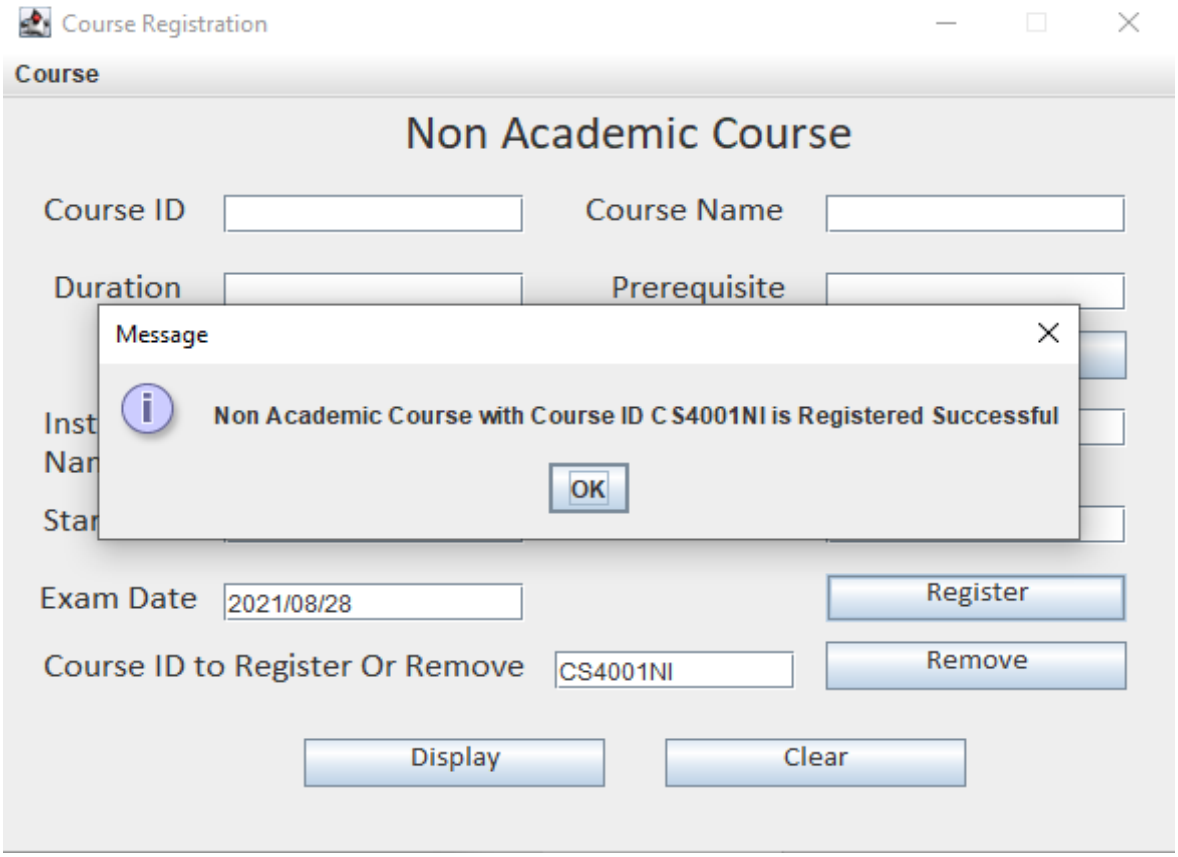

Figure 5-9: Course Successfully Registered Display

Test Number	2.c
Objective	To Remove course from Non-Academic Course.
Action	The Course ID is placed in TextField and Remove button is pressed.
Expected Result	To remove course from non-academic Course with suitable display.
Actual Result	The course is successfully removed.
Conclusion	The test is Successful.

Table 5-4: Test 2 Testing Table for Remove Button

OUTPUT:

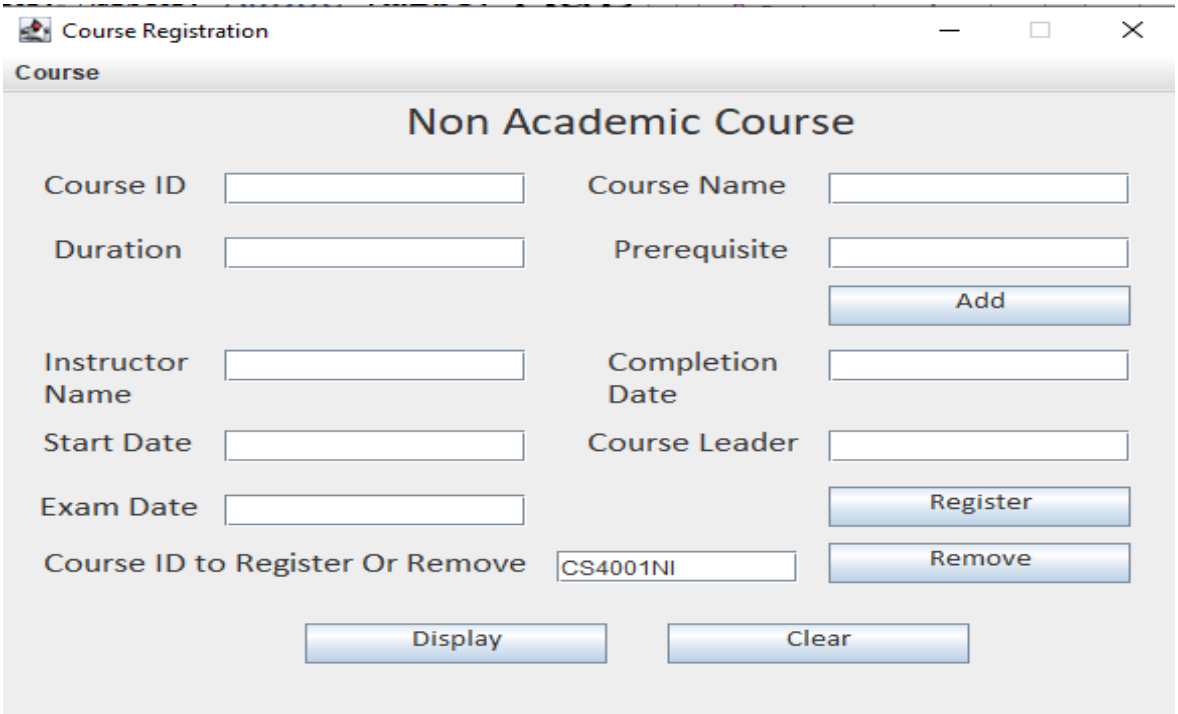

Figure 5-10: Adding Course ID for Removal

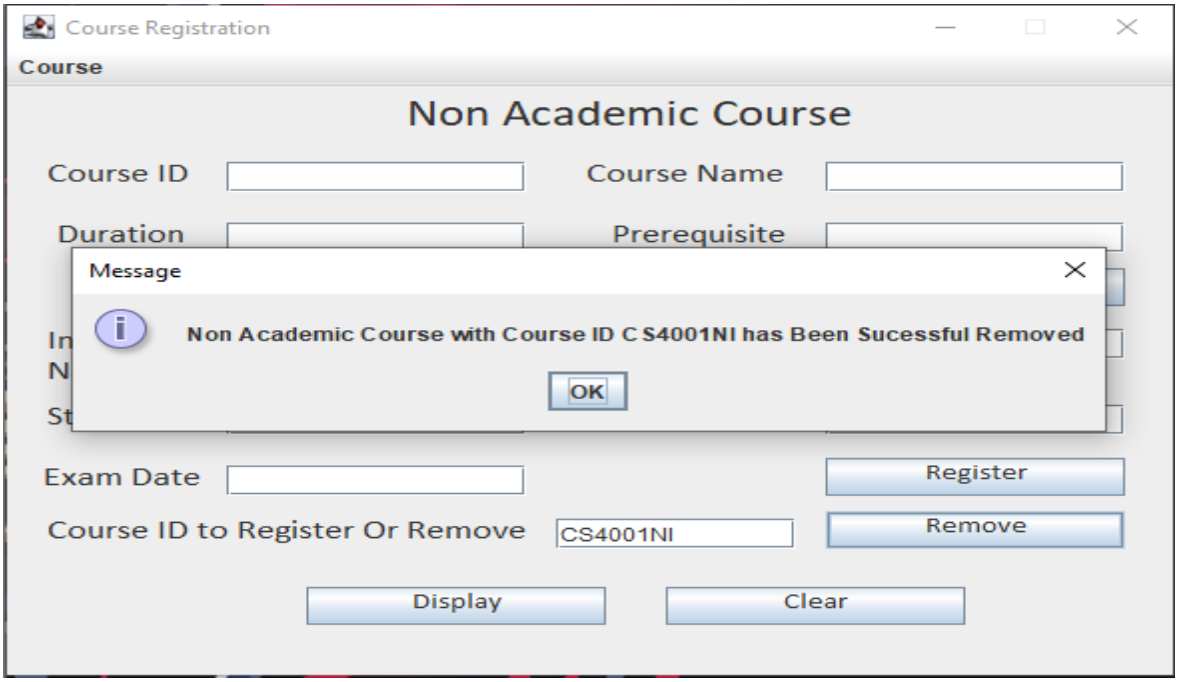

Figure 5-11: Course Successfully Removed Display

5.3. Test 3

Test	3.a
Objective	To Add and Register duplicate course ID Academic Course
Action	All the text fields is properly field, and courses is added and registered successfully. Again, the course is tried to add and registered using same course ID.
Expected Result	An error message was to show when duplicate courseID was added and registered again in Academic Course.
Actual Result	An error message is successfully displayed.
Observation	The text is Successful.

Table 5-5: Test 3 Testing Table for Academic Course

Output

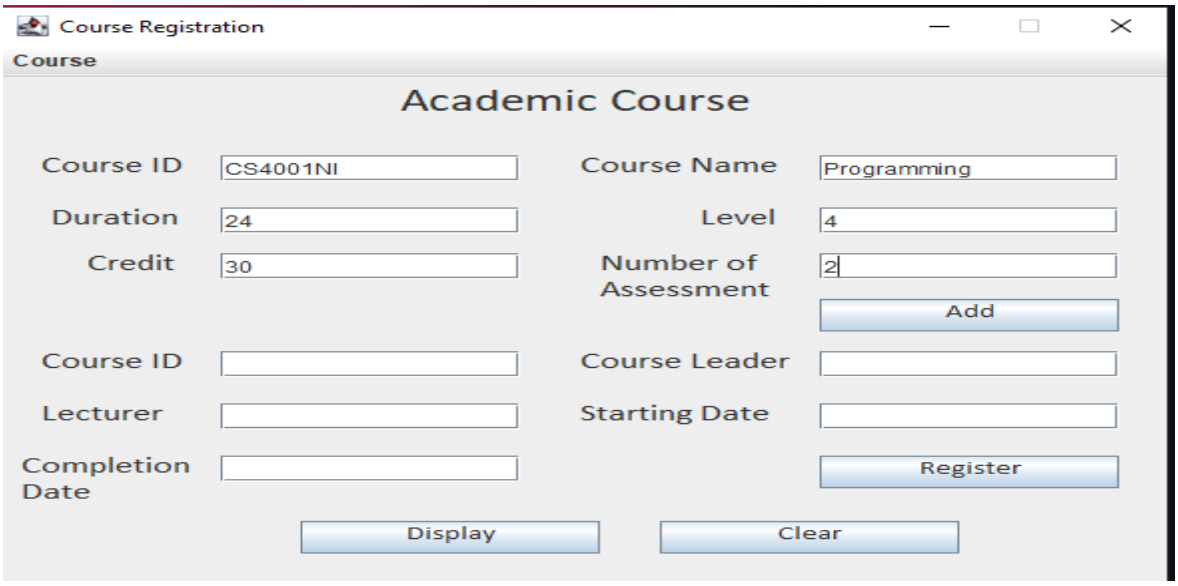

Figure 5-12: Adding Duplicate Course ID in Text Field

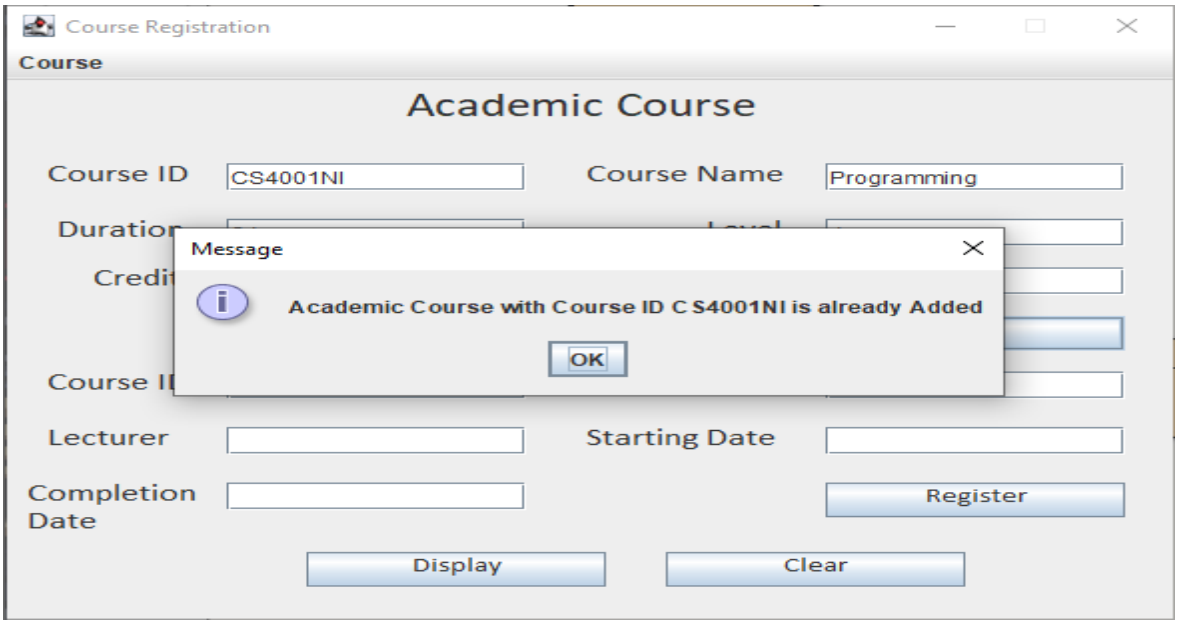

Figure 5-13: Course ID already Added Display

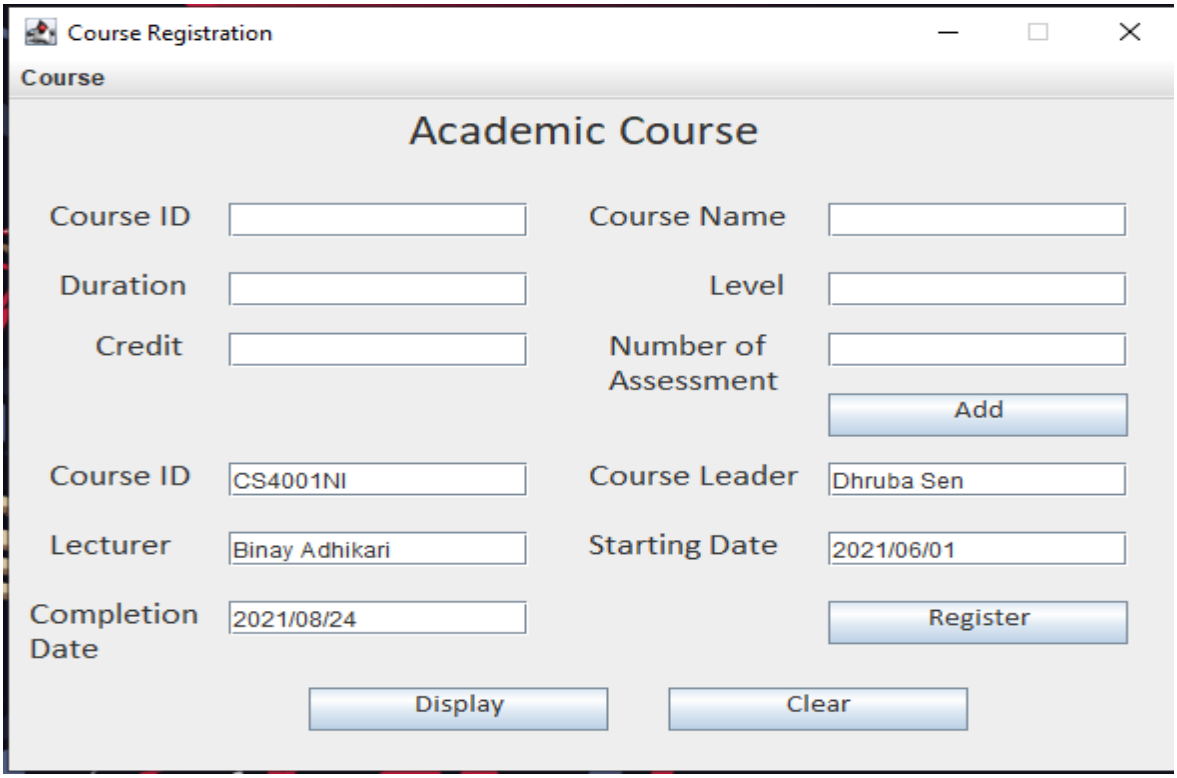

Figure 5-14: Adding Duplicate Course ID for Registration

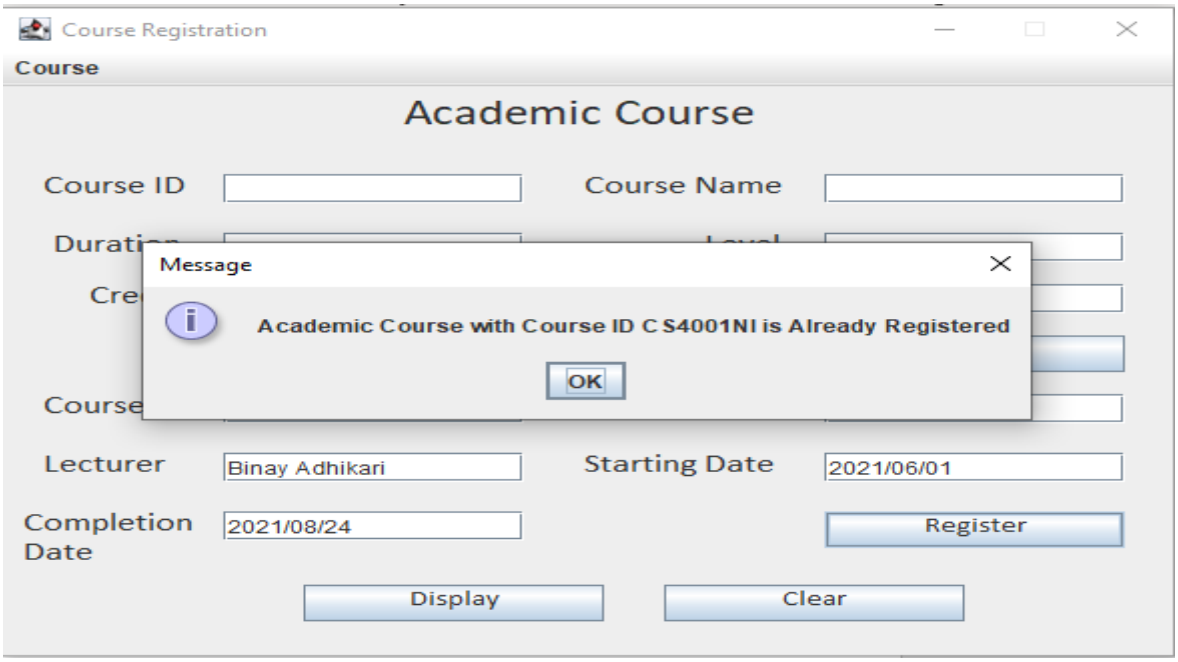

Figure 5-15: Course ID Already Registered Display

Test Number	3.b
Objective	To Add and Register duplicate course ID in Non-Academic Course
Action	All the text field is properly filled, and courses is added and registered successfully. Again, the ocurse is tried to add and register using same course id in Non-Academic Course.
Expected Result	An error message was to display when duplicate courseID was added and registered again in Non-Academic Course.
Actual Result	An error message is successfully displayed.
Observation	The test is Successful.

Table 5-6: Test 3 Testing Table for Non-Academic Course

OUTPUT:

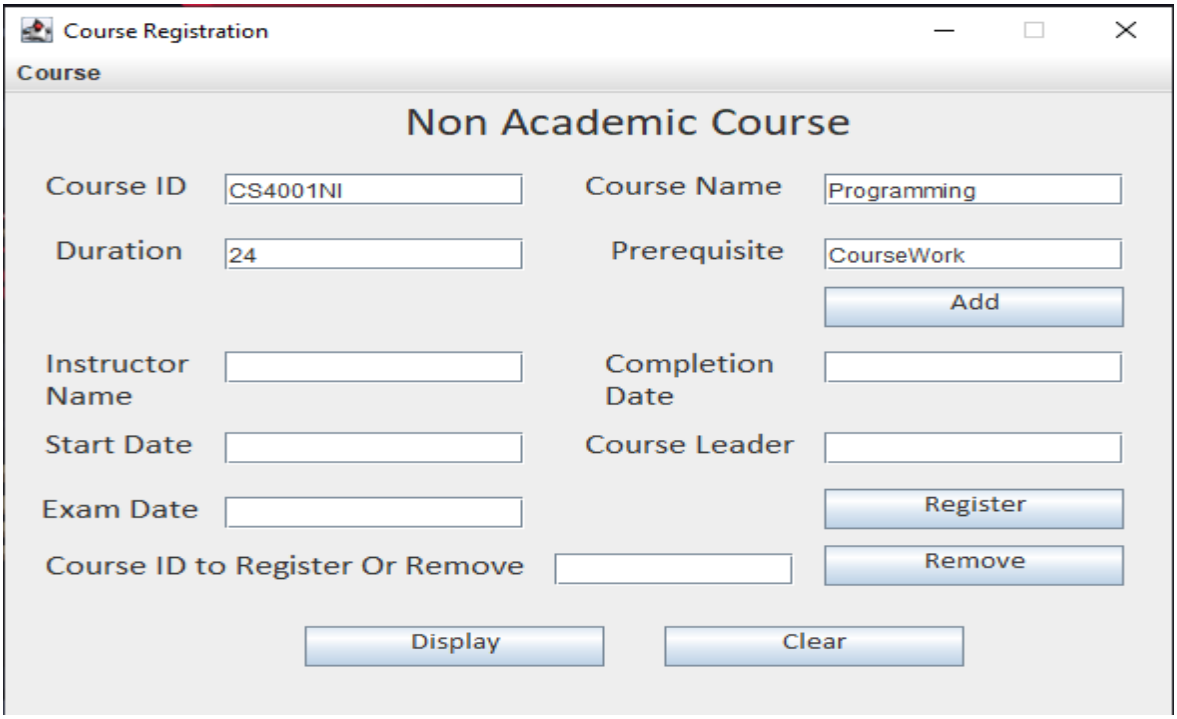

Figure 5-16: Adding Duplicate Course ID in Text Field

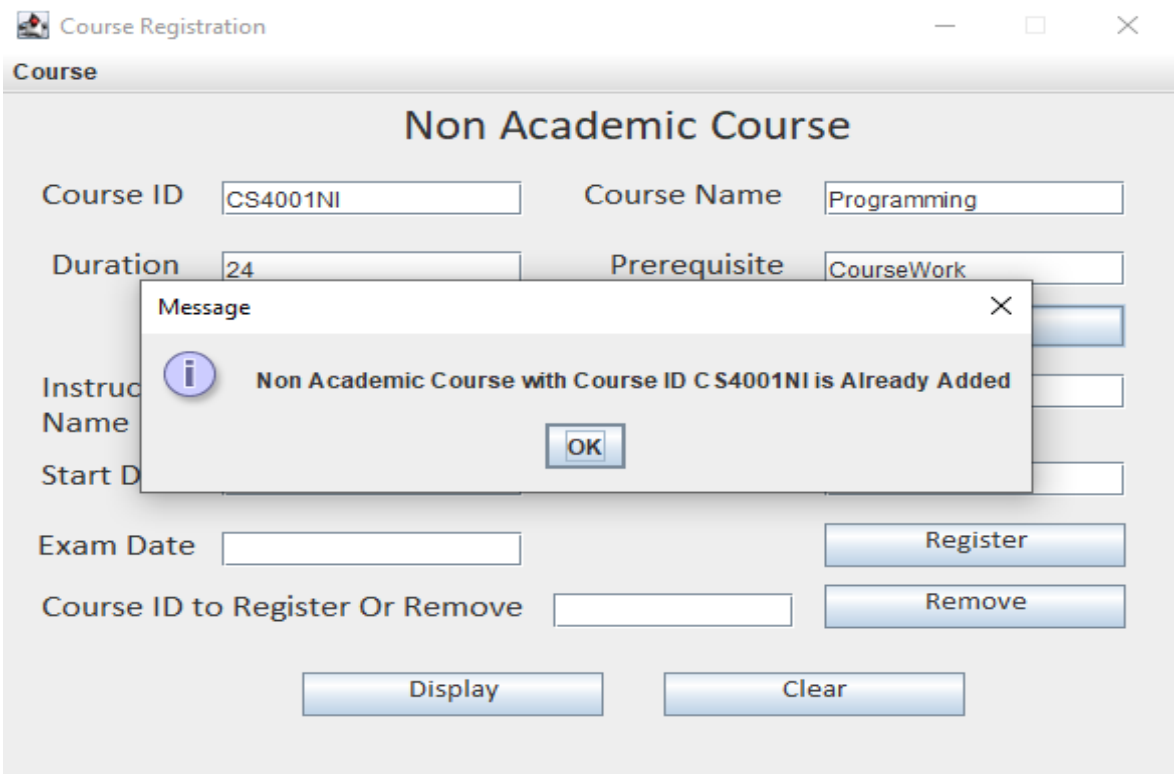

Figure 5-17: Course ID already Added Display

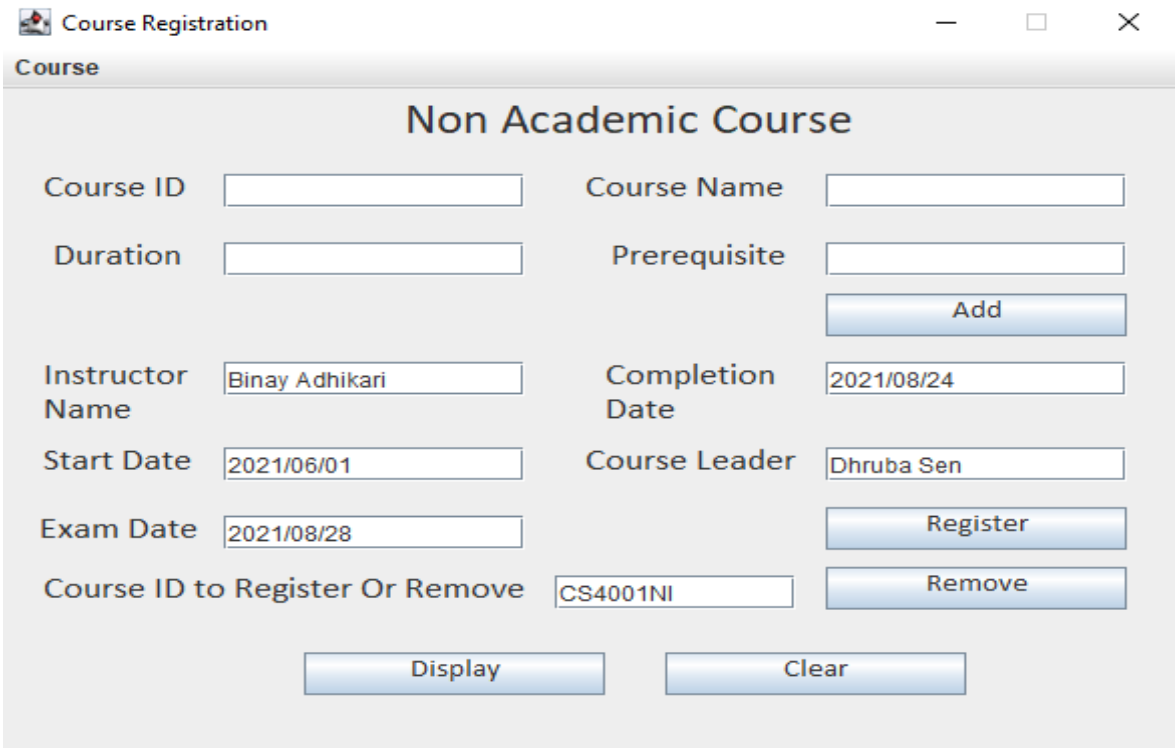

Figure 5-18: Adding duplicate Course ID for Registration

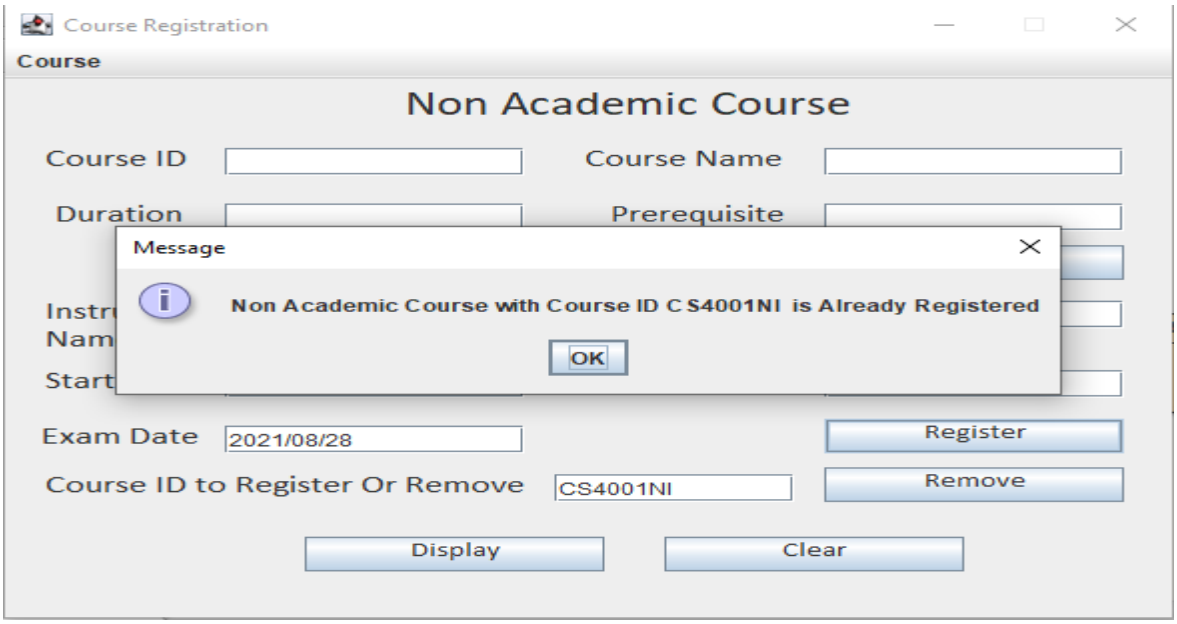

Figure 5-19: Course ID already Registered

Test Number	3.c
Objective	To Remove Same Course which was already Removed.
Action	The course is added and registered successfully, and is removed. Again, the text field is properly filled with same CourseID which was removed and remove button is pressed.
Expected Result	An error message should be display.
Actual Result	An error messsage is displayed successfully displaying no course with this cousreID exists.
Observation	The test is Successful.

Table 5-7: Test 3 Testing Table for Remove Button

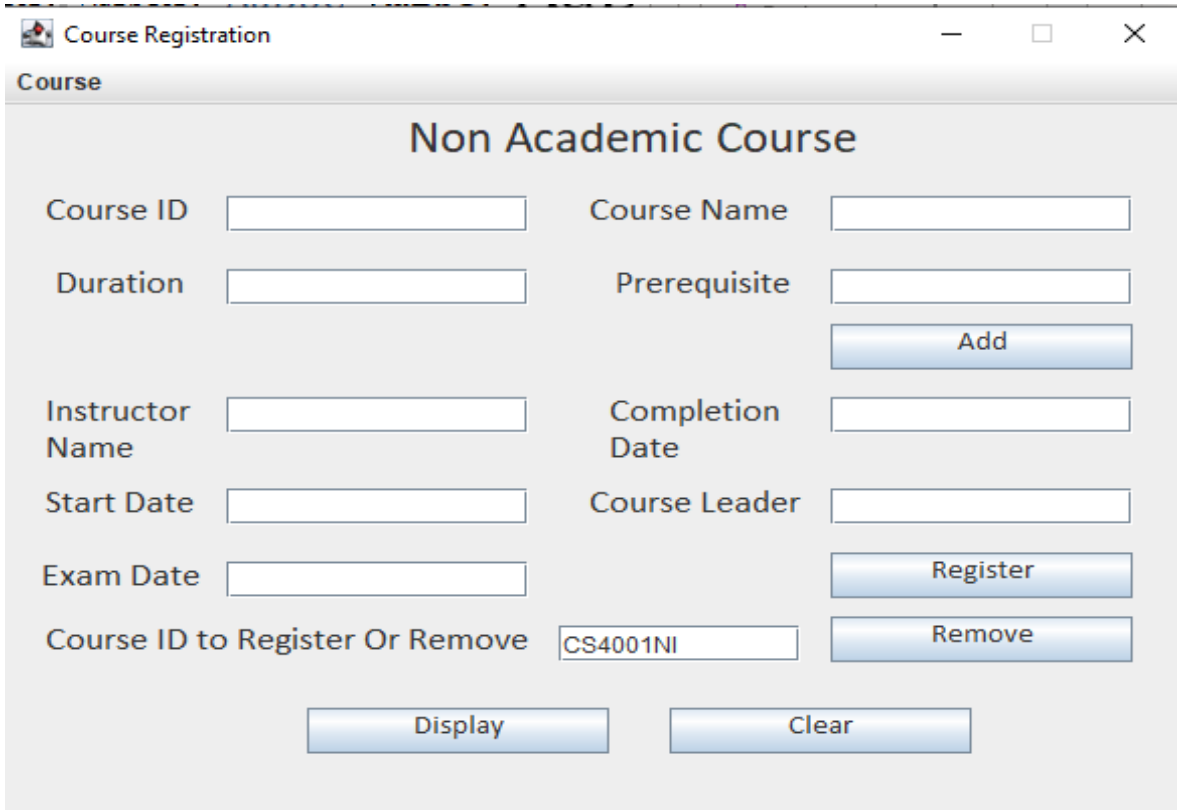

Figure 5-20: Inserting Course ID in Text Field

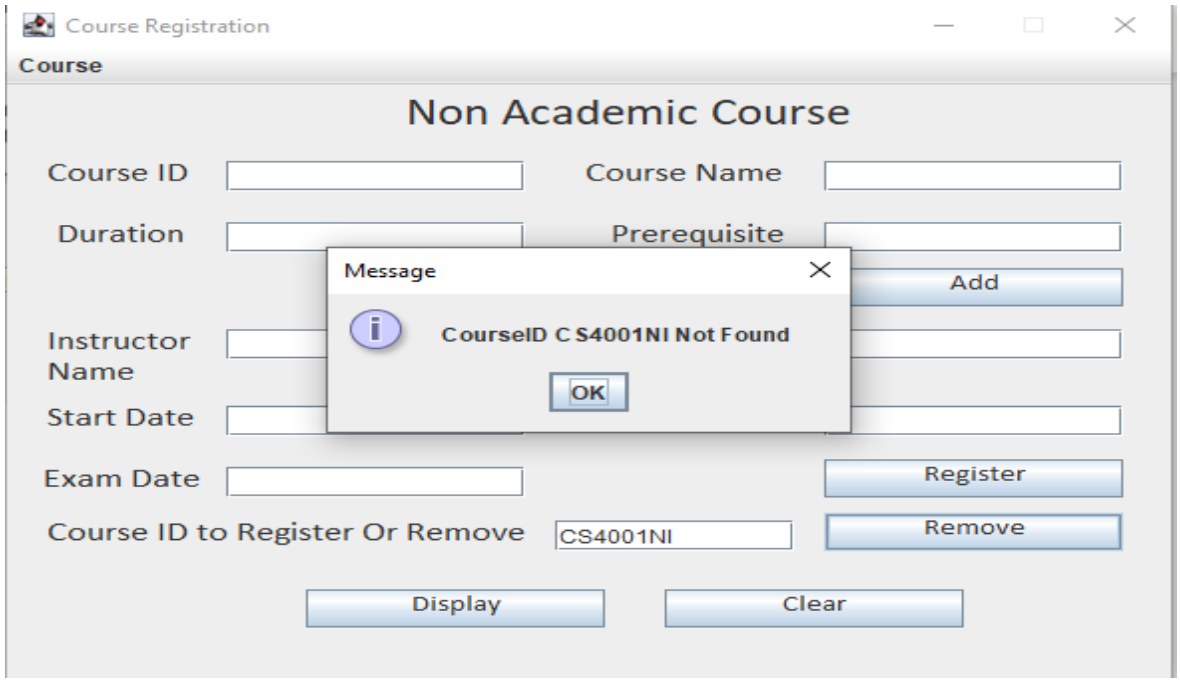

Figure 5-21: Course ID not Found Display

6. Error Detection and Correction

There are three types of error that arise during the coding in bluej. Some of the errors and their correction are as follow:

6.1. Syntax Error

Syntax error is an error in the source code which can be caused by grammatical mistakes such as missing a semicolon at the end of a line or an extra bracket at the end of function. (Christensson P., 2012)

One of the Syntax error which can be commonly found is given below:

```
public INGCollege(){

    //Creating a Frame
    jf = new JFrame("Course Registration");
    jf.setDefaultCloseOperation(jf.EXIT_ON_CLOSE);

    // Creating a Frame for Result
    jfResult = new JFrame("Display")
```

Figure 6-1: Syntax Error Missing Semicolon

```
public INGCollege(){

    //Creating a Frame
    jf = new JFrame("Course Registration");
    jf.setDefaultCloseOperation(jf.EXIT_ON_CLOSE);

    // Creating a Frame for Result
    jfResult = new JFrame("Display");
```

Figure 6-2: Syntax Error Correction

6.2. Semantic Error

Semantic error is an error which occurs when a statement is syntactically valid but doesn't give the intended result which can occur due to wrong variable name or function and can be a tricky one to solve. (Alex, 2020). In java, the compilar finds the particular sematic error in most cases.

One of the semantic errors that can be found in programming is given below:

```
btnAregister.addActionListener ( new ActionListener(){
    public void actionPerformed(ActionEvent e){
        String courseID, courseLeader, lecturerName, startingDate, completionDate;
        boolean isRegistered = false;
        if ( txtAcourseID.getText().isEmpty() || txtAcourseLeader.getText().isEmpty() || txtAlecturerName.getText().isEmp
            JOptionPane.showMessageDialog(jf,"All of the Field are Required","Aleart",JOptionPane.WARNING_MESSAGE);
        }
```

Figure 6-3: Semantic Error in txtAcourseID.getText()

In if() condition, txtAcourseID.getText().isEmpty() is an wrong command where the correct code should have been txtAcourseID2.getText().isEmpty(). The code is valid but the code checks an different textfield for empty which causes an sementric error in program.

```
btnAregister.addActionListener ( new ActionListener(){
    public void actionPerformed(ActionEvent e){
        String courseID, courseLeader, lecturerName, startingDate, completionDate;
        boolean isRegistered = false;
        if ( txtAcourseID2.getText().isEmpty() || txtAcourseLeader.getText().isEmpty() || txtAlecturerName.getText().isEm
            JOptionPane.showMessageDialog(jf,"All of the Field are Required","Aleart",JOptionPane.WARNING_MESSAGE);
        }
```

Figure 6-4: Semantic Error Correction

6.3. Logical Error

Logical error is an unexpected behavior or wrong output that is caused by mistake in a program source code. It can also be classified as runtime error that can result in program to display incorrect output or sometime crash when running.

```
// Adding the Register Button in Academic Course
btnAregister = new JButton("Register");
btnAregister.setBounds(410,290,0,25);
btnAregister.setFont(fb);
jAcademicCourse.add(btnAregister);
```

Figure 6-5: Logical Error in setBounds()

In btnAregister.setBounds(), the width of button should have been 150 but 0 is given. The code is valid, but the intended result is not display.

```
// Adding the Register Button in Academic Course
btnAregister = new JButton("Register");
btnAregister.setBounds(410,290,150,25);
btnAregister.setFont(fb);
jAcademicCourse.add(btnAregister);
```

Figure 6-6: Logical Error Corrected

7. Conclusion

This is the second coursework of programming module where we were asked to carry a task to create a Graphical User Interface (GUI) keeping first coursework as base. This coursework helped in better understanding about class diagram, pseudocode, GUI, and proper use of classes. Pseudocode can be understood by non-programmer or any person unfamiliar with programming language. getText() property was used for extracting user input from GUI to passing the code into classes. We were able to learn Object casting, iteration, and different function which was important for development of this project.

All the difficulties and problems that arise during coursework was solved and research with help of Instructor, website, articles, friends, and other resources. Error occurred during development of coursework in GUI was solved by discussion with friends and instructor guide.

8. Appendix 1

➔ INGCollege Class:

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

public class INGCollege
{
    // Frame Declaration and Menu Bar
    JFrame jf, jfResult;
    JPanel jWelcometoCourse, jAcademicCourse;
    Font ff, ft, fb, fr;
    JMenuBar jmb;
    JMenu file;
    JMenuItem academicCourse, nonAcademicCourse, exitApp;

    // JDeclaring Label and Button for Welcome Screen
    JLabel homeWelcome, homeSelect, homeAcademicCourse,
homeNonAcademicCourse;
    JButton btnHacademicCourse, btnHnonAcademicCourse, btnHhelp, btnHexit;

    //Declaring JLabel, JTextField and Button for Academic Course
    JLabel lblAcademicCourse, lblAcourseID, lblAcourseName, lblAduration,
lblAlevel, lblAcredit, lblAnumOfAssessment, lblAnumOfAssessment2,
lblAcourseID2, lblAcourseLeader, lblAlecturerName, lblAstartingDate,
lblAcompletionDate, lblAcompletionDate2;
    JTextField txtAcourseID, txtAcourseName, txtAduration, txtAlevel, txtAcredit,
txtAnumOfAssessment, txtAcourseID2, txtAcourseLeader, txtAlecturerName,
txtAstartingDate, txtAcompletionDate;
```

```
    JButton btnAadd, btnAregister, btnAdisplay, btnAclear;

    // Declaring JLabel, TextField and Button for Non Academic Course
    JPanel jNonAcademicCourse;
    JLabel lblNonAcademicCourse, lblNAcourseID, lblNAcourseName,
lblNAduration, lblNAprerequisite, lblNAinstructorName, lblNAinstructorName2,
lblNAcompletionDate, lblNAcompletionDate2, lblNAstartDate,
lblNAcourseLeader, lblNAexamDate, lblNAcourseID2;
    JTextField txtNAcourseID, txtNAcourseName, txtNAduration,
txtNAprerequisite, txtNAinstructorName, txtNAcompletionDate, txtNAstartDate,
txtNAcourseLeader, txtNAexamDate, txtNAcourseID2;
    JButton btnNAadd, btnNAregister, btnNAremove, btnNAdisplay, btnNAclear;


    public INGCollege(){

        //Creating a Frame
        jf = new JFrame("Course Registration");
        jf.setDefaultCloseOperation(jf.EXIT_ON_CLOSE);

        // Creating a Frame for Result
        jfResult = new JFrame("Display");

        // Creating a Font for Both Academic Course and Non_Academic Course
        ff = new Font("Calibri",Font.PLAIN,18);
        ft = new Font("Calibri",Font.PLAIN,25);
        fb = new Font("Calibri",Font.PLAIN,15);
        fr = new Font("Calibri",Font.PLAIN,13);

        // Creating Menu Bar
        jmb = new JMenuBar();
```

```
jf.setJMenuBar(jmb);

// Creating Menu
file = new JMenu("Course");
jmb.add(file);

//Creating Menu Item
academicCourse = new JMenuItem("Academic Course");
nonAcademicCourse = new JMenuItem("Non Academic Course");
exitApp = new JMenuItem("Exit");

// Adding MenuItems in Menu and Adding Seperator
file.add(academicCourse);
file.addSeparator();
file.add(nonAcademicCourse);
file.addSeparator();
file.add(exitApp);

// Creating a Panel for Welcome to Course Registration
jWelcometoCourse = new JPanel();


// Adding Welcome Screen Label to Course Registration Panel
homeWelcome = new JLabel("Welcome  to  Course  Registration");
homeWelcome.setBounds(90,10,600,30);
Font hf = new Font("Calibri",Font.PLAIN,30);
homeWelcome.setBackground(Color.RED);
homeWelcome.setFont(hf);
jWelcometoCourse.add(homeWelcome);

// Adding Select Course Label
```

```
homeSelect = new JLabel ("Select Course Below to Register");
homeSelect.setBounds(125,75,600,25);
homeSelect.setFont(ft);
jWelcometoCourse.add(homeSelect);

// Adding Academic Course Label
homeAcademicCourse = new JLabel("For Academic Course");
homeAcademicCourse.setBounds(70,125,600,20);
homeAcademicCourse.setFont(ff);
jWelcometoCourse.add(homeAcademicCourse);

// Adding Non Academic Course Label
homeNonAcademicCourse = new JLabel("For Non Academic Course");
homeNonAcademicCourse.setBounds(70,165,600,20);
homeNonAcademicCourse.setFont(ff);
jWelcometoCourse.add(homeNonAcademicCourse);

// Adding Buttons on Welcome Screen
btnHacademicCourse = new JButton("Academic Course");
btnHacademicCourse.setBounds( 300,125,250,20);
btnHacademicCourse.setFont(ff);
jWelcometoCourse.add(btnHacademicCourse);

// Button For Home Registration Course
btnHnonAcademicCourse = new JButton("Non Academic Course");
btnHnonAcademicCourse.setBounds(300,165,250,20);
btnHnonAcademicCourse.setFont(ff);
jWelcometoCourse.add(btnHnonAcademicCourse);

btnHhelp = new JButton("Help /  Contact Info");
btnHhelp.setBounds(70,255,485,25);
```

```
btnHhelp.setFont(ff);
jWelcometoCourse.add(btnHhelp);

btnHexit = new JButton("Exit Course Registration");
btnHexit.setBounds(70,305,485,25);
btnHexit.setFont(ff);
jWelcometoCourse.add(btnHexit);

// Adding Properties to Welcome to Course Registration
jWelcometoCourse.setLayout(null);
jWelcometoCourse.setBounds(0,0,600,450);
jWelcometoCourse.setVisible(true);

// Adding WelcometoCourse to Main Frame
jf.add(jWelcometoCourse);

// Creating a Panel For Academic Course
jAcademicCourse = new JPanel();
jAcademicCourse.setLayout(null);
jAcademicCourse.setBounds(0,0,600,450);
jAcademicCourse.setVisible(false);

// Creating a Title for Academic Course
lblAcademicCourse = new JLabel("Academic Course");
lblAcademicCourse.setBounds(200,10,300,25);
lblAcademicCourse.setFont(ft);
jAcademicCourse.add(lblAcademicCourse);

/* Creating all the Necessary Label and Text Field
for Academic Course */
```

```
// Adding CourseID Label AND TextField on Academic Course
lblAcourseID = new JLabel("Course ID");
JTextField txtAcourseID = new JTextField();
lblAcourseID.setBounds(20,60,150,20);
txtAcourseID.setBounds(110,60,150,20);
lblAcourseID.setFont(ff);
jAcademicCourse.add(lblAcourseID);
jAcademicCourse.add(txtAcourseID);

// Adding CourseName Label And TextField on Academic Course
lblAcourseName = new JLabel("Course Name");
txtAcourseName = new JTextField();
lblAcourseName.setBounds(290,60,150,20);
txtAcourseName.setBounds(410,60,150,20);
lblAcourseName.setFont(ff);
jAcademicCourse.add(lblAcourseName);
jAcademicCourse.add(txtAcourseName);

// Adding Duration Label And TextField on Academic Course
lblAduration = new JLabel("Duration");
txtAduration = new JTextField();
lblAduration.setBounds(25,100,150,20);
txtAduration.setBounds(110,100,150,20);
lblAduration.setFont(ff);
jAcademicCourse.add(lblAduration);
jAcademicCourse.add(txtAduration);

// Adding Level Label and TextField on Academic Course
lblAlevel = new JLabel("Level");
txtAlevel = new JTextField();
lblAlevel.setBounds(350,100,150,20);
```

```
txtAlevel.setBounds(410,100,150,20);
lblAlevel.setFont(ff);
jAcademicCourse.add(txtAlevel);
jAcademicCourse.add(lblAlevel);

// Adding Credit Label and TextField on Academic Course
lblAcredit = new JLabel("Credit");
txtAcredit = new JTextField();
lblAcredit.setBounds(43,135,150,20);
txtAcredit.setBounds(110,135,150,20);
lblAcredit.setFont(ff);
jAcademicCourse.add(lblAcredit);
jAcademicCourse.add(txtAcredit);

// Adding Number of Assessment Label and TextField on Academic Course
lblAnumOfAssessment = new JLabel("Number of");
lblAnumOfAssessment2 = new JLabel("Assessment");
txtAnumOfAssessment = new JTextField();
lblAnumOfAssessment.setBounds(300,135,250,20);
lblAnumOfAssessment2.setBounds(300,155,250,20);
txtAnumOfAssessment.setBounds(410,135,150,20);
lblAnumOfAssessment.setFont(ff);
lblAnumOfAssessment2.setFont(ff);
jAcademicCourse.add(lblAnumOfAssessment);
jAcademicCourse.add(lblAnumOfAssessment2);
jAcademicCourse.add(txtAnumOfAssessment);

// Adding CourseID Label and TextField
lblAcourseID2 = new JLabel("Course ID");
txtAcourseID2 = new JTextField();
lblAcourseID2.setBounds(20,210,150,20);
```

```
txtAcourseID2.setBounds(110,210,150,20);
lblAcourseID2.setFont(ff);
jAcademicCourse.add(lblAcourseID2);
jAcademicCourse.add(txtAcourseID2);
// Adding CourseLeader Label and TextField in Academic Course
lblAcourseLeader = new JLabel("Course Leader");
txtAcourseLeader = new JTextField();
lblAcourseLeader.setBounds(290,210,150,20);
txtAcourseLeader.setBounds(410,210,150,20);
lblAcourseLeader.setFont(ff);
jAcademicCourse.add(lblAcourseLeader);
jAcademicCourse.add(txtAcourseLeader);

// Adding LecturerName Label and TextField in Academic Course
lblAlecturerName = new JLabel("Lecturer");
txtAlecturerName = new JTextField();
lblAlecturerName.setBounds(20,250,150,20);
txtAlecturerName.setBounds(110,250,150,20);
lblAlecturerName.setFont(ff);
jAcademicCourse.add(lblAlecturerName);
jAcademicCourse.add(txtAlecturerName);

// Adding Starting Date Label and TextField in Academic Course
lblAstartingDate = new JLabel("Starting Date");
txtAstartingDate = new JTextField();
lblAstartingDate.setBounds(290,250,150,20);
txtAstartingDate.setBounds(410,250,150,20);
lblAstartingDate.setFont(ff);
jAcademicCourse.add(lblAstartingDate);
jAcademicCourse.add(txtAstartingDate);
```

```
// Adding Completion Date Label and TextField in Academic Course
lblAcompletionDate = new JLabel("Completion");
lblAcompletionDate2 = new JLabel("Date");
txtAcompletionDate = new JTextField();
lblAcompletionDate.setBounds(10,290,150,20);
lblAcompletionDate2.setBounds(10,310,150,20);
txtAcompletionDate.setBounds(110,290,150,20);
lblAcompletionDate.setFont(ff);
lblAcompletionDate2.setFont(ff);
jAcademicCourse.add(lblAcompletionDate);
jAcademicCourse.add(lblAcompletionDate2);
jAcademicCourse.add(txtAcompletionDate);

// Creating a Array for ArrayList of AcademicCourse
ArrayList <Course> AcademicCourseList = new ArrayList <Course>();

// Creating the Buttons Required for Academic Course

// Adding the Add Button in Academic Course
btnAadd = new JButton("Add");
btnAadd.setBounds(410,170,150,25);
btnAadd.setFont(fb);
jAcademicCourse.add(btnAadd);

btnAadd.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
        String courseID, courseName, level, credit;
        int duration, numOfAssessment;
        boolean checkInt = false;
        boolean check = false;
        duration = 0;
```

```
            numOfAssessment = 0;
            if (txtAcourseID.getText().isEmpty() ||
txtAcourseName.getText().isEmpty() || txtAlevel.getText().isEmpty() ||
txtAcredit.getText().isEmpty() || txtAduration.getText().isEmpty() ||
txtAnumOfAssessment.getText().isEmpty() ){
                JOptionPane.showMessageDialog(jf,"All of the Field are
Required","Aleart",JOptionPane.WARNING_MESSAGE);
            }
            else{
                courseID = txtAcourseID.getText();
                courseName = txtAcourseName.getText();
                level = txtAlevel.getText();
                credit = txtAcredit.getText();

                try{
                    duration = Integer.parseInt(txtAduration.getText());
                    numOfAssessment =
Integer.parseInt(txtAnumOfAssessment.getText());
                    checkInt = true;
                }
                catch(Exception ex){
                    JOptionPane.showMessageDialog(jf,"Input Valid Value in
Duration and Number of
Assessment","Aleart",JOptionPane.WARNING_MESSAGE);
                }

                // Checking If CourseID has been Added
                for (Course ac: AcademicCourseList){
                    if(ac.getCourseID().equals(courseID)){
                        check = true;
                        break;
```

```
                }
            }

            // Adding AcademicCourse Details
            if(check == false && checkInt == true){
                AcademicCourse aCourse = new AcademicCourse(courseID,
courseName, duration,level, credit, numOfAssessment);
                AcademicCourseList.add(aCourse);
                JOptionPane.showMessageDialog(jf,"Academic Course with
Course ID "+ courseID + " has Been Successfully Added");
            }
            else if (checkInt == true) {
                    JOptionPane.showMessageDialog(jf,"Academic Course with
Course ID "+ courseID +" is already Added");
            }
        }
    }
});


// Adding the Register Button in Academic Course
btnAregister = new JButton("Register");
btnAregister.setBounds(410,290,150,25);
btnAregister.setFont(fb);
jAcademicCourse.add(btnAregister);

btnAregister.addActionListener ( new ActionListener(){
    public void actionPerformed(ActionEvent e){
        String courseID, courseLeader, lecturerName, startingDate,
completionDate;
        boolean isRegistered = false;
```

```
            if ( txtAcourseID2.getText().isEmpty() ||
txtAcourseLeader.getText().isEmpty() || txtAlecturerName.getText().isEmpty() ||
txtAstartingDate.getText().isEmpty() || txtAcompletionDate.getText().isEmpty() ){
                JOptionPane.showMessageDialog(jf,"All of the Field are
Required","Aleart",JOptionPane.WARNING_MESSAGE);
            }
            else{
                courseID = txtAcourseID2.getText();
                courseLeader = txtAcourseLeader.getText();
                lecturerName = txtAlecturerName.getText();
                startingDate = txtAstartingDate.getText();
                completionDate = txtAcompletionDate.getText();

                int i = 0;
                for ( Course ac : AcademicCourseList){
                    if(ac.getCourseID().equals(courseID)){
                        AcademicCourse acGet =
(AcademicCourse)(AcademicCourseList.get(i));
                        isRegistered = true;
                        if (!acGet.getIsRegistered()){
                            acGet.register(courseLeader, lecturerName, startingDate,
completionDate);
                            JOptionPane.showMessageDialog(jf,"Academic Course with
Course ID "+ courseID +" is Registered Successful");
                        }
                        else{
                            JOptionPane.showMessageDialog(jf,"Academic Course with
Course ID "+ courseID +" is Already Registered");
                        }
                    }
                    i = i + 1;
```

```
                }

                if (isRegistered == false){
                    JOptionPane.showMessageDialog(jf,"Academic Course with
CourseID " + courseID + " has not been Registered Yet");
                }
            }
        }
    });

    // Adding Display Button in Academic Course
    btnAdisplay = new JButton("Display");
    btnAdisplay.setBounds(150,340,150,25);
    btnAdisplay.setFont(fb);
    jAcademicCourse.add(btnAdisplay);

    btnAdisplay.addActionListener (new ActionListener(){
        public void actionPerformed (ActionEvent e){
            if (AcademicCourseList.size() == 0){
                JOptionPane.showMessageDialog(jf,"No Course added Yet");
            }
            else{
                int i = 0;
                for (Course ac : AcademicCourseList){
                    AcademicCourse acGet = (AcademicCourse)
(AcademicCourseList.get(i));
                    acGet.display();
                    i = i + 1;
                }
            }
        }
```

```
});


//Adding Clear Button in Academic Course
btnAclear = new JButton("Clear");
btnAclear.setBounds(330,340,150,25);
btnAclear.setFont(fb);
jAcademicCourse.add(btnAclear);

btnAclear.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        txtAcourseID.setText("");
        txtAcourseName.setText("");
        txtAduration.setText("");
        txtAlevel.setText("");
        txtAcredit.setText("");
        txtAnumOfAssessment.setText("");
        txtAcourseLeader.setText("");
        txtAlecturerName.setText("");
        txtAstartingDate.setText("");
        txtAcompletionDate.setText("");
        txtAcourseID2.setText("");
    }
});

//Adding Academic Course Panel to Frame
jf.add(jAcademicCourse);

// Creating a Panel For Non-Academic Course
```

```
jNonAcademicCourse = new JPanel();
jNonAcademicCourse.setBounds(0,0,600,500);
jNonAcademicCourse.setVisible(false);
jNonAcademicCourse.setLayout(null);

// Adding Title in Non-Academic Course
lblNonAcademicCourse = new JLabel("Non Academic Course");
lblNonAcademicCourse.setBounds(200,10,300,25);
lblNonAcademicCourse.setFont(ft);
jNonAcademicCourse.add(lblNonAcademicCourse);

/* Creating all the Necessary Label and TextField for
   Non-Academic Course*/

// Adding the Course ID Label and TextField in Non-Academic Course
lblNAcourseID = new JLabel("Course ID");
txtNAcourseID = new JTextField();
lblNAcourseID.setBounds(20,50,150,20);
txtNAcourseID.setBounds(110,50,150,20);
lblNAcourseID.setFont(ff);
jNonAcademicCourse.add(lblNAcourseID);
jNonAcademicCourse.add(txtNAcourseID);

// Adding the Course Name Label and TextField in Non-Academic Course
Panel
lblNAcourseName = new JLabel("Course Name");
txtNAcourseName = new JTextField();
lblNAcourseName.setBounds(290,50,150,20);
txtNAcourseName.setBounds(410,50,150,20);
lblNAcourseName.setFont(ff);
jNonAcademicCourse.add(lblNAcourseName);
```

```
jNonAcademicCourse.add(txtNAcourseName);

// Adding the Duration Label and TextField in Non-Academic Course
lblNAduration = new JLabel("Duration");
txtNAduration = new JTextField();
lblNAduration.setBounds(25,90,150,20);
txtNAduration.setBounds(110,90,150,20);
lblNAduration.setFont(ff);
jNonAcademicCourse.add(lblNAduration);
jNonAcademicCourse.add(txtNAduration);

// Adding the Prerequisite Label and TextField in Non-Academic Course
lblNAprerequisite = new JLabel("Prerequisite");
txtNAprerequisite = new JTextField();
lblNAprerequisite.setBounds(303,90,150,20);
txtNAprerequisite.setBounds(410,90,150,20);
lblNAprerequisite.setFont(ff);
jNonAcademicCourse.add(lblNAprerequisite);
jNonAcademicCourse.add(txtNAprerequisite);

//Adding CourseID for Registration
lblNAcourseID2 = new JLabel("Course ID to Register Or Remove");
txtNAcourseID2 = new JTextField();
lblNAcourseID2.setBounds(20,285,250,20);
txtNAcourseID2.setBounds(275,285,120,20);
lblNAcourseID2.setFont(ff);
jNonAcademicCourse.add(lblNAcourseID2);
jNonAcademicCourse.add(txtNAcourseID2);

// Adding the Instructor Name and TextField in Non-Academic Course
lblNAinstructorName = new JLabel("Instructor");
```

```
lblNAinstructorName2 = new JLabel("Name");
txtNAinstructorName = new JTextField();
lblNAinstructorName.setBounds(20,160,150,20);
lblNAinstructorName2.setBounds(20,180,150,20);
txtNAinstructorName.setBounds(110,160,150,20);
lblNAinstructorName.setFont(ff);
lblNAinstructorName2.setFont(ff);
jNonAcademicCourse.add(lblNAinstructorName);
jNonAcademicCourse.add(lblNAinstructorName2);
jNonAcademicCourse.add(txtNAinstructorName);

// Adding StartDate Label And TextField for Non-Academic Course
lblNAstartDate = new JLabel("Start Date");
txtNAstartDate = new JTextField();
lblNAstartDate.setBounds(20,210,150,20);
txtNAstartDate.setBounds(110,210,150,20);
lblNAstartDate.setFont(ff);
jNonAcademicCourse.add(lblNAstartDate);
jNonAcademicCourse.add(txtNAstartDate);

// Addimg Lecturer Label And TextFlield for Non-Academic Course
lblNAcourseLeader = new JLabel("Course Leader");
txtNAcourseLeader = new JTextField();
lblNAcourseLeader.setBounds(290,210,150,20);
txtNAcourseLeader.setBounds(410,210,150,20);
lblNAcourseLeader.setFont(ff);
jNonAcademicCourse.add(txtNAcourseLeader);
jNonAcademicCourse.add(lblNAcourseLeader);

// Adding CompletionDate and TextField for Non-Academic Course
lblNAcompletionDate = new JLabel("Completion");
```

```
lblNAcompletionDate2 = new JLabel("Date");
txtNAcompletionDate = new JTextField();
lblNAcompletionDate.setBounds(300,160,150,20);
lblNAcompletionDate2.setBounds(300,180,150,20);
txtNAcompletionDate.setBounds(410,160,150,20);
lblNAcompletionDate.setFont(ff);
lblNAcompletionDate2.setFont(ff);
jNonAcademicCourse.add(lblNAcompletionDate);
jNonAcademicCourse.add(lblNAcompletionDate2);
jNonAcademicCourse.add(txtNAcompletionDate);

// Adding the Exam Date Label and TextField on Non-Academic Course
lblNAexamDate = new JLabel("Exam Date");
txtNAexamDate = new JTextField();
lblNAexamDate.setBounds(18,250,150,20);
txtNAexamDate.setBounds(110,250,150,20);
lblNAexamDate.setFont(ff);
jNonAcademicCourse.add(lblNAexamDate);
jNonAcademicCourse.add(txtNAexamDate);

//ArrayList for NonAcademicCourse
ArrayList <Course> NonAcademicCourseList = new ArrayList <Course>();

// Creating all the Required Button in Non-Academic Course

// Adding Add Button for Non-Academic Course
btnNAadd = new JButton("Add");
btnNAadd.setBounds(410,120,150,25);
btnNAadd.setFont(fb);
jNonAcademicCourse.add(btnNAadd);
```

```
        btnNAadd.addActionListener (new ActionListener(){
            public void actionPerformed (ActionEvent e){
                String courseID, courseName, prerequisite;
                int duration;
                boolean checkInt = false;
                boolean check = false;
                duration = 0;
                if (txtNAcourseID.getText().isEmpty() ||
txtNAcourseName.getText().isEmpty() || txtNAduration.getText().isEmpty() ||
txtNAprerequisite.getText().isEmpty()){
                    JOptionPane.showMessageDialog(jf,"All of The Field are
Required","Aleart",JOptionPane.WARNING_MESSAGE);
                }
                else{
                    courseID = txtNAcourseID.getText();
                    courseName = txtNAcourseName.getText();
                    prerequisite = txtNAprerequisite.getText();
                    try{
                        duration = Integer.parseInt(txtNAduration.getText());
                        checkInt = true;
                    }
                    catch(Exception ex){
                        JOptionPane.showMessageDialog(jf,"Please Enter Valid
Input","Aleart",JOptionPane.WARNING_MESSAGE);
                    }
                    for ( Course nac : NonAcademicCourseList){
                        if (nac.getCourseID().equals(courseID)){
                            check = true;
                            break;
                        }
                    }
```

```
                if (check == false && checkInt == true){
                    NonAcademicCourse nacCourse = new
NonAcademicCourse(courseID,courseName,duration,prerequisite);
                    NonAcademicCourseList.add(nacCourse);
                    JOptionPane.showMessageDialog(jf,"Non Academic Course with
Course ID "+courseID+" has Been Successfully Added.");
                }
                else if (checkInt == true){
                    JOptionPane.showMessageDialog(jf,"Non Academic Course with
Course ID "+courseID+" is Already Added");
                }
            }
        }
    });

    //Adding Register Button for Non-Academic Course
    btnNAregister = new JButton("Register");
    btnNAregister.setBounds(410,245,150,25);
    btnNAregister.setFont(fb);
    jNonAcademicCourse.add(btnNAregister);

    btnNAregister.addActionListener (new ActionListener(){
        public void actionPerformed (ActionEvent e){
            String courseID, courseLeader, instructorName, startDate,
completionDate, examDate;
            boolean isRegistered = false;
            if (txtNAcourseID2.getText().isEmpty() ||
txtNAcourseLeader.getText().isEmpty() ||
txtNAinstructorName.getText().isEmpty() || txtNAstartDate.getText().isEmpty() ||
txtNAcompletionDate.getText().isEmpty() || txtNAexamDate.getText().isEmpty()){
```

```
                JOptionPane.showMessageDialog(jf,"All Field are
Required","Aleart",JOptionPane.WARNING_MESSAGE);
            }
            else{
                courseID = txtNAcourseID2.getText();
                courseLeader = txtNAcourseLeader.getText();
                instructorName = txtNAinstructorName.getText();
                startDate = txtNAstartDate.getText();
                completionDate = txtNAcompletionDate.getText();
                examDate = txtNAexamDate.getText();
                int i = 0;
                for (Course nac : NonAcademicCourseList){
                    if (nac.getCourseID().equals(courseID)){
                        NonAcademicCourse nacGet =
(NonAcademicCourse)(NonAcademicCourseList.get(i));
                        isRegistered = true;
                        if(!nacGet.getIsRegistered()){
                            nacGet.register(courseLeader, instructorName, startDate,
completionDate, examDate);
                            JOptionPane.showMessageDialog(jf,"Non Academic Course
with Course ID "+ courseID +" is Registered Successful");
                        }
                        else{
                            JOptionPane.showMessageDialog(jf,"Non Academic Course
with Course ID " +courseID+ "  is Already Registered");
                        }
                    }
                    i = i + 1;
                }
                if (isRegistered == false){
```

```
                    JOptionPane.showMessageDialog(jf,"Non Academic Course with
CourseID "+courseID+" has not been Registered Yet");
                }
            }
        }
    });

    //Adding Remove Button for Non-Academic Course
    btnNAremove = new JButton("Remove");
    btnNAremove.setBounds(410,280,150,25);
    btnNAremove.setFont(fb);
    jNonAcademicCourse.add(btnNAremove);

    btnNAremove.addActionListener (new ActionListener(){
        public void actionPerformed (ActionEvent e){
            if (txtNAcourseID2.getText().isEmpty()){
                JOptionPane.showMessageDialog(jf,"Course ID cannot Be
Empty","Aleartt",JOptionPane.WARNING_MESSAGE);
            }
            else{
                int a=JOptionPane.showConfirmDialog(jf,"Are you sure you want to
Remove?","Remove",JOptionPane.YES_NO_OPTION);
                if(a== JOptionPane.YES_OPTION){
                String courseID = txtNAcourseID2.getText();
                Iterator<Course> itr = NonAcademicCourseList.iterator();
                boolean recordFound = false;
                while(itr.hasNext()){
                    Course c = itr.next();
                    if (c.getCourseID().equals(courseID)){
                        recordFound = true;
                        NonAcademicCourse nac = (NonAcademicCourse) c;
```

```
                    if (!nac.getIsRemoved()){
                        nac.remove();
                    }
                    itr.remove();
                    JOptionPane.showMessageDialog(jf,"Non Academic Course
with Course ID " +courseID+ " has Been Sucessful Removed");
                }
            }

            if (recordFound == false){
                JOptionPane.showMessageDialog(jf,"Course ID " +courseID+ "
Not Found");
            }
            }
        }
    }
});

// Adding Display Button in Academic Course
btnNAdisplay = new JButton("Display");
btnNAdisplay.setBounds(150,330,150,25);
btnNAdisplay.setFont(fb);
jNonAcademicCourse.add(btnNAdisplay);

btnNAdisplay.addActionListener (new ActionListener(){
    public void actionPerformed(ActionEvent e){
        String prerequisite;
        if (NonAcademicCourseList.size() == 0){
            JOptionPane.showMessageDialog(jf,"No Course Registered Yet");
        }
        else{
```

```
                int i = 0;
                for (Course nac : NonAcademicCourseList){
                    NonAcademicCourse nacGet = (NonAcademicCourse)
(NonAcademicCourseList.get(i));
                    nacGet.display();
                    prerequisite = nacGet.getPrerequisite();
                    System.out.println("The Prerequisite is: "+ prerequisite);
                    i = i + 1;
                }
            }
        }
    });

    //Adding Clear Button in Academic Course
    btnNAclear = new JButton("Clear");
    btnNAclear.setBounds(330,330,150,25);
    btnNAclear.setFont(fb);
    jNonAcademicCourse.add(btnNAclear);

    btnNAclear.addActionListener (new ActionListener() {
        public void actionPerformed( ActionEvent e){
            txtNAcourseID.setText("");
            txtNAcourseName.setText("");
            txtNAduration.setText("");
            txtNAprerequisite.setText("");
            txtNAinstructorName.setText("");
            txtNAstartDate.setText("");
            txtNAcompletionDate.setText("");
            txtNAexamDate.setText("");
            txtNAcourseID2.setText("");
            txtNAcourseLeader.setText("");
```

```
            }
        });

        // Adding NonAcademic Course Panel to Frame
        jf.add(jNonAcademicCourse);

        // Event Handling for JavaMenu
        exitApp.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e){
                int a=JOptionPane.showConfirmDialog(jf,"Are you sure you want to
exit?","Exit",JOptionPane.YES_NO_OPTION);
                if(a== JOptionPane.YES_OPTION){
                    System.exit(0);
                }
            }
        });

        nonAcademicCourse.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                jNonAcademicCourse.setVisible(true);
                jAcademicCourse.setVisible(false);
            }
        });

        academicCourse.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
```

```
            jAcademicCourse.setVisible(true);
            jNonAcademicCourse.setVisible(false);
        }
    });

    btnHacademicCourse.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
            jAcademicCourse.setVisible(true);
            jWelcometoCourse.setVisible(false);
        }
    });

    btnHnonAcademicCourse.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
            jNonAcademicCourse.setVisible(true);
            jWelcometoCourse.setVisible(false);
        }
    });

    btnHexit.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
            int a=JOptionPane.showConfirmDialog(jf,"Are you sure you want to
exit?","Exit",JOptionPane.YES_NO_OPTION);
            if(a== JOptionPane.YES_OPTION){
                System.exit(0);
            }
        }
    });

    btnHhelp.addActionListener(new ActionListener()
    {
```

```
            public void actionPerformed(ActionEvent e){
                JOptionPane.showMessageDialog(jf,"For any help on this Application
or incase of any queries Contact: np01nt4s210047@islingtoncollege.edu.np");
            }
        });

        //Adding Properties to jfResult
        jfResult.setResizable(false);
        jfResult.setLayout(null);
        jfResult.setSize(600,450);

        // Adding Properties to Frame
        jf.setResizable(false);
        jf.setLayout(null);
        jf.setSize(600,450);
        jf.setVisible(true);

    }

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

9. Appendix 2

➔ Course

```
public class Course
{
    // Defining the Four Attributes of Course Class
    private String courseID;
    private String courseName;
    private String courseLeader;
    private int duration;

    // Creating a Course Constructor which accepts 3 parameter and Assigning the
value for each attribute
    public Course (String courseID, String courseName, int duration){
        this.courseID = courseID;
        this.courseName = courseName;
        this.duration = duration;
        this.courseLeader = "";
    }

    // Creating a Accessor Method for Each Attribute
    public String getCourseID(){
        return courseID;
    }

    public String getCourseName(){
        return courseName;
    }

    public String getCourseLeader(){
        return courseLeader;
```

```
    }

    public int getDuration(){
        return duration;
    }

    // Set method to set the new name of Course Leader
    public void setCourseLeader(String courseLeader){
        this.courseLeader = courseLeader;
    }

    //Display method is made to display CourseID, CourseName, Duration and
CourseLeader if Exists.
    public void display(){
        System.out.println("ID of Course is: " + getCourseID());
        System.out.println("Name of Course is: " + getCourseName());
        System.out.println("Duration to Complete Course is: " + getDuration());
        if(courseLeader != "")
            System.out.println("Name of Course Leader is: " + getCourseLeader());
        else
            System.out.println("There is no current Course Leader");
    }
}
```

➔ Academic Course

```
public class AcademicCourse extends Course
{
    // Defining the Seven Attributes of Academic Course
    private String lecturerName;
    private String level;
    private String credit;
```

```
    private String startingDate;
    private String completionDate;
    private int numberOfAssessment;
    private boolean isRegistered;

    /** The AcademicCourse Constructor accepts six parameters where 3
parameters are
        assigned to Course Class and stores the assigned value to seven
parameters */

    public AcademicCourse(String courseID, String courseName, int duration,
String level, String credit,int numberOfAssessment){
        super(courseID, courseName, duration);
        this.level = level;
        this.credit = credit;
        this.numberOfAssessment = numberOfAssessment;
        this.lecturerName = "";
        this.startingDate = "";
        this.completionDate = "";
        this.isRegistered = false;
    }

    // Creating a Accessor Method for Each Attribute
    public String getLecturerName(){
        return lecturerName;
    }

    public String getLevel(){
        return level;
    }
```

```
    public String getCredit(){
        return credit;
    }

    public String getStartingDate(){
        return startingDate;
    }

    public String getCompletionDate(){
        return completionDate;
    }

    public int getNumberOfAssessment(){
        return numberOfAssessment;
    }

    public boolean getIsRegistered(){
        return isRegistered;
    }

    // Creating a Setter Method for Lecuture name and Number of Assessment
    public void setLectureName(String lecturerName){
        this.lecturerName = lecturerName;
    }

    public void setNumberOfAssessment(int numberOfAssessment){
        this.numberOfAssessment = numberOfAssessment;
    }

    /** The Method register accepts Four parameter to register the academic
course.If the coure exists then Instructor Name, Starting Date and Completion
```

Date is Printed. Where as if course is not registered yet, All of Four parament is stored with their Respective Attributes And CourseLeader Name is set in setCourseLeader Method of Course Class*/

```
    public void register(String courseLeader, String lecturerName, String
startingDate, String completionDate){
        if(isRegistered == true)
        {
            System.out.println("Name of Instructor: " + getLecturerName());
            System.out.println("Starting Date is " + getStartingDate());
            System.out.println("Completion Date is " + getCompletionDate());
        }
        else
        {
            super.setCourseLeader(courseLeader);
            this.lecturerName = lecturerName;
            this.startingDate = startingDate;
            this.completionDate = completionDate;
            this.isRegistered = true;
        }
    }
```

/** Display Method is used to display the Detial of Course. Display Method is called from Course Class which displays CourseID, CourseName, Duration and CourseLeader if Exists. If the Course is Registered, Details of Course are Displayed. */

```
    public void display(){
        super.display();
        if (isRegistered == true){
            System.out.println("The Name of Lecturer is: " + getLecturerName());
```

```
            System.out.println("The level is: " + getLevel());
            System.out.println("The Credit is: " + getCredit());
            System.out.println("The Starting Date of Course is: " + getStartingDate());
            System.out.println("The Completion Date is: " + getCompletionDate());
            System.out.println("Number of Assessment is: " +
getNumberOfAssessment());
        }
    }
}
```

➔ Non-Academic Course

```
public class NonAcademicCourse extends Course
{
    // Defining the Seven Attributes of Academic Course
    private String instructorName;
    private String startDate;
    private String completionDate;
    private String examDate;
    private String prerequisite;
    private boolean isRegistered;
    private boolean isRemoved;

    /** The AcademicCourse Constructor accepts four parameters in which 3
parameters are assigned
        to Course Class and stores the assigned value to six parameters */

    public NonAcademicCourse(String courseID, String courseName, int duration,
String prerequisite){
        super(courseID, courseName, duration);
        this.prerequisite = prerequisite;
        this.startDate = "";
```

```
        this.completionDate = "";
        this.examDate = "";
        this.isRegistered = false;
        this.isRemoved = false;
    }

    // Creating the Accessor method for Each Attribute
    public String getInstructorName(){
        return instructorName;
    }

    public String getStartDate(){
        return startDate;
    }

    public String getCompletionDate(){
        return completionDate;
    }

    public String getExamDate(){
        return examDate;
    }

    public String getPrerequisite(){
        return prerequisite;
    }

    public boolean getIsRegistered(){
        return isRegistered;
    }
```

```
    public boolean getIsRemoved(){
        return isRemoved;
    }

    // Setter Method is Created to Register the Instructor Name if not registered
yet.
    public void setInstructorName(String instructorName){
        if(isRegistered == false){
            this.instructorName = instructorName;
        }
        else{
            System.out.println("The Instructor Name Cannot be Changed");
        }
    }

    /** The register Method Accepts five parameters to set the detial of courses
        If the Course detail is already registered, appropriate message is Displayed.
*/

    public void register(String courseLeader, String instructorName, String
startDate, String completionDate, String examDate){
        if (isRegistered == false){
            setInstructorName(instructorName);
            super.setCourseLeader(courseLeader);
            this.startDate = startDate;
            this.completionDate = completionDate;
            this.examDate = examDate;
            this.isRegistered = true;
        }
        else{
            System.out.println("The Course is already Registered");
```

```
        }
    }

    // Remove Method is used to Removed all the details of Course by Replacing it
with "" if not Removed yet..
    public void remove(){
        if (isRemoved == true){
            System.out.println("The course has been Already Removed");
        }
        else{
            super.setCourseLeader("");
            this.instructorName = "";
            this.startDate = "";
            this.completionDate = "";
            this.examDate = "";
            this.isRegistered = false;
            this.isRemoved = true;
        }
    }

    /** Display Method is used to display the Detial of Course.
        Display Method is called from Course Class which displays  CourseID,
CourseName, Duration and CourseLeader if Exists.
        If the Course is Registered, Details of Course are Displayed such as
Instructor Name, Start, Completion and Exam Date. */

    public void display(){
        super.display();
        if (isRegistered == true){
            System.out.println("The Name of Instructor is: " + getInstructorName());
            System.out.println("The Starting Date of Course is: " + getStartDate());
```

```
            System.out.println("The Completion Date of Course is: " +
getCompletionDate());
            System.out.println("The Exam Date is: " + getExamDate());
        }
    }
}
```

10. References

Alex, 2020. *Syntax and Semantic Errors.* [Online]
Available at: https://www.learncpp.com/cpp-tutorial/syntax-and-semantic-errors/#:~:text=A%20semantic%20error%20occurs%20when,2
[Accessed 19 08 2021].

Christensson P., 2012. *Syntax Error Defination.* [Online]
Available at: https://techterms.com/definition/syntax_error
[Accessed 19 08 2021].

Guru 99 , 2021. *What is Java? Definition, Meaning & Features of Java Platforms.* [Online]
Available at: https://www.guru99.com/java-platform.html
[Accessed 18 08 2021].

The Economic Times, n.d. *Definitin of 'Pseudocode'.* [Online]
Available at: https://economictimes.indiatimes.com/definition/pseudocode
[Accessed 19 08 2021].