LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CC4001NI Programming

COURSEWORK-2

Assessment Weightage & Type

30% Individual Coursework

Semester and Year

Spring 2021

Student Name: Sarthak Bikram Rana

Group: N1

London Met ID: 20049228

College ID: NP01NT4S210129

Assignment Due Date: August 20,2021

Assignment Submission Date: August 20,2021

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

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Introduction to the topic

Java is a general-purpose, object-oriented programming language focused on classes that is designed to have less implementation dependencies. It is a computing medium for the advancement of applications. As a result, Java is fast, stable, and dependable. It's commonly used in notebooks, data centers, game consoles, science supercomputers, mobile phones, and other places to build Java applications. (Guru99, 2021)

This is the second coursework in the "Programming" module. The main goal of this coursework is to add a class to the project that we developed for the first part of the coursework to make a new class INGCollege where we made a graphical user interface (GUI) for a system which stores details of Course that includes both academic and non-academic course.

The GUI was created using the Java Programming language and the AWT and Swing APIs to accept data from the user, read the data, store the data entered, and display the data stored. This GUI represents a registration form with text fields for entering data, text field labels, an add button for adding course details, a register button for registering details of academic and non-academic courses, a remove button for non-academic courses, a clear button for removing details of a course with a specific Course ID, and a display button for displaying all records which have been entered in academic course and non-academic course respectively.

This coursework is done using different applications like BlueJ, Draw.io and MS - Word.

2. Class Diagram

2.1 Introduction

Class diagrams illustrate characteristics, processes, and relationships between classes to explain structures. They operate on the basis of object orientation principles. The interaction between objects is defined by this orientation. With the help of class diagrams, we can generate models with attributes, relationships, operations, and intersections. A class diagrams show the relationships between classes through aggregations and associations, as well as the transmission of properties and behaviour between classes. It is mostly important in software development. Class diagrams are the most effective way to depict a system's structure in detail, displaying its attributes, operations, and inter-relationships. (MicroTool, 2020)

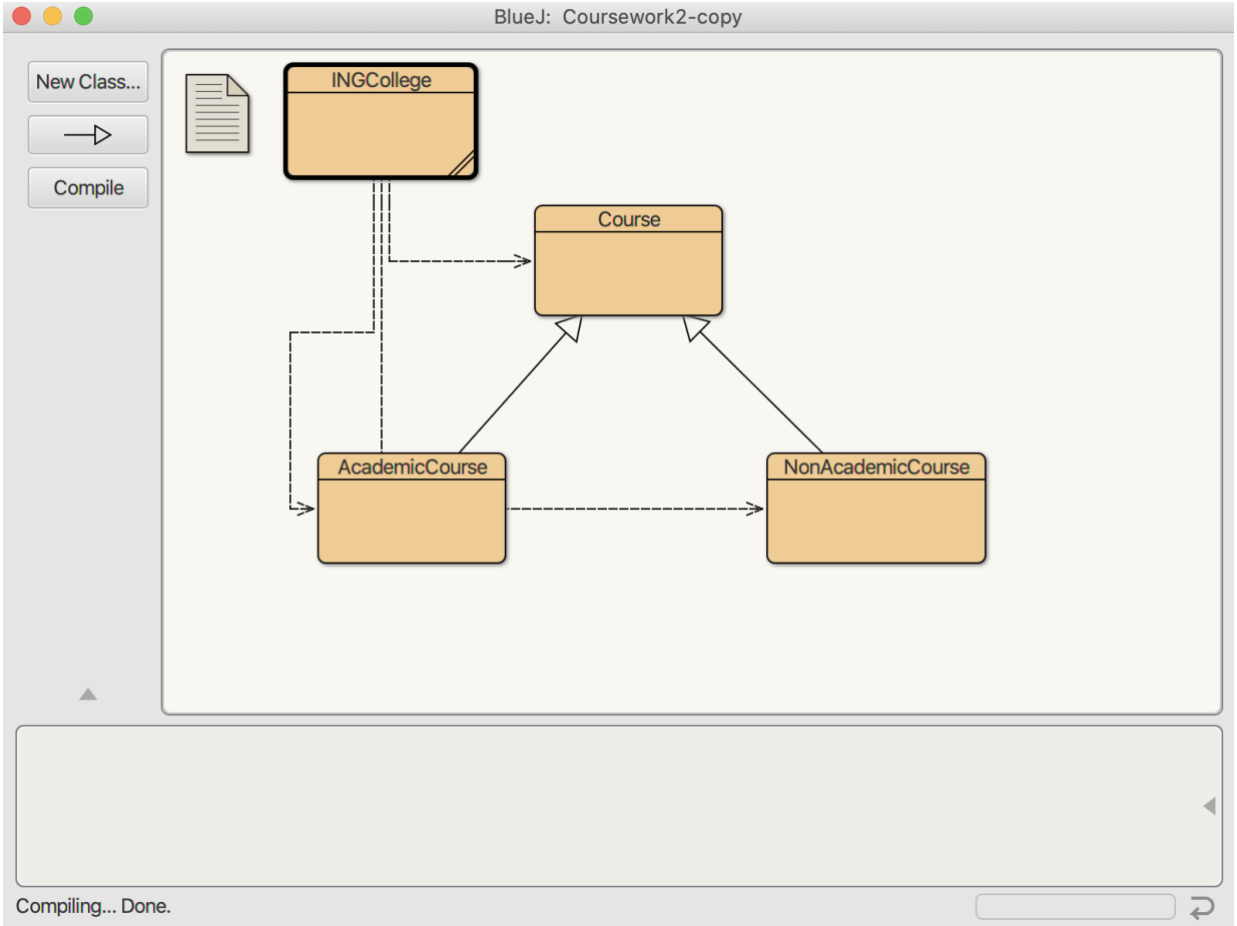

Figure 1: Class diagram of classes in BlueJ

INGCollege

- jf: JFrame
- Academic_display_jf: JFrame
- NAcademic_display_jf: JFrame
- WPjpane: JPanel
- ACjpane: Jpanel
- NCjpane: Jpanel
- lbl_WP: JLabel
- lbl_WP_to: JLabel
- lbl_WP_m: JLabel
- lbl_WP_f: JLabel
- lbl_WP_CT: JLabel
- lbl_pb: JLabel
- lbl_WP_Academic: JLabel
- lbl_Academic_ID: JLabel
- lbl_Academic_Name: JLabel
- lbl_Academic_Duration: JLabel
- lbl_Academic_Level: JLabel
- lbl_Academic_Credit: JLabel
- lbl_Academic_NOA: JLabel
- lbl_Academic_Leader: JLabel
- lbl_Academic_Lecturer: JLabel
- lbl_Academic_SDate: JLabel
- lbl_Academic_CDate: JLabel
- lbl_NAcademic: JLabel
- lbl_NAcademic_ID: JLabel
- lbl_NAcademic_Name: JLabel
- lbl_NAcademic_Prerequisite: JLabel
- lbl_NAcademic_Duration: JLabel
- lbl_NAcademic_Leader: JLabel
- lbl_NAcademic_Instructor: JLabel
- lbl_NAcademic_EDate: JLabel
- lbl_NAcademic_SDate: JLabel
- lbl_NAcademic_CDate: JLabel
- txt_Academic_ID: JTextField
- txt_Academic_Name: JTextField
- txt_Academic_Duration: JTextField
- txt_Academic_Level: JTextField
- txt_Academic_Credit: JTextField
- txt_Academic_NOA: JTextField
- txt_Academic_Leader: JTextField
- txt_Academic_Lecturer: JTextField
- txt_Academic_SDate: JTextField
- txt_Academic_CDate: JTextField
- txt_NAcademic_ID: JTextField
- txt_NAcademic_Name: JTextField
- txt_NAcademic_Prerequisite: JTextField
- txt_NAcademic_Duration: JTextField
- txt_NAcademic_Leader: JTextField
- txt_NAcademic_Instructor: JTextField
- txt_NAcademic_EDate: JTextField
- txt_NAcademic_SDate: JTextField
- txt_NAcademic_CDate: JTextField
- btn_WP_AC: JButton
- btn_WP_NC: JButton
- btn_Academic_Add: JButton
- btn_Academic_Register: JButton
- btn_NAcademic: JButton
- btn_Academic_Clear: JButton
- btn_Academic_Display: JButton
- btn_NAcademic_add: JButton
- btn_NAcademic_Register: JButton
- btn_NAcademic_Remove: JButton
- btn_Academic: JButton
- btn_NAcademic_Clear: JButton
- btn_NAcademic_Display: JButton
- fnt1: Font
- fnt2: Font
- fnt3: Font
- fnt4: Font
- ing: Icon
- islington: Icon
- Academic_table: JTable
- NAcademic_table: JTable
- Academic_table_model: DefaultTableModel
- NAcademic_table_model: DefaultTableModel
- academicCourseList: ArrayList<Course>
- nonAcademicCourseList: ArrayList<Course>

+<<constructor>>INGCollege()
+ actionPerformed(ActionEvent e)
+ register(courseLeader_Academic, lecturerName_Academic, startingDate_Academic, completionDate_Academic)
+ register(courseLeader_NAcademic, instructorrName_NAcademic, startingDate_NAcademic, completionDate_NAcademic, examDate_NAcademic)
+ remove()
+ main(String [] args)

Figure 2: Class Diagram of INGCollege Class

3. Pseudocode

3.1 Introduction

Pseudocode (pronounced SOO-doh-kohd) is a comprehensive and understandable explanation of what a computer program or algorithm must do that is written in a formally styled natural language rather than a programming language. Pseudocode is sometimes used as a comprehensive phase in the development of a software. It enables developers or lead programmers to articulate the concept in great detail and gives programmers a comprehensive template for writing code in a specific programming language in the next step. Since pseudocode is complex but readable, it can be examined by a team of designers and programmers to ensure that actual programming matches design requirements. Finding mistakes early in the development process is less expensive than catching them later. If the pseudocode has been approved, it is rewritten in a programming language's vocabulary and syntax. Pseudocode is sometimes used in conjunction with methodologies based on computer-aided software engineering. Programs can be written to convert a given pseudocode language into a given programming language. (TechTarget Contributor, 2005)

3.2 Pseudocode for INGCollege Class

```
IMPORT packages in program

    CREATE class INGCollege

    DEFINE instance variables of the UI components

        CREATE constructor

        DEFINE Frame
            DECLARE PRIVATE jf, Academic_Display_jf,
            NAcademic_display_jf

        DEFINE Panel
            DECLARE PRIVATE WPjpane, ACjpane, NCjpane

        DEFINE Label
            DECLARE PRIVATE lbl_WP, lbl_WP_to, lbl_WP_m, lbl_WP_f,
            lbl_WP_CT, lbl_pb, lbl_Academic, lbl_Academic_ID,
            lbl_Academic_Name, lbl_Academic_Duration, lbl_Academic_Level,
            lbl_Academic_Credit, lbl_Academic_NOA, lbl_Academic_Leader,
            lbl_Academic_Lecturer, lbl_Academic_SDate, lbl_Academic_CDate,
            lbl_NAcademic, lbl_NAcademic_ID, lbl_NAcademic_Name,
            lbl_NAcademic_Prerequiste,l bl_NAcademic_Duration,
            lbl_NAcademic_Leader, lbl_NAcademic_Instructor,
            lbl_NAcademic_EDate, lbl_NAcademic_SDate,
            lbl_NAcademic_CDate;

        DEFINE Text Field
            DECLARE PRIVATE txt_Academic_ID, txt_Academic_Name,
            txt_Academic_Duration, txt_Academic_Level, txt_Academic_Credit,
            txt_Academic_NOA, txt_Academic_Leader, txt_Academic_Lecturer,
            txt_Academic_SDate, txt_Academic_CDate, txt_NAcademic_ID,
            txt_NAcademic_Name, txt_NAcademic_Prerequiste,
            txt_NAcademic_Duration, txt_NAcademic_Leader,
            txt_NAcademic_Instructor, txt_NAcademic_EDate,
            txt_NAcademic_SDate, txt_NAcademic_CDate;
```

DEFINE Button
DECLARE PRIVATE btn_WP_AC, btn_WP_NC, btn_Academic_add, btn_Academic_Register, btn_NAcademic, btn_Academic_Clear, btn_Academic_Display, btn_NAcademic_add, btn_NAcademic_Register, btn_NAcademic_Remove, btn_Academic, btn_NAcademic_Clear, btn_NAcademic_Display;

DEFINE Font
DECLARE PRIVATE fnt1, fnt2, fnt3, fnt4;

DEFINE Icon
DECLARE PRIVATE ing, islington;

DEFINE Table
DECLARE PRIVATE Academic_tabel, NAcademic_tabel;

DEFINE Default Table Model
DECLARE PRIVATE Academic_table_model, NAcademic_table_model;

DEFINE ArrayList of Course type
DECLARE PRIVATE academicCourseList, nonAcademicCourseList;

CREATE a constructor for INGCollege class

CREATE frame Course

CREATE panel WPjpane

CREATE panel ACjpane

CREATE panel NCjpane

CLICK btn_Academic Action Performed
SET Visible true for ACjpane
SET Visible false for NCjpane
SET Visible false for WPjpane

CLICK btn_NAcademic Action Performed

SET Visible false for ACjpane
SET Visible true for NCjpane
SET Visible false for WOjpane

CLICK btn_WP_AC Action Performed
SET Visible true for ACjpane
SET Visible false for NCjpane
SET Visible false for WPjpane

CLICK btn_WP_NC Action Performed
SET Visible false for ACjpane
SET Visible true for NCjpane
SET Visible false for WPjpane

CLICK btn_Academic_add Action Performed
IF cousrseID, courseName, level_Academic, credit_Academic is Empty
THEN text field empty message
ELSE ADD all the values of AcademicCourseList

CLICK btn_NAcademic_add Action Performed
IF courseID, courseName, prerequisite is Empty
THEN text field empty message
ELSE ADD all the values of NonAcademicCourseList

CLICK btn_Academic_Register Action Performed
IF courseLeader_Academic, lecturerName_Academic, startingDate_Academic, completionDate_Academic is Empty
THEN text field empty message
ELSE
IF arraylist courseID equal txt_Academic_ID
THEN Call method of AcademicCourseClass Register
ELSE ID doesn't match message

CLICK btn_NAcademic_Register Action Performed
IF courseLeader, instructorName, startingDate, completionDate is Empty

THEN text field empty message
ELSE
IF arraylist courseID equal txt_NAcademic_ID
THEN Call method of NonAcademicCourseClass Register
ELSE ID doesn't match message

CLICK btn_NAcademic_Remove Action Performed
IF courseLeader_NAcademic, instructorName, startingDate, completionDate, examDate is Empty
THEN text field empty message
ELSE
IF arraylist courseID equal txt_NAcademic_ID
THEN Call method of NonAcademicCourseClass Remove
ELSE ID doesn't match message

CLICK btn_Academic_Clear Action Performed
CLEAR all the text field of Academic Course Class

CLICK btn_NAcademic_Clear Action Performed
CLEAR all the text field of Non Academic Course Class

CLICK btn_Academic_Display Action Performed
DISPLAY all the data added or registered in Academic Course Class

CLICK btn_NAcademic_Display Action Performed
DISPLAY all the data added or registered in Non Academic Course Class

CREATE main Method
PASS new INGCollege
END main Method

END constructor INGCollege class

4. Method Description

Different methods have been used in this program. This program consists of three different classes which has used different methods. In the child class, various methods from the parent class have been used.

4.1 INGCollege Class

Different methods are used in the INGCollege class which are given bellow:

- actionPerformed(ActionEvent e)

  An action listener in Java is a class that deals with all action events, such as when a user clicks on a component. This is an action listener constructor method. (ActionEvent e) is a class, and e is an instance of that class. Its primary job in the program is to invoke the methods and properties of the program. When the button is clicked in this application, the function of the button is triggered. It is used in both academic and non-academic GUI courses to add, register, remove, and clear input values of text fields, as well as to display the entered data.

- + register(courseLeader_Academic, lecturerName_Academic, startingDate_Academic, completionDate_Academic)

  The register button is part of the program's constructor method. This method accepts course leader, lecturer name, starting date, and completion date of academic course as parameters and runs the register methods with those parameters.

- + register(courseLeader_NAcademic, instructorrName_NAcademic, startingDate_NAcademic, completionDate_NAcademic, examDate_NAcademic)

  The register button is part of the program's constructor method. This method accepts course leader, instructor name, starting date, and completion date of non-academic course as parameters and runs the register methods with those parameters.

- + remove()

The remove method is part of the program's constructor method. This method removes course leader, lecturer name, starting date, and completion date if registered in non-academic course.

- + main(String [] args)

It is the Java's main method. A new constructor class INGCollege is created within this main method.

5. Testing

5.1 Test 1:

- **Test that the program can be compiled and run using the command prompt**

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

Test No:	1
Objective:	Compile Using Terminal
Action:	>> Find INGCollege.java file in command prompt >> javac INGCollege.java >> java INGCollege.java
Expected Result:	GUI should display.
Actual Result:	GUI was displayed.
Conclusion	The test is successful.

```
sarthakrana@Sarthaks-MacBook-Pro Coursework2-copy % javac INGCollege.java
sarthakrana@Sarthaks-MacBook-Pro Coursework2-copy % java INGCollege.java
```

Figure 3: Screenshot of Code assigned in Terminal

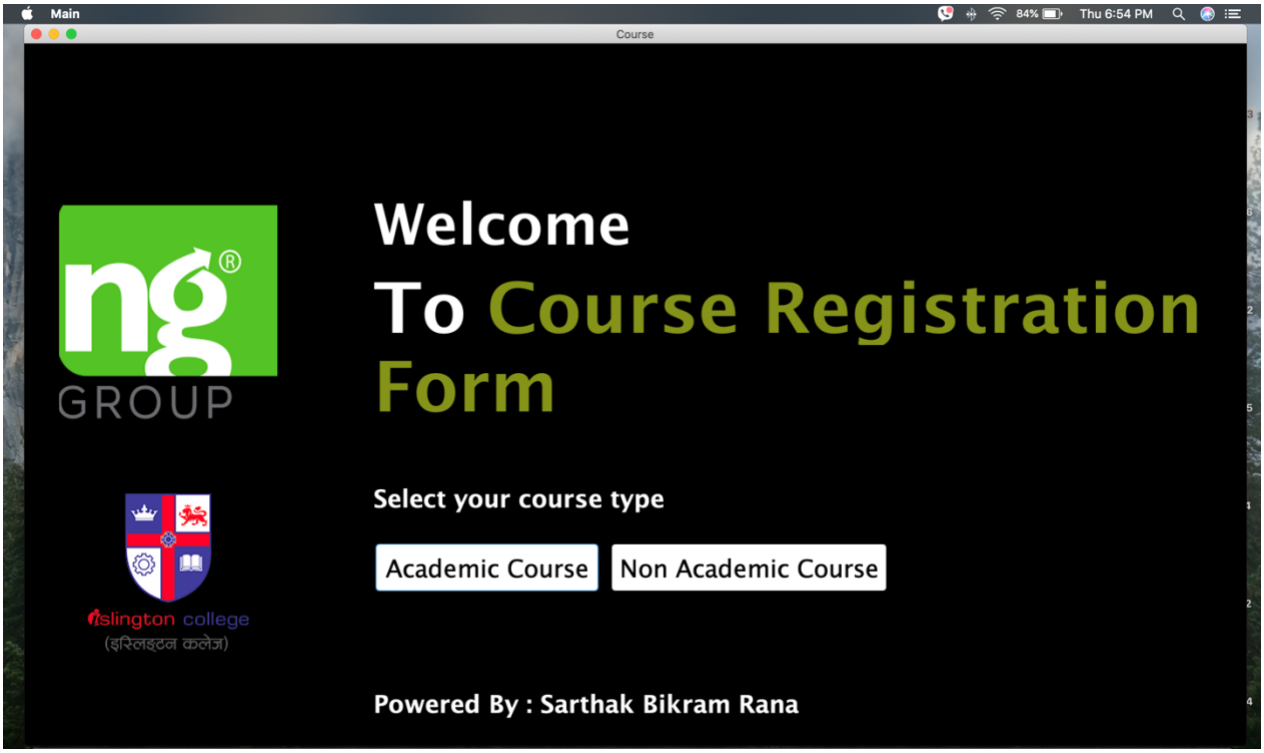

Figure 4: Screenshot of Welcome Page showed form terminal

5.2 Test 2:

- **Evidences should be shown of:**

5.2.a. Add course for Academic Course.

Table 2: To add course for Academic Course

Test No:	2.a
Objective:	Add course for Academic Course.
Action:	>> Assign values in Course ID, Course Name, Duration, Level, Credit, Number of Assesments. courseID = "322623ac" courseName = "Networking" duration = 4 level = "4" credit = "10" numberOfAssments = 5 >>Click on Add button >>Click on Display button
Expected Result:	"All of your records have been added" dialogue box should display.
Actual Result:	"All of your records have been added" dialogue box was displayed.
Conclusion	The test is successful.

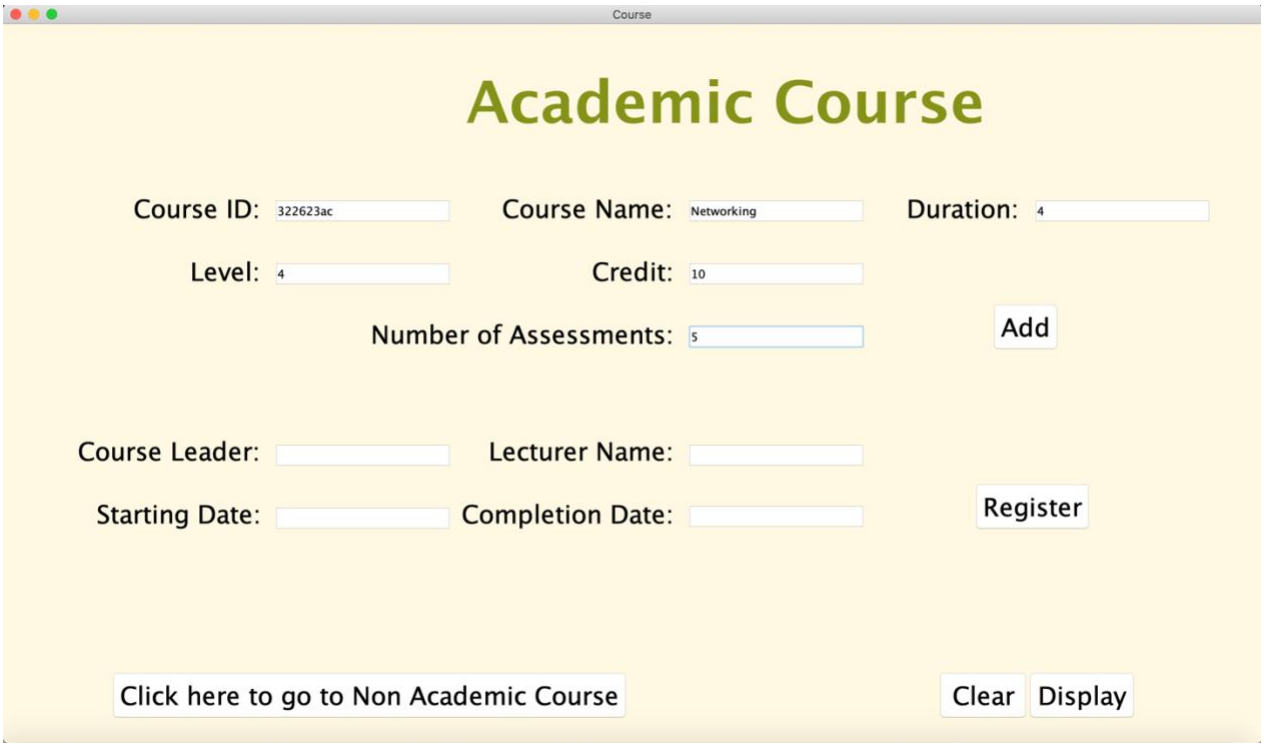

Figure 5: Screenshot of entering values in text field of Academic Course

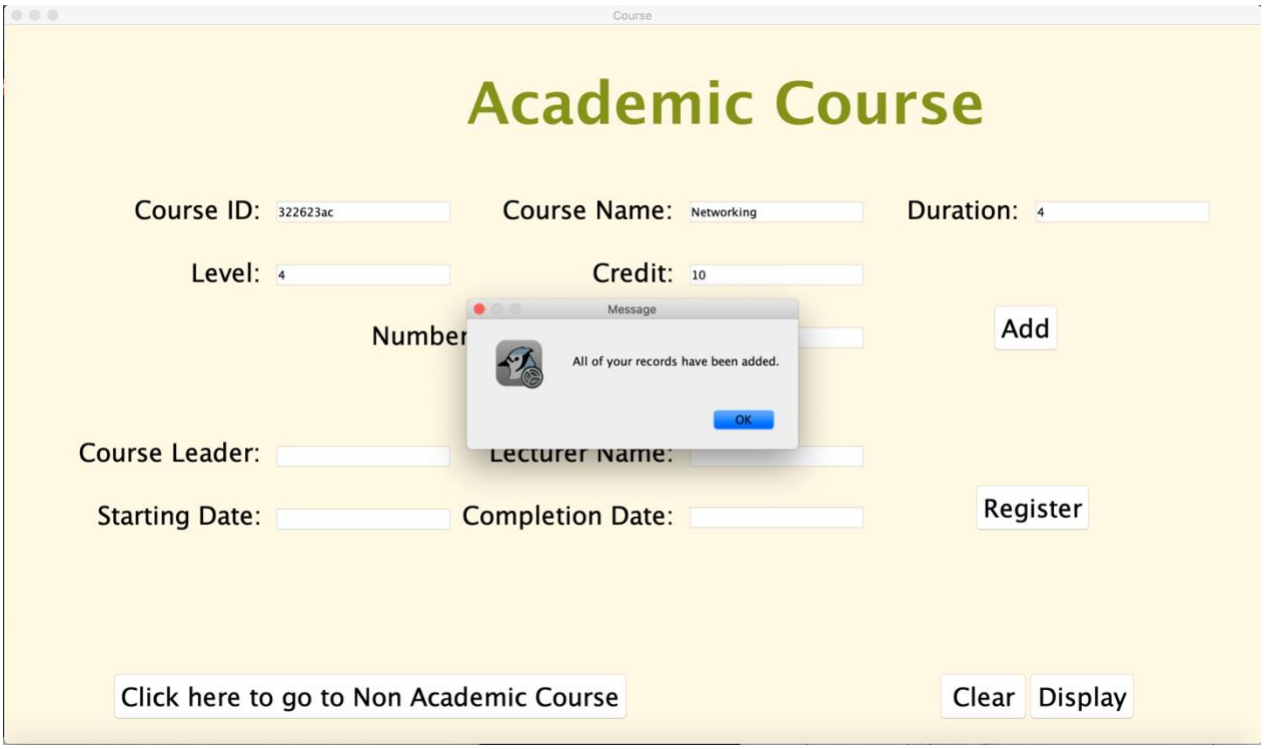

Figure 6: Screenshot of dialog box when clicking add button

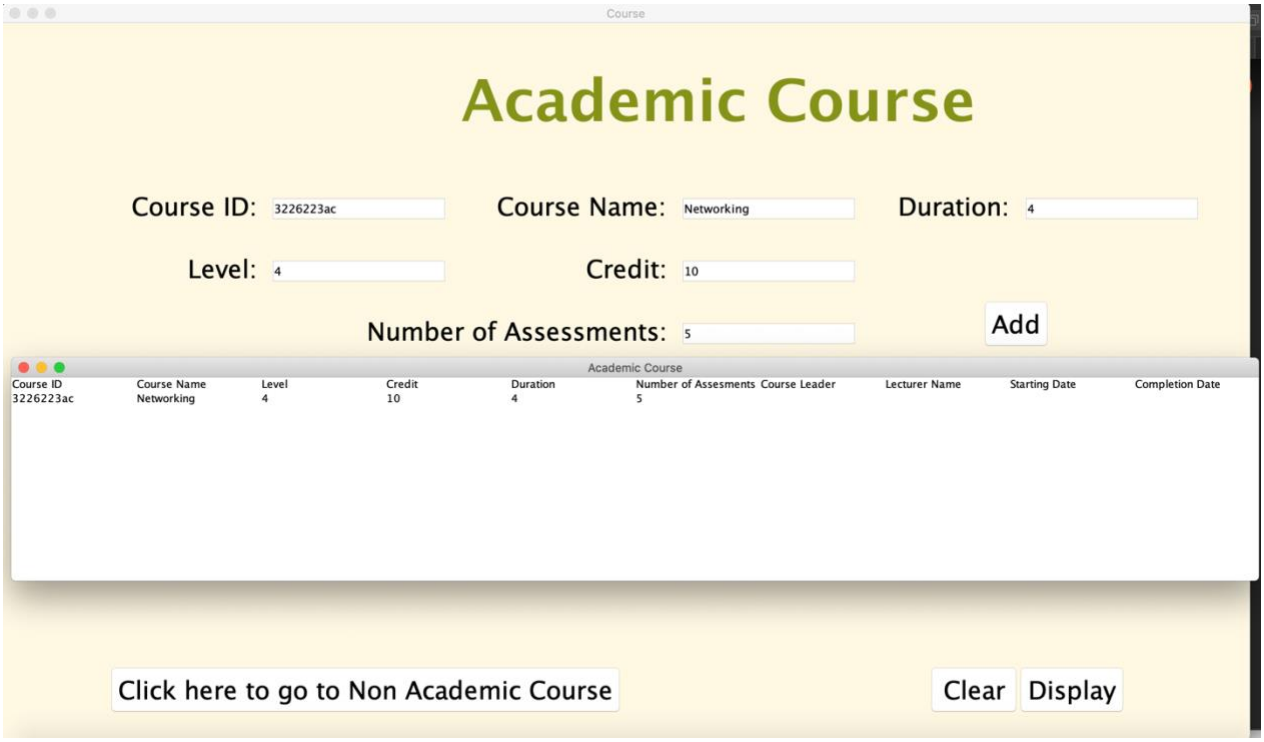

Figure 7: Screenshot of display frame which only includes data entered in Academic course while clicking add button

5.2.b. Add course for Non-Academic Course.

Table 3: To add course for Non-Academic Course.

Test No:	2.b
Objective:	Add course for Non-Academic Course.
Action:	>> Assign values in Course ID, Course Name, Prerequisites, Duration. courseID = "527253cd" courseName = "Animation" prerequisites = "B or above in Animation" duration = 4 >>Click on Add button >>Click on Display button
Expected Result:	"All of your records have been added" dialogue box should display.
Actual Result:	"All of your records have been added" dialogue box was displayed.
Conclusion	The test is successful.

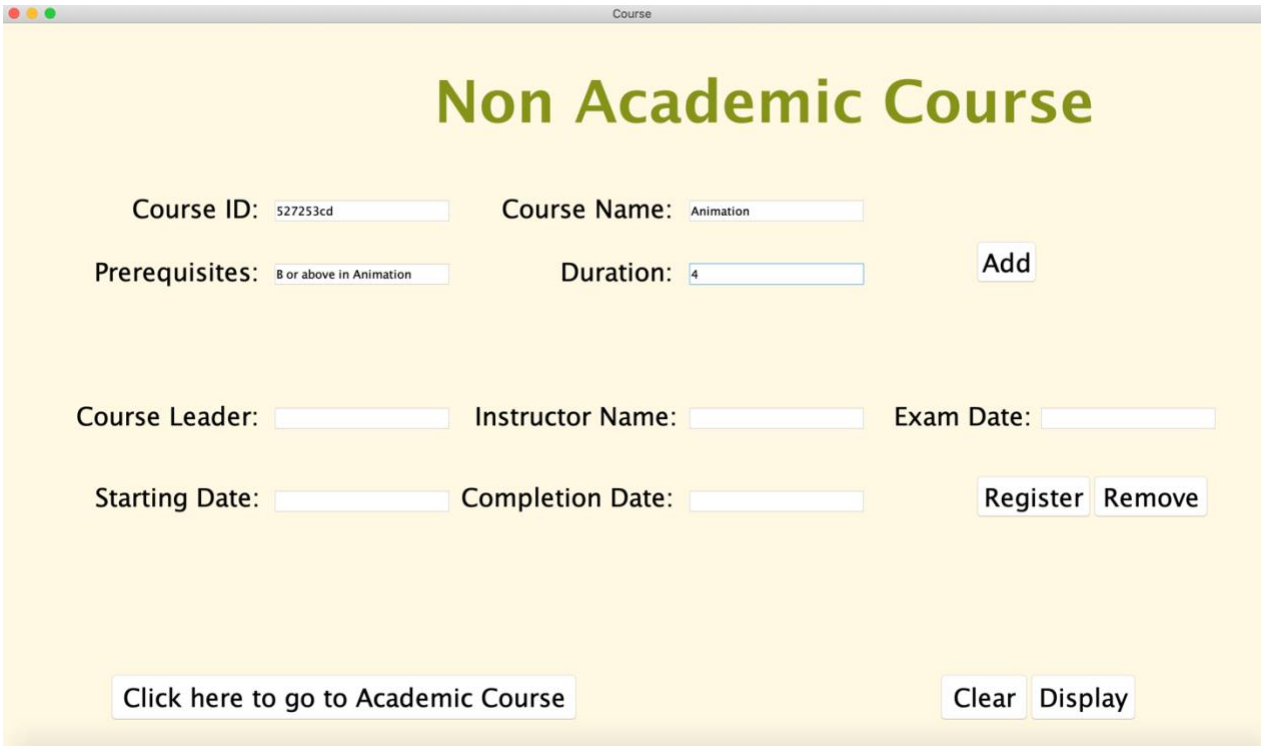

Figure 8: Screenshot of entering values in text field of Non-Academic Course

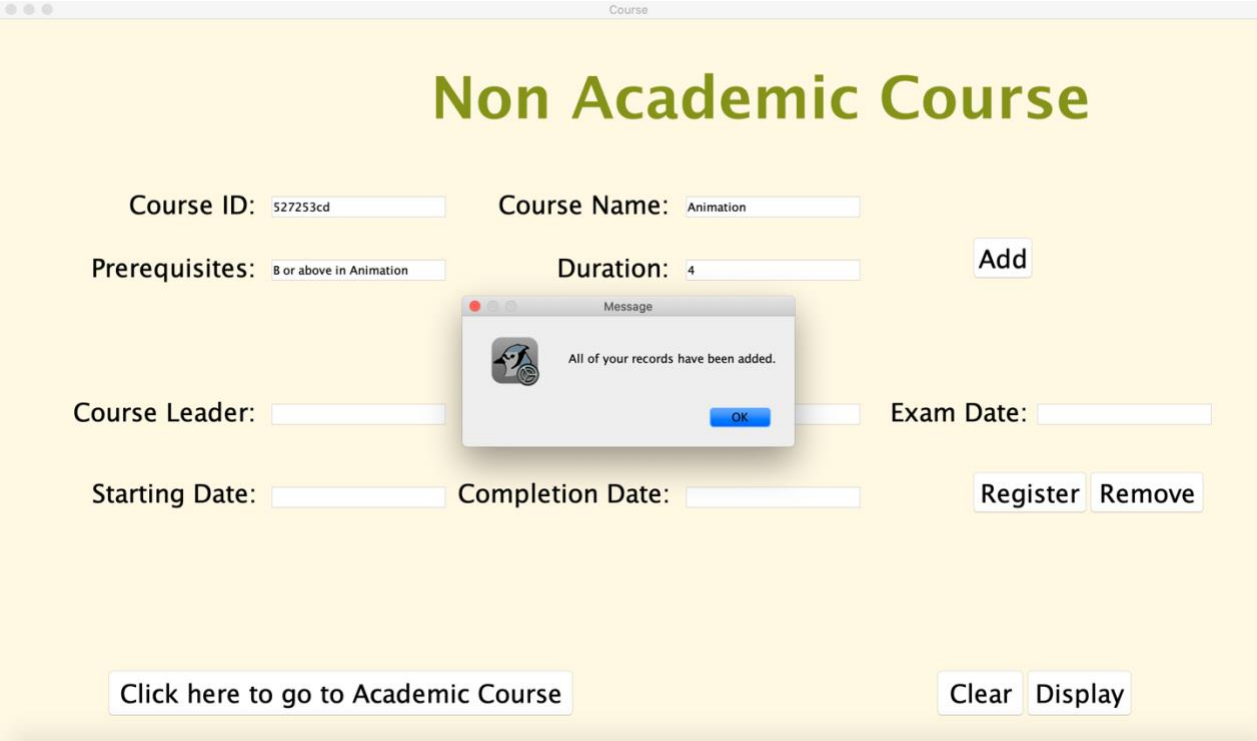

Figure 9: Screenshot of dialog box when clicking add button

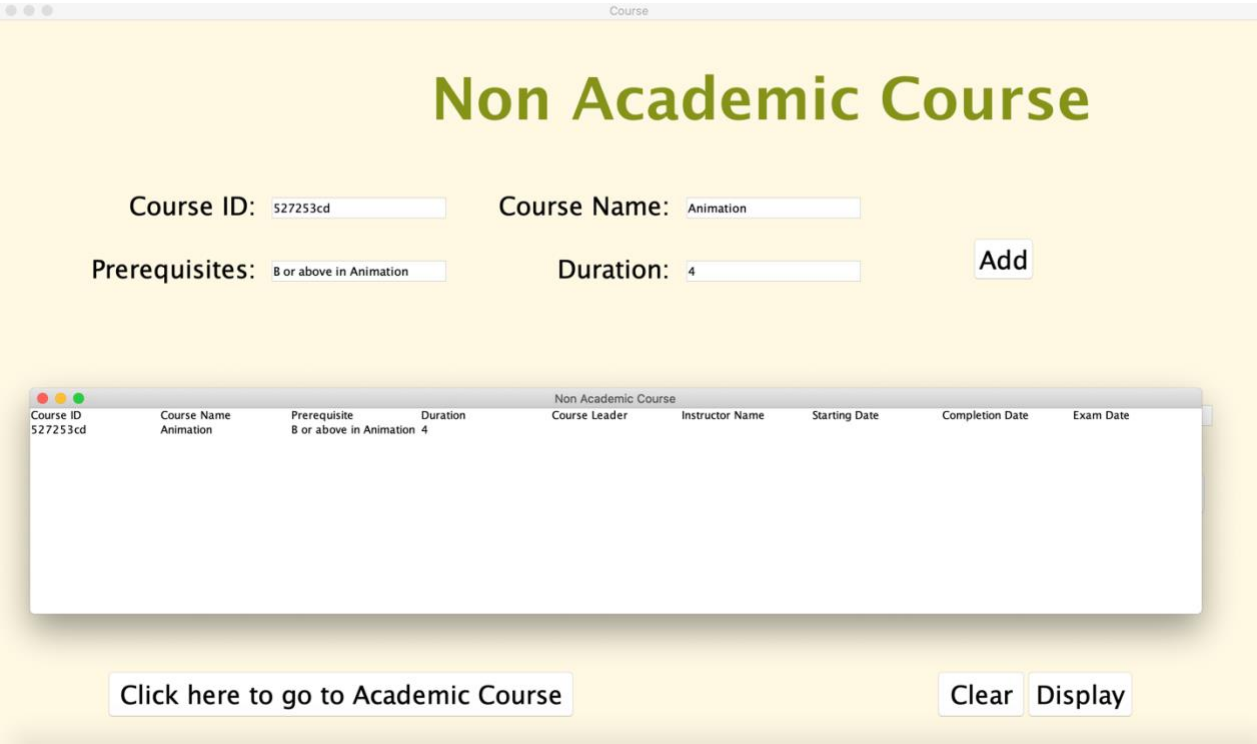

Figure 10: Screenshot of display frame which only includes data entered in Non-Academic course while clicking add button

5.2.c. Register academic course.

Table 4: To register Academic Course

Test No:	2.c
Objective:	Register Academic Course.
Action:	>> Assign values in Course Leader, Lecturer Name, Starting Date, Completion Date. courseLeader = "Steve" lecturerName = "Nash" startingDate = "March" completionDate = "July" >>Click on Register button >>Click on Display button
Expected Result:	Should display "Academic Course is registered" dialog box and all records in a new frame.
Actual Result:	"Academic Course is registered" dialog box and display frame was displayed.
Conclusion	The test is successful.

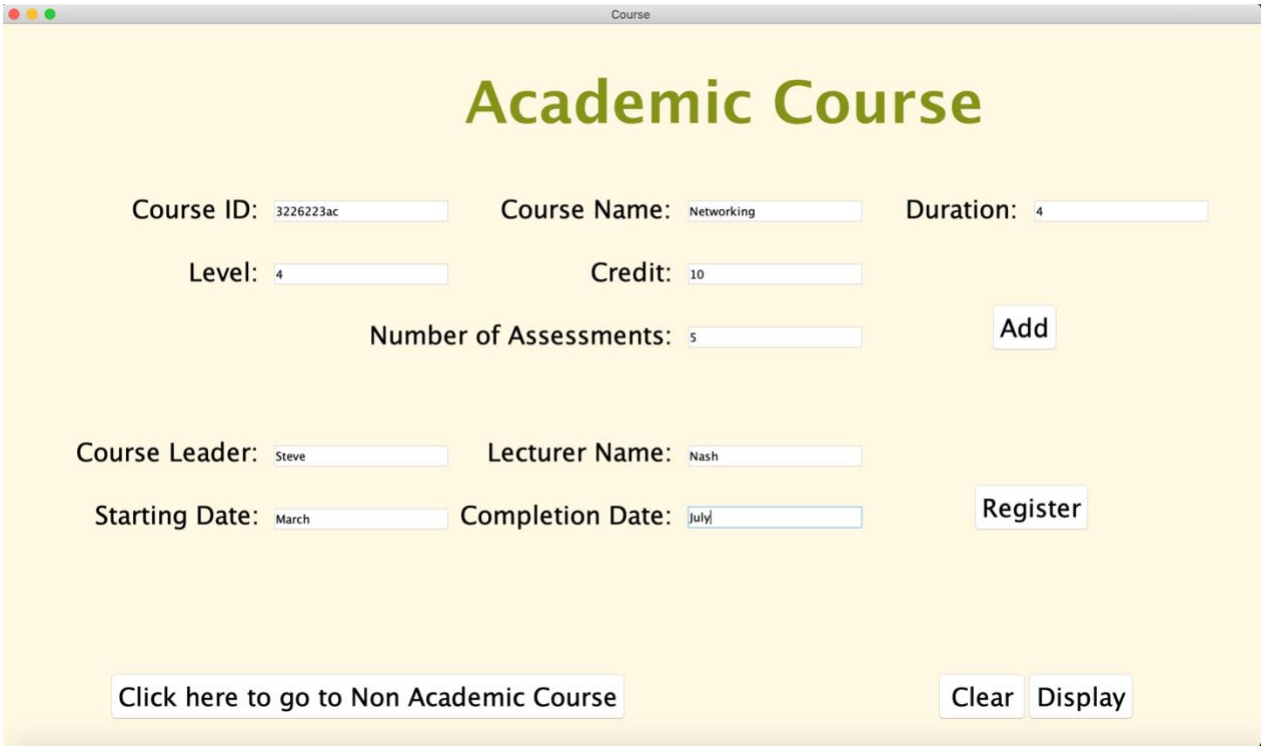

Figure 11: Screenshot of entering values in text field of Academic Course

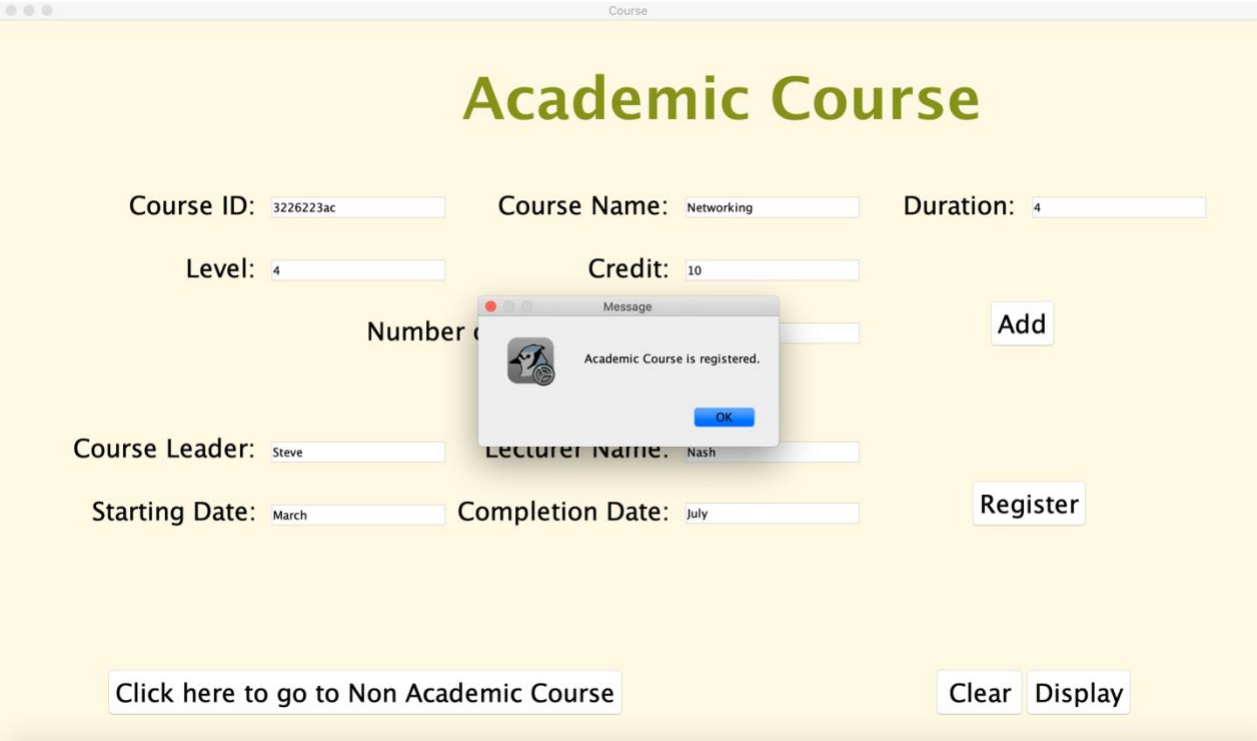

Figure 12: Screenshot of dialog box when clicking register button

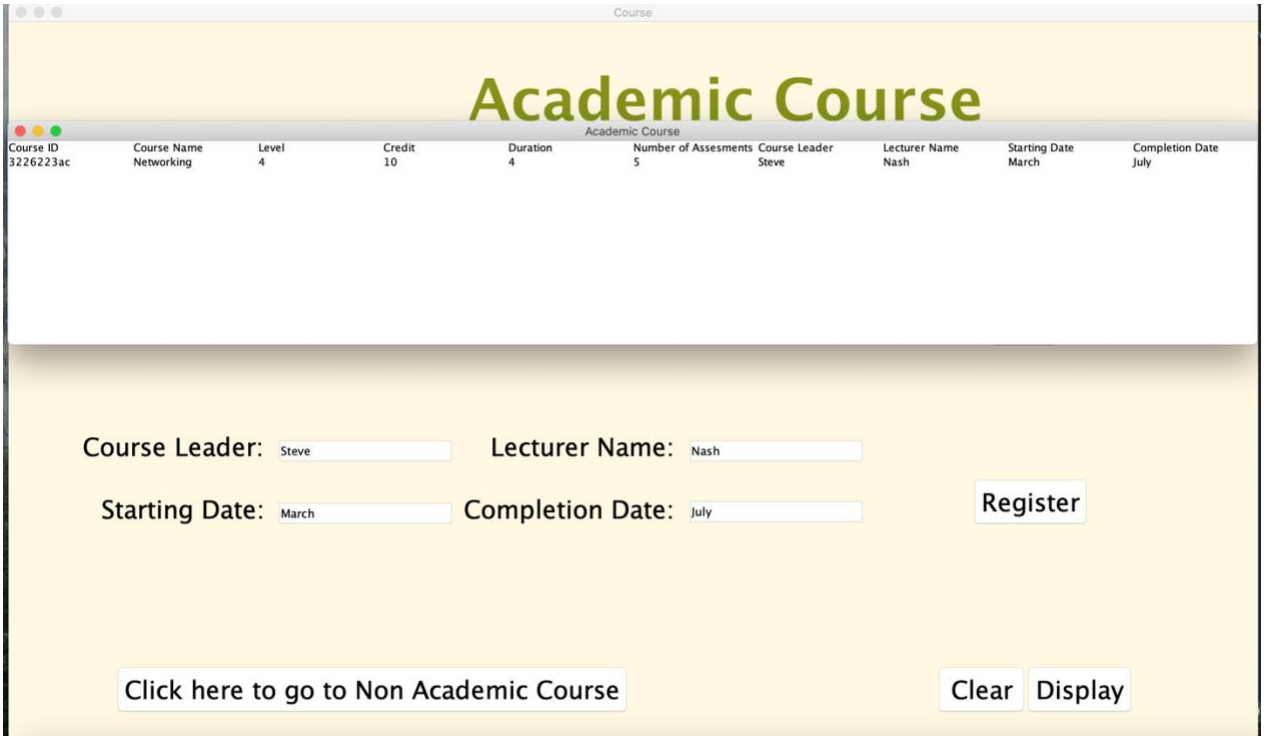

Figure 13: Screenshot of display frame which includes data entered in Academic course while clicking register button

5.2.d. Register non-academic course.

Table 5: To register Non-Academic Course

Test No:	2.d
Objective:	Register Non-Academic Course.
Action:	>> Assign values in Course Leader, Instructor Name, Starting Date, Completion Date, Exam Date. courseLeader = "Lewis" instructorName = "Roddy" startingDate = "April" completionDate = "August" examDate = "September" >>Click on Register button >>Click on Display button
Expected Result:	Should display "Non Academic Course is registered" dialog box and all records in a new frame.
Actual Result:	"Non Academic Course is registered" dialog box and display frame was displayed.
Conclusion	The test is successful.

Figure 14: Screenshot of entering values in text field of Non-Academic Course

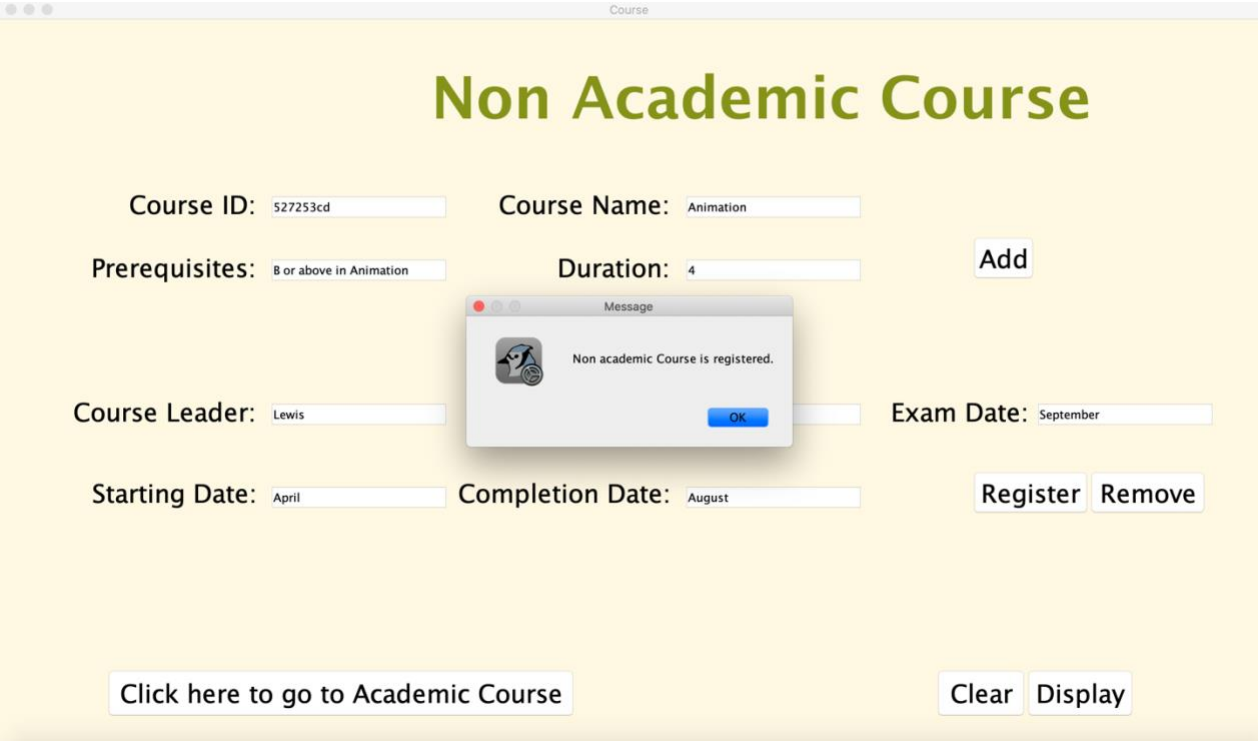

Figure 15: Screenshot of dialog box when clicking register button

Figure 16: Screenshot of display frame which includes data entered in Non-Academic course while clicking register button

5.2.e. Remove non-academic course.

Table 6: To remove Non-Academic Course

Test No:	2.e
Objective:	Remove Non-Academic Course.
Action:	>>Click on Remove button after registering >>Click on Display button
Expected Result:	Should display "Non Academic Course is removed" dialog box and all records in a new frame.
Actual Result:	"Non Academic Course is removed" dialog box and display frame with removed values of register was displayed.
Conclusion	The test is successful.

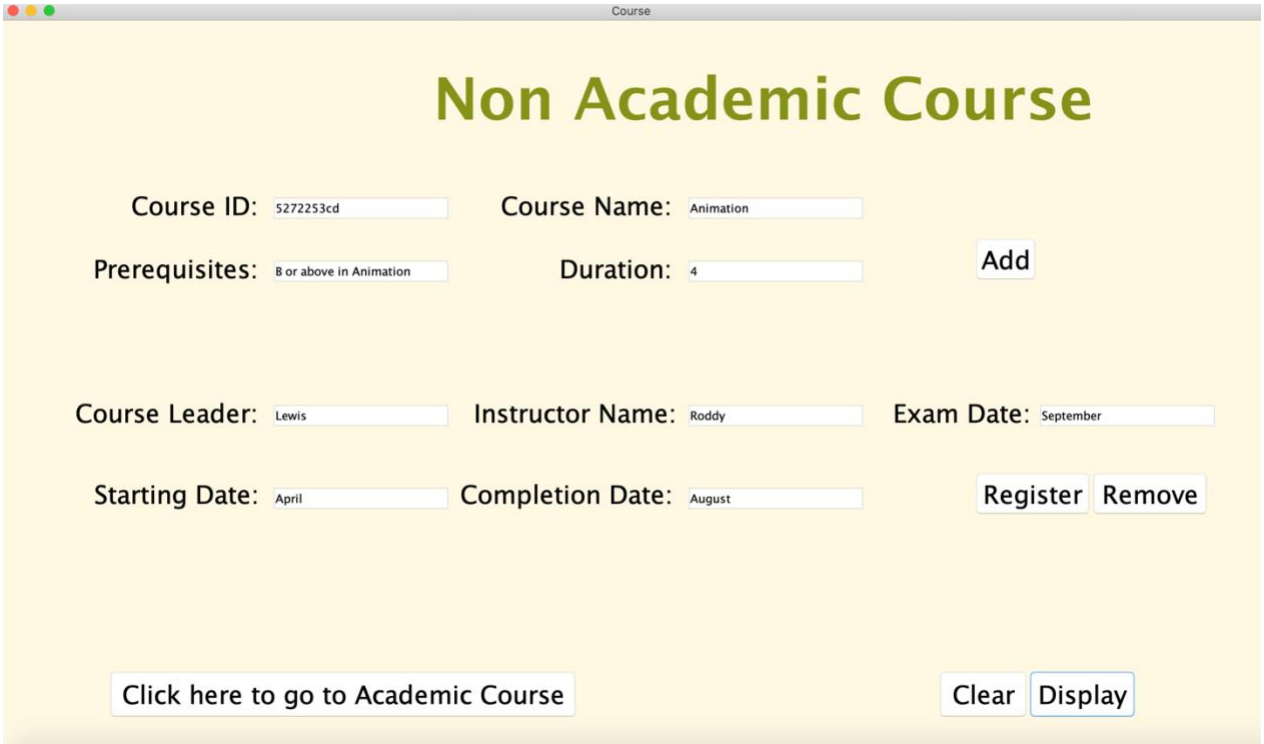

Figure 17: Screenshot of entering values in text field of Non-Academic Course

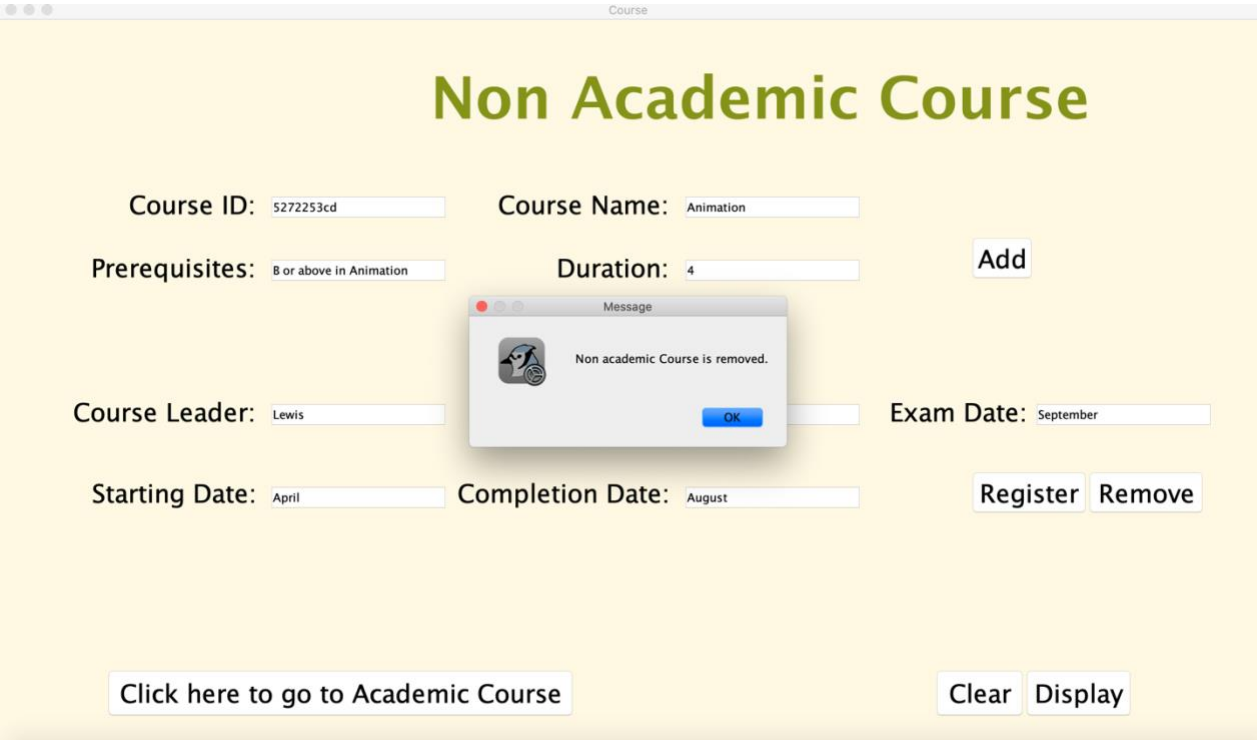

Figure 18: Screenshot of dialog box when clicking remove button

Figure 19: Screenshot of display frame which includes data entered in Non-Academic course after clicking remove button

5.3 Test 3:

- **Test that appropriate dialog boxes appear when:**

a. Trying to add duplicate courseID.

Table 7: To add duplicate courseID in Academic Course

Test No:	3.a
Objective:	Trying to add duplicate courseID in Academic Course.
Action:	>>In text fields, use the same value as in test 2.a. >>Click on add button
Expected Result:	Should display "The given courseID is already used. Please enter a different one" dialog box.
Actual Result:	"The given courseID is already used. Please enter a different one" dialog box was displayed.
Conclusion	The test is successful.

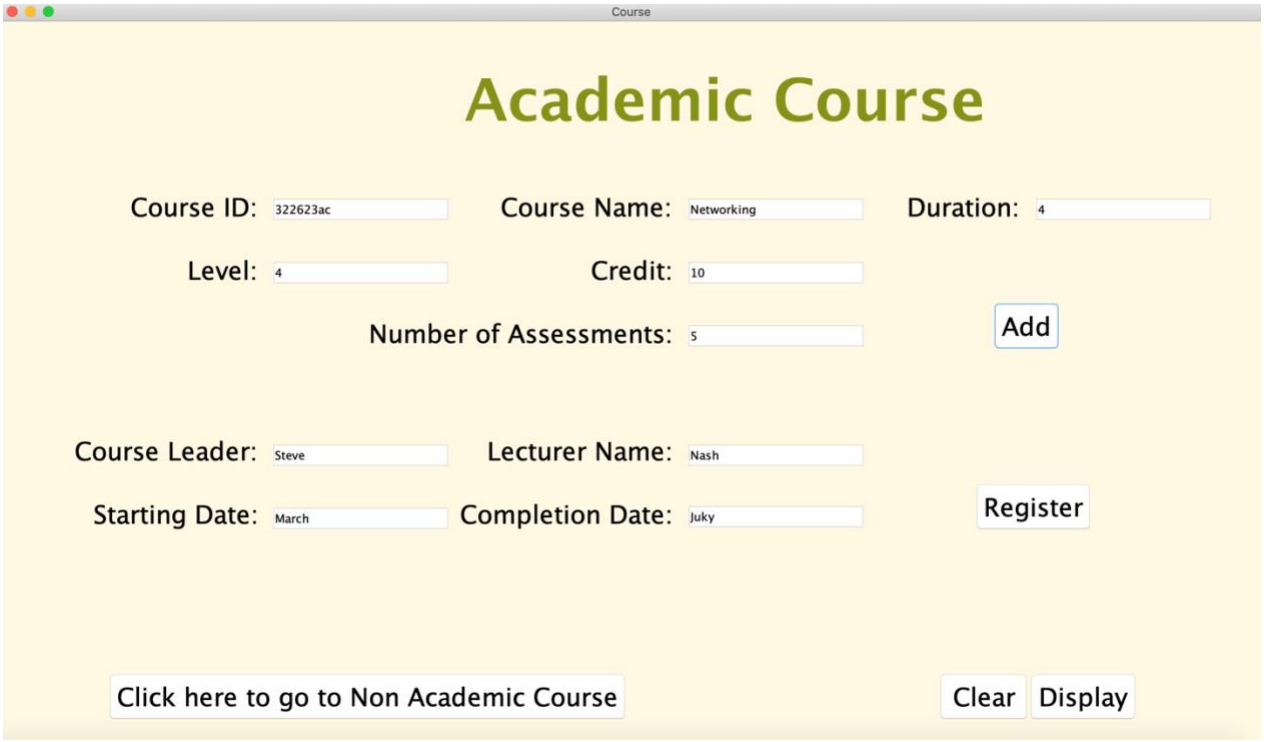

Figure 20: Screenshot of text fields in Academic Course where using same courseID which was registered before

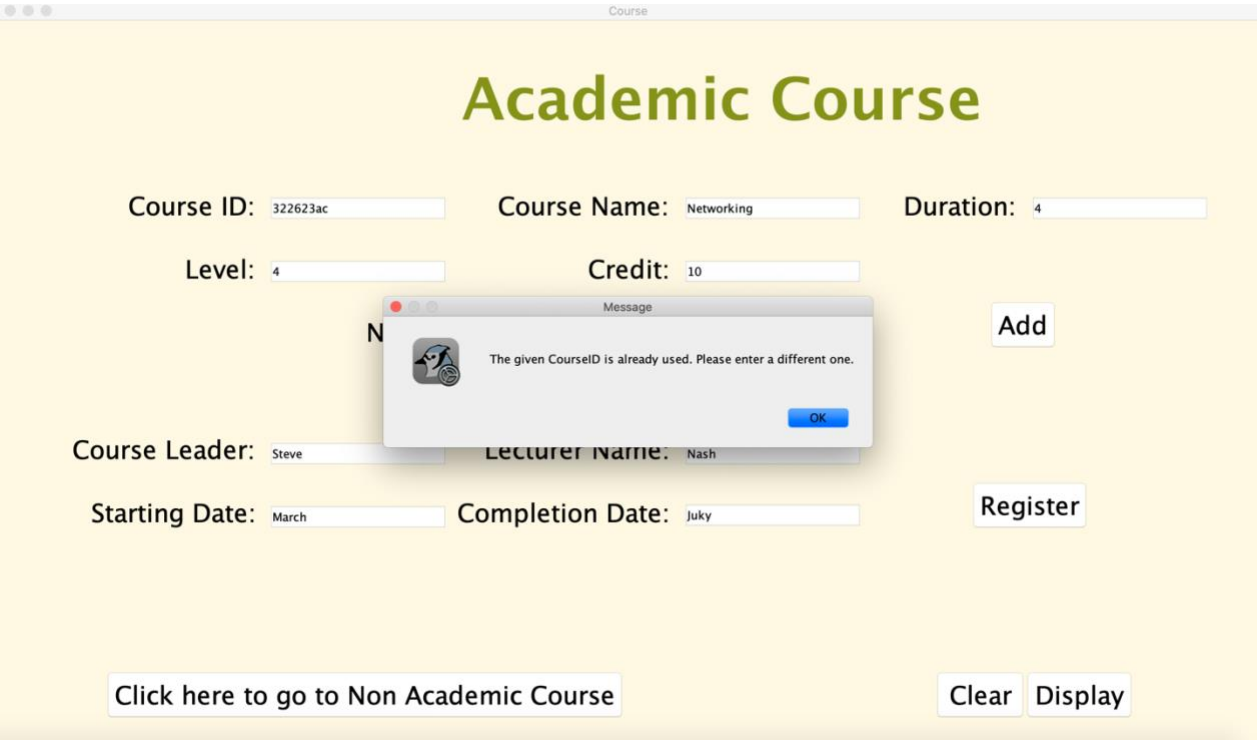

Figure 21: Screenshot of dialog box when trying to add duplicate courseID in Academic Course

Table 8: To add duplicate courseID in Non-Academic Course

Test No:	3.a
Objective:	Trying to add duplicate courseID in Non-Academic Course.
Action:	>>In text fields, use the same value as in test 2.b. >>Click on add button
Expected Result:	Should display "The given courseID is already used. Please enter a different one" dialog box.
Actual Result:	"The given courseID is already used. Please enter a different one" dialog box was displayed.
Conclusion	The test is successful.

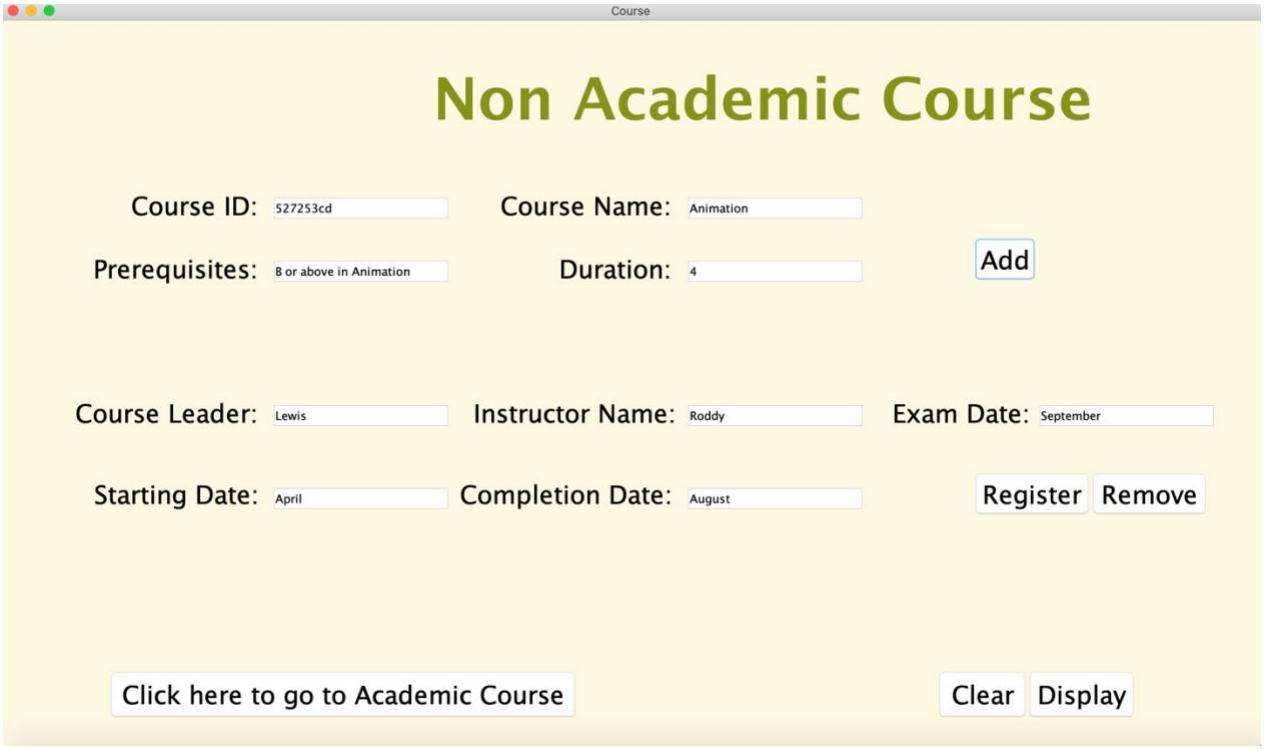

Figure 22: Screenshot of text fields in Non-Academic Course where using same courseID which was registered before

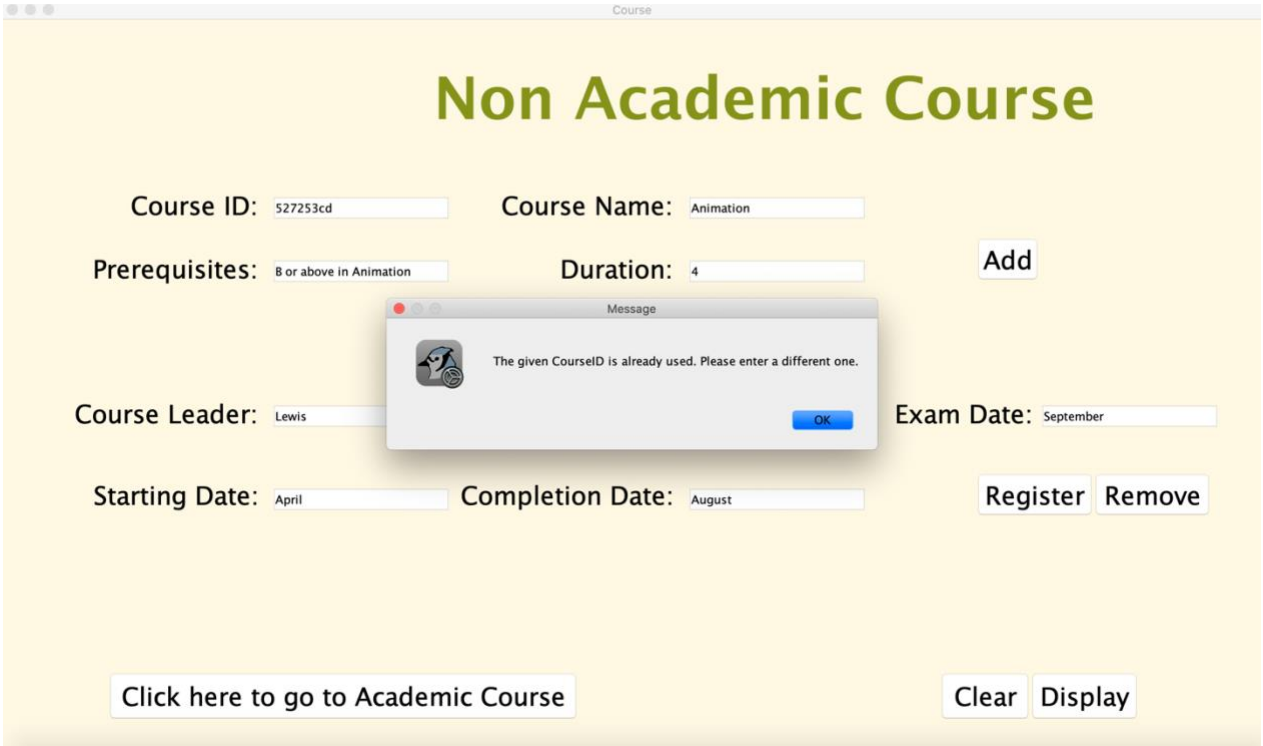

Figure 23: Screenshot of dialog box when trying to add duplicate courseID in Non-Academic Course

b. Trying to register already registered course.

Table 9: To register already registered course in Academic Course

Test No:	3.b
Objective:	Trying to register already registered course in Academic Course.
Action:	>>In text fields, use the same value as in test 2.c. >>Click on Register button
Expected Result:	Should display "Academic Course is Already registered" dialog box.
Actual Result:	"Academic Course is already registered" dialog box was displayed.
Conclusion	The test is successful.

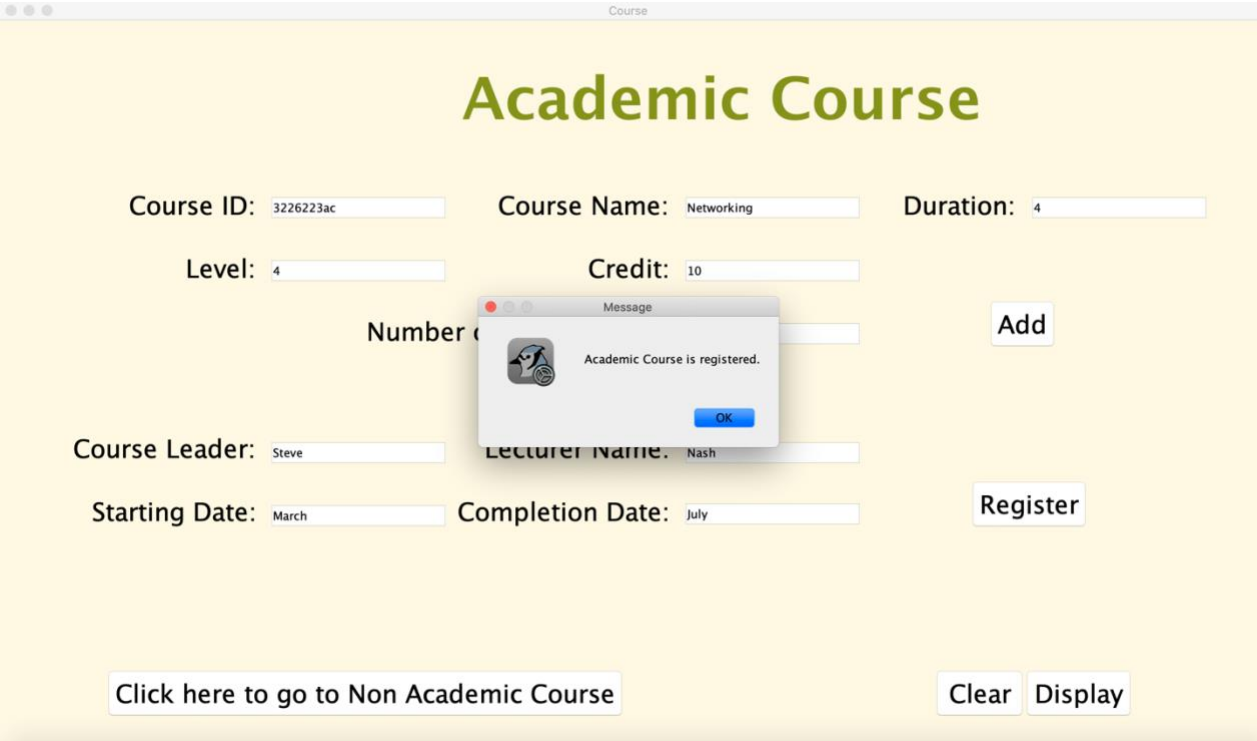

Figure 24: Screenshot of dialog box when clicking register button in Academic Course

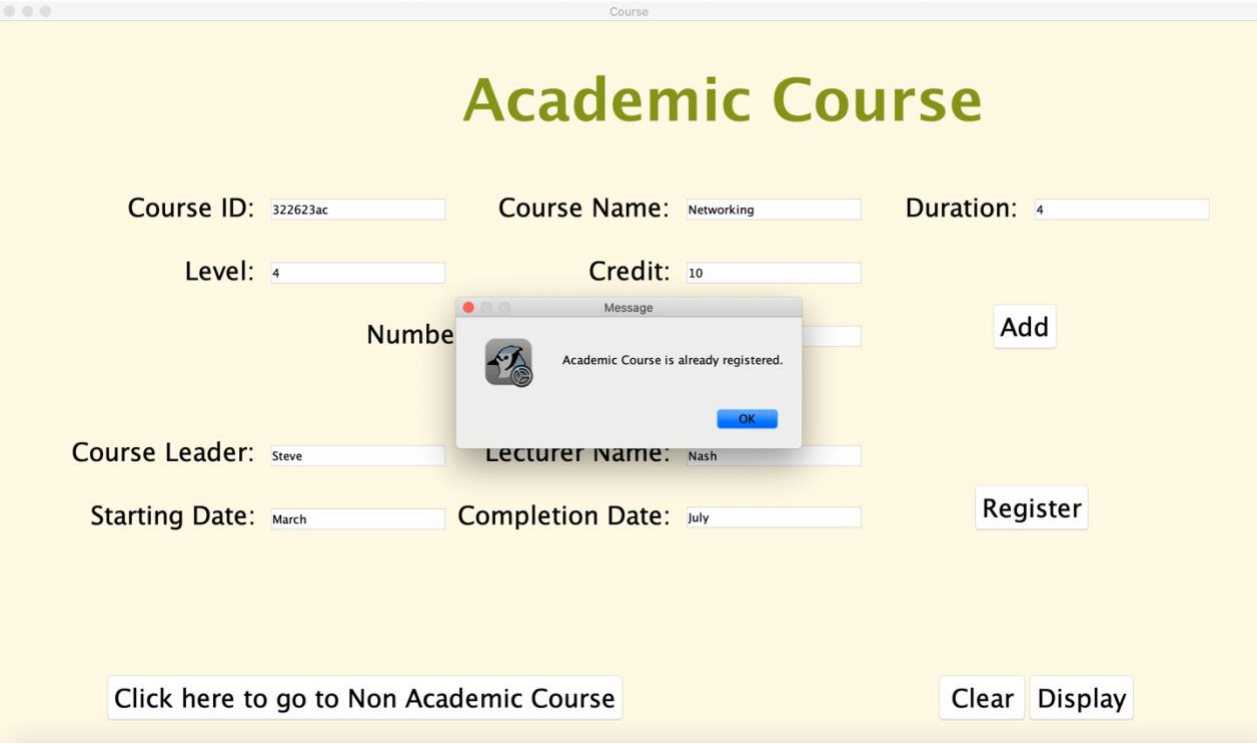

Figure 25: Screenshot of dialog box when trying to register already registered course in Academic Course

Table 10: To register already registered course in Non-Academic Course

Test No:	3.b
Objective:	Trying to register already registered course in Non Academic Course.
Action:	>>In text fields, use the same value as in test 2.d. >>Click on Register button
Expected Result:	Should display "Non academic Course is Already registered" dialog box.
Actual Result:	"Non academic Course is already registered" dialog box was displayed.
Conclusion	The test is successful.

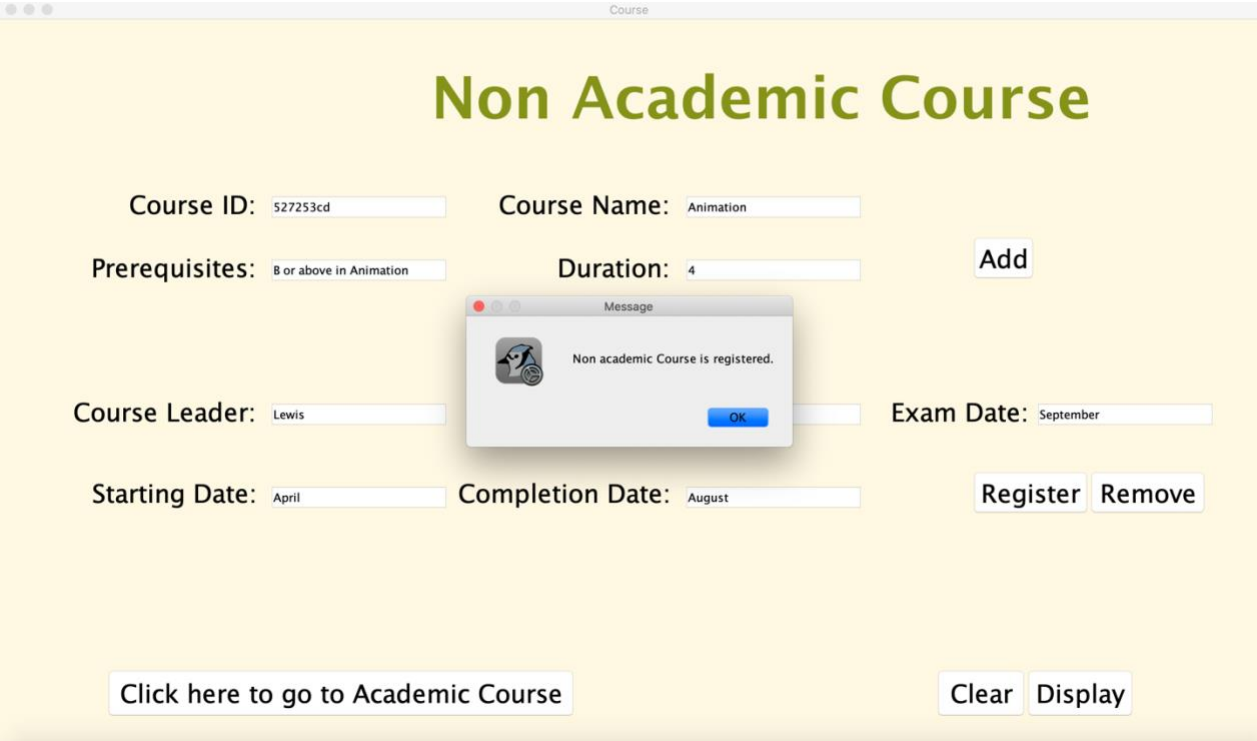

Figure 26: Screenshot of dialog box when clicking register button in Non-Academic Course

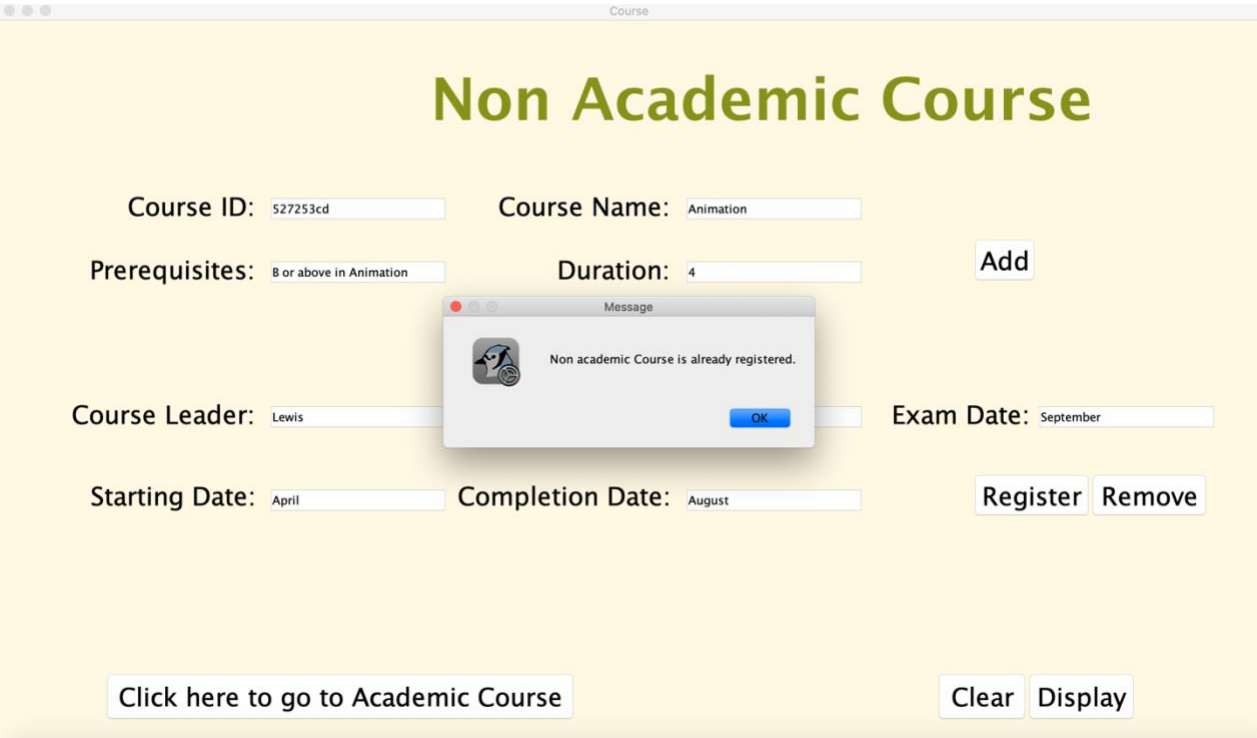

Figure 27: Screenshot of dialog box when trying to register already registered course in Non-Academic Course

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

Table 11: To remove the non-academic course which is already removed

Test No:	3.c
Objective:	Trying to remove the non-academic course which is already removed.
Action:	>>In text fields, use the same value as in test 2.e. >>Click on Remove button
Expected Result:	Should display "Non academic Course is Already removed" dialog box.
Actual Result:	"Non academic Course is already removed" dialog box was displayed.
Conclusion	The test is successful.

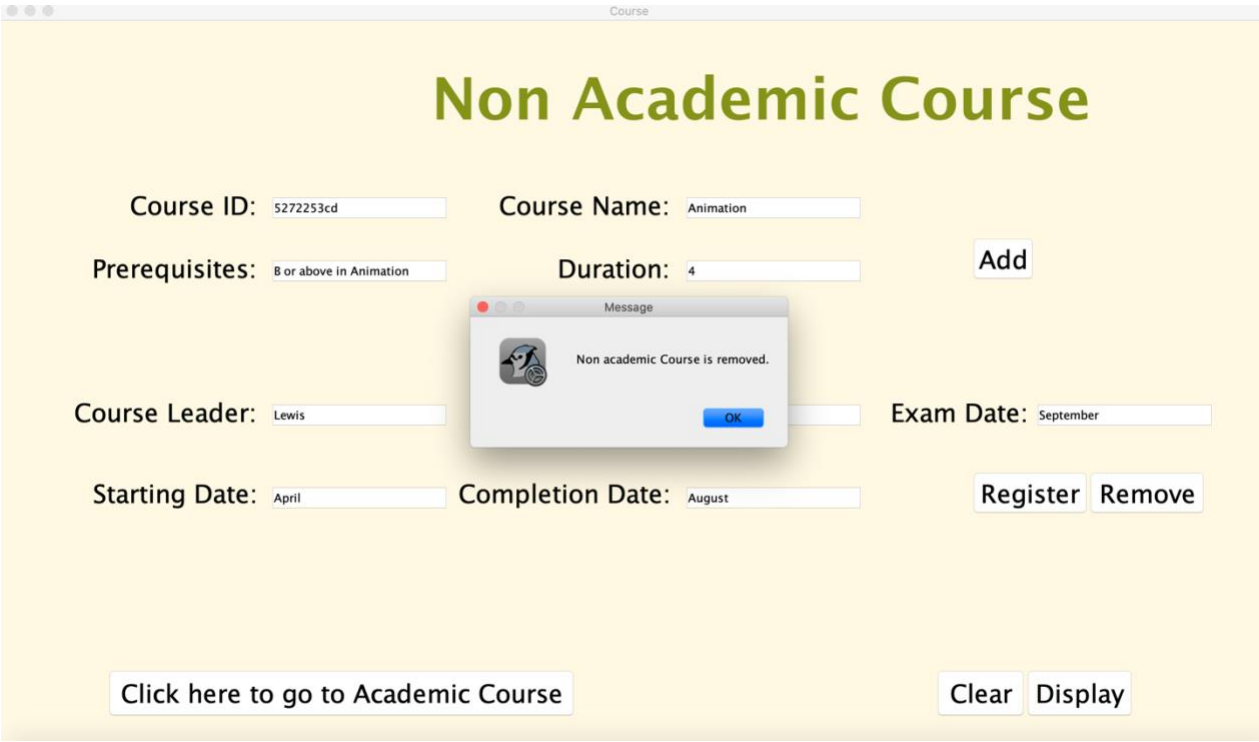

Figure 28: Screenshot of dialog box when clicking remove button in Non-Academic Course

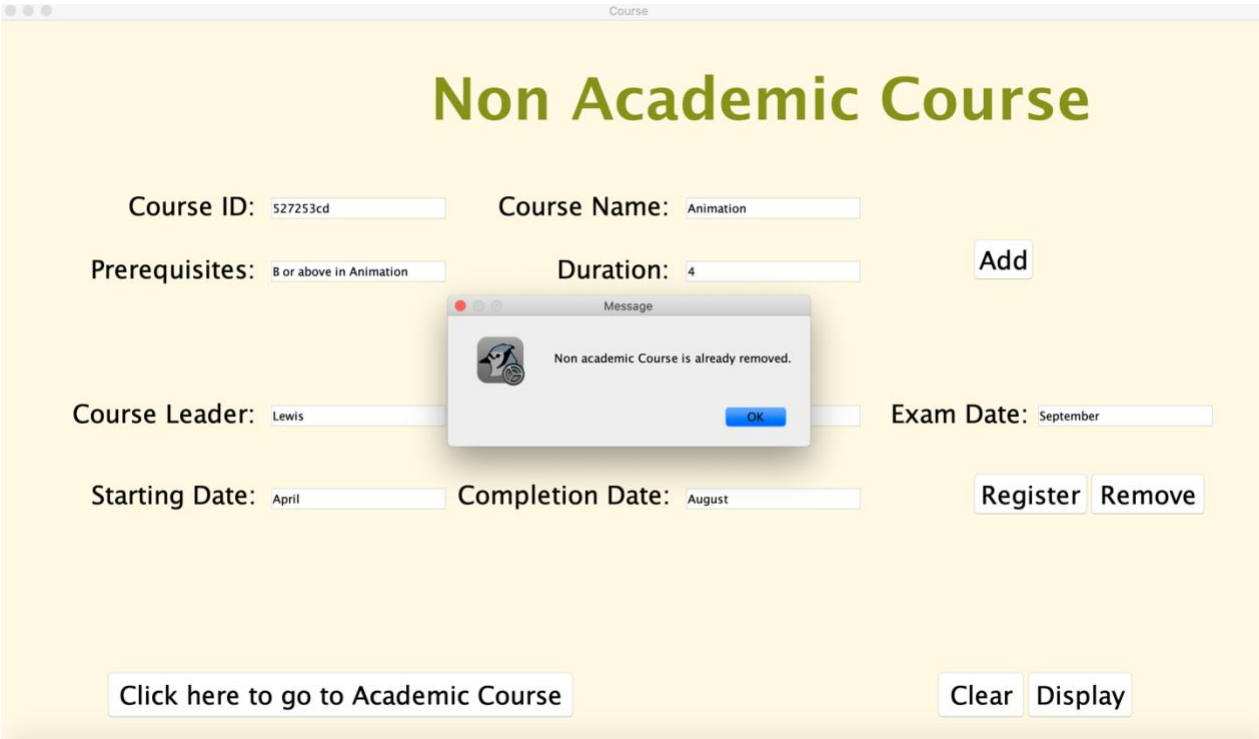

Figure 29: Screenshot of dialog box when trying to register already removed course in Non-Academic Course

6. Error

6.1 Syntax error

A syntax error is a mistake in a program's source code. Since computer programs must adhere to strict syntax in order to compile correctly, any parts of the code that do not follow the programming language's syntax will result in a syntax error. (TechTerms, 2012)

Here a small error was made while assigning parentheses.

```
ction listener for Add button of Academic Course
_Academic_add.addActionListener(new ActionListener({
     public void actionPerformed(ActionEvent e)
         {
             //try catch for Integer datatype
```

Figure 30: Screenshot of syntax error

To solve the error, parentheses were properly assigned.

```
//Action listener for Add button of Academic Course
btn_Academic_add.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
            {
```

Figure 31: Screenshot of solved syntax error

6.2 Logical error

A logical error is a mistake in the source code of a program that causes it to behave incorrectly or unexpectedly. It's a form of runtime error that can cause a program to crash or simply produce the incorrect performance. (TechTerms, 2012)

Here when the text field was left empty and using add button it was showing invalid data type. Whereas, while passing a string in text field of duration it was showing the text field is empty. Therefore all the output that if was meant to give is in Else and vice versa.

```
String duration_NAcademic_temp = txt_NAcademic_Duration.getText();
if (duration_NAcademic_temp.isEmpty())
{
    JOptionPane.showMessageDialog(jf,"You have entered invalid data type.");
}
else
{
    JOptionPane.showMessageDialog(jf,"The text field is empty, please fill it up.");
}
```

Figure 32: Screenshot of logical error

To solve the error, all the output of if and else is put in their respective place.

```
String duration_NAcademic_temp = txt_NAcademic_Duration.getText();
if (duration_NAcademic_temp.isEmpty())
{
    JOptionPane.showMessageDialog(jf,"The text field is empty, please fill it up.");
}
else
{
    JOptionPane.showMessageDialog(jf,"You have entered invalid data type.");
}
```

Figure 33: Screenshot of solved logical error

6.3 Semantic error

Semantic errors are issues with a program that runs without error messages but doesn't do what it's supposed to do. For example, an expression cannot be evaluated in the expected order, resulting in an incorrect result. (thinkpython21, 2020)

Here while performing display method of Academic Course the name of the table was misspelled so a semantic error occurred while using display buuton.

```
//Action listener for Display button of Academic Course
btn_Academic_Display.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
            {
                Academic_display_jf= new JFrame("Academic Course");
                Academic_display_jf.setBounds(10,10,1400,250);

                DefaultTableModel Academic_table_model= new DefaultTableModel();
                //Creating table
                NAcademic_table = new JTable(Academic_table_model);
                //Columns in table
                Academic_table_model.addColumn("Course ID");
                Academic_table_model.addColumn("Course Name");
```

Figure 34: Screenshot of semantic error

```
Can only enter input while your programming is running

Exception in thread "AWT-EventQueue-0" java.lang.NullPointerException
        at java.desktop/java.awt.Container.addImpl(Container.java:1117)
        at java.desktop/java.awt.Container.add(Container.java:1029)
        at java.desktop/javax.swing.JFrame.addImpl(JFrame.java:553)
        at java.desktop/java.awt.Container.add(Container.java:436)
        at INGCollege$12.actionPerformed(INGCollege.java:801)
        at java.desktop/javax.swing.AbstractButton.fireActionPerformed(Abst
        at java.desktop/javax.swing.AbstractButton$Handler.actionPerformed(
```

Figure 35: Screenshot of program crash in java due to semantic error

To solve the error, the spelling was correction was made.

```
//Action listener for Display button of Academic Course
btn_Academic_Display.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
            {
                Academic_display_jf= new JFrame("Academic Course");
                Academic_display_jf.setBounds(10,10,1400,250);

                DefaultTableModel Academic_table_model= new DefaultTableModel();
                //Creating table
                Academic_table = new JTable(Academic_table_model);
                //Columns in table
                Academic_table_model.addColumn("Course_ID");
```

Figure 36: Screenshot of solved semantic error

7. Conclusion

Finally, this was the programming module's second coursework. This coursework was entirely focused on developing a user interface for course registration. This lesson taught us to new Java components. The coursework' framework was built using constructor methods, which included a single java frame, multiple panels, jlabel, text fields, buttons, font, color, and many other features.

We were given the challenge of designing an INGCollege class for this course, as well as dealing with new programming approaches and concepts throughout the course. New methods like actionPerformed(ActionEvent e), register, delete, and main string provided me a new perspective on the java programming language. Each method's functioning mechanism ensured that the GUI ran smoothly and without errors. This was my first time simultaneously studying and building a GUI.

The syntax of the Java programming language utilized in this training was utterly unfamiliar to me. The concept of approaches was extremely difficult to grasp and put into practice. Due to a mismatch of constructor names from prior coursework courses, working on the methods of different buttons such as add, register, remove, and display took a long time. The duration and number of assessments were not accepted when working on the add button at first. It was fixed after using the try catch in the add button. Because I didn't specify the panel's visibility at the end of the program, one of the biggest issues I experienced during this coursework was that it wouldn't display the panel's components. I had to minimize and reopen the java frame every time. These were the few challenges I encountered in this course. To overcome these difficulties, I thoroughly watched our workshop class videos and learned the java syntax.

This report, in particular, includes a class diagram of the INGCollege class, pseudo codes, method descriptions, all tests performed, and errors encountered while carrying out the coursework.

8. Appendix

8.1 List of the code:

- **INGCollege Class:**

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.table.DefaultTableModel;
public class INGCollege
{
   private JFrame jf, Academic_display_jf, NAcademic_display_jf;
   private JPanel WPjpane, ACjpane, NCjpane;
   private JLabel lbl_WP, lbl_WP_to, lbl_WP_m, lbl_WP_f, lbl_WP_CT, lbl_pb,
lbl_Academic, lbl_Academic_ID, lbl_Academic_Name,
                lbl_Academic_Duration, lbl_Academic_Level, lbl_Academic_Credit,
lbl_Academic_NOA, lbl_Academic_Leader, lbl_Academic_Lecturer,
                lbl_Academic_SDate, lbl_Academic_CDate, lbl_NAcademic,
lbl_NAcademic_ID, lbl_NAcademic_Name, lbl_NAcademic_Prerequiste,
                lbl_NAcademic_Duration, lbl_NAcademic_Leader,
lbl_NAcademic_Instructor, lbl_NAcademic_EDate, lbl_NAcademic_SDate,
                lbl_NAcademic_CDate;
   private JTextField txt_Academic_ID, txt_Academic_Name, txt_Academic_Duration,
txt_Academic_Level, txt_Academic_Credit, txt_Academic_NOA,
                   txt_Academic_Leader, txt_Academic_Lecturer, txt_Academic_SDate,
txt_Academic_CDate, txt_NAcademic_ID, txt_NAcademic_Name,
                   txt_NAcademic_Prerequiste, txt_NAcademic_Duration,
txt_NAcademic_Leader, txt_NAcademic_Instructor, txt_NAcademic_EDate,
                   txt_NAcademic_SDate, txt_NAcademic_CDate;
   private JButton btn_WP_AC, btn_WP_NC, btn_Academic_add,
btn_Academic_Register, btn_NAcademic, btn_Academic_Clear, btn_Academic_Display,
                  btn_NAcademic_add, btn_NAcademic_Register, btn_NAcademic_Remove,
btn_Academic, btn_NAcademic_Clear, btn_NAcademic_Display;
   private Font fnt1, fnt2, fnt3, fnt4;
   private Icon ing, islington;
   private JTable Academic_table, NAcademic_table;
   private DefaultTableModel Academic_table_model, NAcademic_table_model;
   private ArrayList<Course> academicCourseList, nonAcademicCourseList;
   public INGCollege()
   {
      ///Creating frame for Course
```

```
jf = new JFrame("Course");
jf.setBounds(10,10,1400,900);
jf.setLayout(null);
jf.setDefaultCloseOperation(jf.EXIT_ON_CLOSE);

///Font for title
fnt1 = new Font("Areal",Font.BOLD,65);

///Font for text fields, labels and buttons
fnt2 = new Font("Areal",Font.PLAIN,28);

///Font for Welcome
fnt3 = new Font("Areal",Font.BOLD,80);

///Font for text of welcome page
fnt4 = new Font("Areal",Font.BOLD,28);

///Creating pannel for welcome page
WPjpane = new JPanel();
WPjpane.setBounds(0,0,1500,1000);
WPjpane.setLayout(null);
WPjpane.setBackground(Color.BLACK);

///Add logo in welcome page
ing = new ImageIcon(getClass().getResource("ING-Group.png"));
JLabel img1 = new JLabel(ing);
img1.setBounds(40,185,250,250);
WPjpane.add(img1);

islington = new ImageIcon(getClass().getResource("islington-logo.png"));
JLabel img2 = new JLabel(islington);
img2.setBounds(40,480,250,250);
WPjpane.add(img2);

///For Welcome
lbl_WP = new JLabel("Welcome");
lbl_WP.setBounds(400,170,500,70); //220
lbl_WP.setFont(fnt1);
lbl_WP.setForeground(Color.WHITE);
WPjpane.add(lbl_WP);

///for To
```

```
lbl_WP_to = new JLabel("To");
lbl_WP_to.setBounds(400,255,500,90); //290
lbl_WP_to.setFont(fnt3);
lbl_WP_to.setForeground(Color.WHITE);
WPjpane.add(lbl_WP_to);

///for message
lbl_WP_m = new JLabel("Course Registration");
lbl_WP_m.setBounds(530,255,850,90); //290
lbl_WP_m.setFont(fnt3);
lbl_WP_m.setForeground(new Color(136,145,51));
WPjpane.add(lbl_WP_m);

///for form
lbl_WP_f = new JLabel("Form");
lbl_WP_f.setBounds(400,340,500,100); //370
lbl_WP_f.setFont(fnt3);
lbl_WP_f.setForeground(new Color(136,145,51));
WPjpane.add(lbl_WP_f);

///for Course Type
lbl_WP_CT = new JLabel("Select your course type");
lbl_WP_CT.setBounds(400,470,500,100);
lbl_WP_CT.setFont(fnt4);
lbl_WP_CT.setForeground(Color.WHITE);
WPjpane.add(lbl_WP_CT);

///AC Button
btn_WP_AC = new JButton("Academic Course");
btn_WP_AC.setBounds(400,570,260,60);
btn_WP_AC.setFont(fnt2);
WPjpane.add(btn_WP_AC);

///NC Button
btn_WP_NC = new JButton("Non Academic Course");
btn_WP_NC.setBounds(670,570,320,60);
btn_WP_NC.setFont(fnt2);
WPjpane.add(btn_WP_NC);

///Powered By
lbl_pb = new JLabel("Powered By : Sarthak Bikram Rana");
lbl_pb.setBounds(400,730,550,50);
lbl_pb.setFont(fnt4);
lbl_pb.setForeground(Color.WHITE);
```

```
WPjpane.add(lbl_pb);

///Creating Pannel for AC
ACjpane = new JPanel();
ACjpane.setBounds(0,0,1500,1000);
ACjpane.setLayout(null);
ACjpane.setBackground(new Color(255,249,230)); //(221,190,169)

///Creating title Academic Courses
lbl_Academic = new JLabel("Academic Course");
lbl_Academic.setBounds(515,55,600,60);
lbl_Academic.setFont(fnt1);
lbl_Academic.setForeground(new Color(136,145,51)); //136,145,51
ACjpane.add(lbl_Academic);

///Creating CourseID
lbl_Academic_ID = new JLabel("Course ID:");
lbl_Academic_ID.setBounds(145,180,170,50);
lbl_Academic_ID.setForeground(Color.BLACK);
lbl_Academic_ID.setFont(fnt2);
ACjpane.add(lbl_Academic_ID);
///Creating CourseID text field
txt_Academic_ID = new JTextField();
txt_Academic_ID.setBounds(300,193,200,30);
txt_Academic_ID.setBackground(Color.WHITE);
ACjpane.add(txt_Academic_ID);

///Creating Course Name
lbl_Academic_Name = new JLabel("Course Name:");
lbl_Academic_Name.setBounds(555,180,190,50);
lbl_Academic_Name.setForeground(Color.BLACK);
lbl_Academic_Name.setFont(fnt2);
ACjpane.add(lbl_Academic_Name);
///Creating Course Name text field
txt_Academic_Name = new JTextField();
txt_Academic_Name.setBounds(760,193,200,30);
txt_Academic_Name.setBackground(Color.WHITE);
ACjpane.add(txt_Academic_Name);

///Creating Duration
lbl_Academic_Duration = new JLabel("Duration:");
lbl_Academic_Duration.setBounds(1005,180,130,50);
```

```
lbl_Academic_Duration.setForeground(Color.BLACK);
lbl_Academic_Duration.setFont(fnt2);
ACjpane.add(lbl_Academic_Duration);
///Creating Duration text field
txt_Academic_Duration = new JTextField();
txt_Academic_Duration.setBounds(1145,193,200,30);
txt_Academic_Duration.setBackground(Color.WHITE);
ACjpane.add(txt_Academic_Duration);

///Creating Level
lbl_Academic_Level = new JLabel("Level:");
lbl_Academic_Level.setBounds(208,250,100,50);
lbl_Academic_Level.setForeground(Color.BLACK);
lbl_Academic_Level.setFont(fnt2);
ACjpane.add(lbl_Academic_Level);
///Creating Level text field
txt_Academic_Level = new JTextField();
txt_Academic_Level.setBounds(300,263,200,30);
txt_Academic_Level.setBackground(Color.WHITE);
ACjpane.add(txt_Academic_Level);

///Creating Credit
lbl_Academic_Credit = new JLabel("Credit:");
lbl_Academic_Credit.setBounds(655,250,100,50);
lbl_Academic_Credit.setForeground(Color.BLACK);
lbl_Academic_Credit.setFont(fnt2);
ACjpane.add(lbl_Academic_Credit);
/// Creating Credit Text field
txt_Academic_Credit = new JTextField();
txt_Academic_Credit.setBounds(760,263,200,30);
txt_Academic_Credit.setBackground(Color.WHITE);
ACjpane.add(txt_Academic_Credit);

///Creating Number of Assessments
lbl_Academic_NOA = new JLabel("Number of Assessments:");
lbl_Academic_NOA.setBounds(409,320,350,50);
lbl_Academic_NOA.setForeground(Color.BLACK);
lbl_Academic_NOA.setFont(fnt2);
ACjpane.add(lbl_Academic_NOA);
///Creating Number of Assessments text field
txt_Academic_NOA = new JTextField();
txt_Academic_NOA.setBounds(760,333,200,30);
txt_Academic_NOA.setBackground(Color.WHITE);
ACjpane.add(txt_Academic_NOA);
```

```
///Add Button
btn_Academic_add = new JButton("Add");
btn_Academic_add.setBounds(1100,310,75,55);
btn_Academic_add.setFont(fnt2);
ACjpane.add(btn_Academic_add);

//Creating Course Leader
lbl_Academic_Leader = new JLabel("Course Leader:");
lbl_Academic_Leader.setBounds(83,450,220,50);
lbl_Academic_Leader.setForeground(Color.BLACK);
lbl_Academic_Leader.setFont(fnt2);
ACjpane.add(lbl_Academic_Leader);
///Creating Course Leader text field
txt_Academic_Leader = new JTextField();
txt_Academic_Leader.setBounds(300,465,200,30);
txt_Academic_Leader.setBackground(Color.WHITE);
ACjpane.add(txt_Academic_Leader);

//Creating Lecturer Name
lbl_Academic_Lecturer = new JLabel("Lecturer Name:");
lbl_Academic_Lecturer.setBounds(540,450,250,50);
lbl_Academic_Lecturer.setForeground(Color.BLACK);
lbl_Academic_Lecturer.setFont(fnt2);
ACjpane.add(lbl_Academic_Lecturer);
/// Creating Lecturer Name text field
txt_Academic_Lecturer = new JTextField();
txt_Academic_Lecturer.setBounds(760,465,200,30);
txt_Academic_Lecturer.setBackground(Color.WHITE);
ACjpane.add(txt_Academic_Lecturer);

///Creating Starting Date
lbl_Academic_SDate = new JLabel("Starting Date:");
lbl_Academic_SDate.setBounds(104,520,200,50);
lbl_Academic_SDate.setForeground(Color.BLACK);
lbl_Academic_SDate.setFont(fnt2);
ACjpane.add(lbl_Academic_SDate);
///Creating Starting Date text field
txt_Academic_SDate = new JTextField();
txt_Academic_SDate.setBounds(300,535,200,30);
txt_Academic_SDate.setBackground(Color.WHITE);
ACjpane.add(txt_Academic_SDate);
```

```
///Creating Completion Date
lbl_Academic_CDate = new JLabel("Completion Date:");
lbl_Academic_CDate.setBounds(510,520,240,50);
lbl_Academic_CDate.setForeground(Color.BLACK);
lbl_Academic_CDate.setFont(fnt2);
ACjpane.add(lbl_Academic_CDate);
///Creating Course Name text field
txt_Academic_CDate = new JTextField();
txt_Academic_CDate.setBounds(760,533,200,30);
txt_Academic_CDate.setBackground(Color.WHITE);
ACjpane.add(txt_Academic_CDate);

///Register Button
btn_Academic_Register = new JButton("Register");
btn_Academic_Register.setBounds(1080,510,130,55);
btn_Academic_Register.setFont(fnt2);
ACjpane.add(btn_Academic_Register);

//Changing Non AcademicCourse Button
btn_NAcademic = new JButton("Click here to go to Non Academic Course");
btn_NAcademic.setBounds(120,720,575,55);
btn_NAcademic.setFont(fnt2);
ACjpane.add(btn_NAcademic);

//Clear Button
btn_Academic_Clear = new JButton("Clear");
btn_Academic_Clear.setBounds(1040,720,100,55);
btn_Academic_Clear.setFont(fnt2);
ACjpane.add(btn_Academic_Clear);

//Display Button
btn_Academic_Display = new JButton("Display");
btn_Academic_Display.setBounds(1140,720,120,55);
btn_Academic_Display.setFont(fnt2);
ACjpane.add(btn_Academic_Display);
///Academic Course pannel ends here

///for Non Academic Course
///Creating panel for Non Academic Course
NCjpane = new JPanel();
NCjpane.setBounds(0,0,1500,1000);
NCjpane.setLayout(null);
NCjpane.setBackground(new Color(255,249,230));
```

```
//Creating Title Non Academic Coures
lbl_NAcademic = new JLabel("Non Academic Course");
lbl_NAcademic.setBounds(480,55,750,60);
lbl_NAcademic.setFont(fnt1);
lbl_NAcademic.setForeground(new Color(136,145,51));
NCjpane.add(lbl_NAcademic);

///Creating CourseID
lbl_NAcademic_ID = new JLabel("Course ID:");
lbl_NAcademic_ID.setBounds(145,180,170,50);
lbl_NAcademic_ID.setForeground(Color.BLACK);
lbl_NAcademic_ID.setFont(fnt2);
NCjpane.add(lbl_NAcademic_ID);
///Creating CourseID text field
txt_NAcademic_ID = new JTextField();
txt_NAcademic_ID.setBounds(300,193,200,30);
NCjpane.add(txt_NAcademic_ID);

///Creating Course Name
lbl_NAcademic_Name = new JLabel("Course Name:");
lbl_NAcademic_Name.setBounds(555,180,190,50);
lbl_NAcademic_Name.setFont(fnt2);
NCjpane.add(lbl_NAcademic_Name);
///Creating Course Name text field
txt_NAcademic_Name = new JTextField();
txt_NAcademic_Name.setBounds(760,193,200,30);
NCjpane.add(txt_NAcademic_Name);

///Creating Prerequisites
lbl_NAcademic_Prerequiste = new JLabel("Prerequisites:");
lbl_NAcademic_Prerequiste.setBounds(103,250,200,50);
lbl_NAcademic_Prerequiste.setFont(fnt2);
NCjpane.add(lbl_NAcademic_Prerequiste);
///Creating Prerequisites text field
txt_NAcademic_Prerequiste = new JTextField();
txt_NAcademic_Prerequiste.setBounds(300,263,200,30);
NCjpane.add(txt_NAcademic_Prerequiste);

///Creating Duration
lbl_NAcademic_Duration = new JLabel("Duration:");
lbl_NAcademic_Duration.setBounds(620,250,150,50);
```

```
lbl_NAcademic_Duration.setFont(fnt2);
NCjpane.add(lbl_NAcademic_Duration);
///Creating Duration text field
txt_NAcademic_Duration = new JTextField();
txt_NAcademic_Duration.setBounds(760,263,200,30);
NCjpane.add(txt_NAcademic_Duration);

///Add Button
btn_NAcademic_add = new JButton("Add");
btn_NAcademic_add.setBounds(1080,240,70,50);
btn_NAcademic_add.setFont(fnt2);
NCjpane.add(btn_NAcademic_add);

//Creating Course Leader
lbl_NAcademic_Leader = new JLabel("Course Leader:");
lbl_NAcademic_Leader.setBounds(83,410,220,50);
lbl_NAcademic_Leader.setFont(fnt2);
NCjpane.add(lbl_NAcademic_Leader);
///Creating Course Leader text field
txt_NAcademic_Leader = new JTextField();
txt_NAcademic_Leader.setBounds(300,423,200,30);
NCjpane.add(txt_NAcademic_Leader);

//Creating Instructor Name
lbl_NAcademic_Instructor = new JLabel("Instructor Name:");
lbl_NAcademic_Instructor.setBounds(525,410,250,50);
lbl_NAcademic_Instructor.setFont(fnt2);
NCjpane.add(lbl_NAcademic_Instructor);
/// Creating Instructor Name text field
txt_NAcademic_Instructor = new JTextField();
txt_NAcademic_Instructor.setBounds(760,423,200,30);
NCjpane.add(txt_NAcademic_Instructor);

///Creating Exam Date
lbl_NAcademic_EDate = new JLabel("Exam Date:");
lbl_NAcademic_EDate.setBounds(990,410,200,50);
lbl_NAcademic_EDate.setFont(fnt2);
NCjpane.add(lbl_NAcademic_EDate);
/// Credit Text field
txt_NAcademic_EDate = new JTextField();
txt_NAcademic_EDate.setBounds(1150,423,200,30);
NCjpane.add(txt_NAcademic_EDate);

///Creating Starting Date
```

```
lbl_NAcademic_SDate = new JLabel("Starting Date:");
lbl_NAcademic_SDate.setBounds(104,500,200,50);
lbl_NAcademic_SDate.setFont(fnt2);
NCjpane.add(lbl_NAcademic_SDate);
///Creating Starting Date text field
txt_NAcademic_SDate = new JTextField();
txt_NAcademic_SDate.setBounds(300,515,200,30);
NCjpane.add(txt_NAcademic_SDate);

/////Creating Completion Date
lbl_NAcademic_CDate = new JLabel("Completion Date:");
lbl_NAcademic_CDate.setBounds(510,500,240,50);
lbl_NAcademic_CDate.setFont(fnt2);
NCjpane.add(lbl_NAcademic_CDate);
///Creating Course Name text field
txt_NAcademic_CDate = new JTextField();
txt_NAcademic_CDate.setBounds(760,515,200,30);
NCjpane.add(txt_NAcademic_CDate);

///Register Button
btn_NAcademic_Register = new JButton("Register");
btn_NAcademic_Register.setBounds(1080,500,130,50);
btn_NAcademic_Register.setFont(fnt2);
NCjpane.add(btn_NAcademic_Register);

///Remove Button
btn_NAcademic_Remove = new JButton("Remove");
btn_NAcademic_Remove.setBounds(1210,500,130,50);
btn_NAcademic_Remove.setFont(fnt2);
NCjpane.add(btn_NAcademic_Remove);

//Changing Academic Course Button
btn_Academic = new JButton("Click here to go to Academic Course");
btn_Academic.setBounds(120,720,520,55);
btn_Academic.setFont(fnt2);
NCjpane.add(btn_Academic);

//Clear Button
btn_NAcademic_Clear = new JButton("Clear");
btn_NAcademic_Clear.setBounds(1040,720,100,55);
btn_NAcademic_Clear.setFont(fnt2);
NCjpane.add(btn_NAcademic_Clear);
```

```
//Display Button
btn_NAcademic_Display = new JButton("Display");
btn_NAcademic_Display.setBounds(1140,720,120,55);
btn_NAcademic_Display.setFont(fnt2);
NCjpane.add(btn_NAcademic_Display);

///Adding pannel in frame
jf.add(ACjpane);
jf.add(NCjpane);
jf.add(WPjpane);

//Action Listener
btn_Academic.addActionListener(new ActionListener()
   {
      public void actionPerformed(ActionEvent e)
         {
            ACjpane.setVisible(true);
            NCjpane.setVisible(false);
            WPjpane.setVisible(false);
         }
   }
);

btn_NAcademic.addActionListener(new ActionListener()
   {
      public void actionPerformed(ActionEvent e)
         {
            ACjpane.setVisible(false);
            NCjpane.setVisible(true);
            WPjpane.setVisible(false);
         }
   }
);

//Action listener for welcome page
btn_WP_AC.addActionListener(new ActionListener()
   {
      public void actionPerformed(ActionEvent e)
         {
            ACjpane.setVisible(true);
            NCjpane.setVisible(false);
            WPjpane.setVisible(false);
         }
```

```
        }
    );

    btn_WP_NC.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
                {
                    ACjpane.setVisible(false);
                    NCjpane.setVisible(true);
                    WPjpane.setVisible(false);
                }
        }
    );

    //Creating Array List of Course Class
    academicCourseList = new ArrayList<Course>();
    nonAcademicCourseList = new ArrayList<Course>();

    //Action listener for Add button of Academic Course
    btn_Academic_add.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
                {
                    //try catch for Integer datatype
                    //for duration and number of assesments
                    try
                    {
                        String duration_Academic_temp = txt_Academic_Duration.getText();
                        int duration_Academic = Integer.parseInt(duration_Academic_temp);
                        String numberOfAssesments_Academic_temp =
txt_Academic_NOA.getText();
                        int numberOfAssesments_Academic =
Integer.parseInt(numberOfAssesments_Academic_temp);
                    }
                    catch(Exception I)
                    {
                        String duration_Academic_temp = txt_Academic_Duration.getText();
                        String numberOfAssesments_Academic_temp =
txt_Academic_NOA.getText();
                        if (duration_Academic_temp.isEmpty() ||
numberOfAssesments_Academic_temp.isEmpty())
                        {
```

```
                    JOptionPane.showMessageDialog(jf,"The text field is empty, please
fill it up.");
                }
                else
                {
                    JOptionPane.showMessageDialog(jf,"You have entered invalid data
type.");
                }
            }

            String courseID_Academic = txt_Academic_ID.getText();
            String courseName_Academic = txt_Academic_Name.getText();
            String level_Academic = txt_Academic_Level.getText();
            String credit_Academic = txt_Academic_Credit.getText();
            String duration_Academic_temp = txt_Academic_Duration.getText();
            String numberOfAssesments_Academic_temp =
txt_Academic_NOA.getText();
            int duration_Academic = Integer.parseInt(duration_Academic_temp);
            int numberOfAssesments_Academic =
Integer.parseInt(numberOfAssesments_Academic_temp);
            if (courseID_Academic.isEmpty() || courseName_Academic.isEmpty() ||
level_Academic.isEmpty() || credit_Academic.isEmpty())
            {
                JOptionPane.showMessageDialog(jf,"The text field is empty, please fill
it up.");
            }
            else
            {
                for (Course w: academicCourseList)
                {
                    if (courseID_Academic.equals(w.getCourseID()))
                    {
                        JOptionPane.showMessageDialog(jf,"The given CourseID is
already used. Please enter a different one.");
                        return;
                    }
                }
                AcademicCourse a = new AcademicCourse(courseID_Academic,
courseName_Academic, duration_Academic, level_Academic, credit_Academic,
numberOfAssesments_Academic);
                academicCourseList.add(a);
                JOptionPane.showMessageDialog(jf,"All of your records have been
added.");
            }
```

```
                }
            }
        );

        //Action listener for Add button of Non Academic Course
        btn_NAcademic_add.addActionListener(new ActionListener()
            {
                public void actionPerformed(ActionEvent e)
                    {
                        //try catch for integer datatype
                        //for duration
                        try
                        {
                            String duration_NAcademic_temp =
txt_NAcademic_Duration.getText();
                            int duration_NAcademic =
Integer.parseInt(duration_NAcademic_temp);
                        }
                        catch(Exception I)
                        {
                            String duration_NAcademic_temp =
txt_NAcademic_Duration.getText();
                            if (duration_NAcademic_temp.isEmpty())
                            {
                                JOptionPane.showMessageDialog(jf,"The text field is empty, please
fill it up.");
                            }
                            else
                            {
                                JOptionPane.showMessageDialog(jf,"You have entered invalid data
type.");
                            }
                            JOptionPane.showMessageDialog(jf,"The text field is empty, please fill
it up.");
                        }

                        String courseID_NAcademic = txt_NAcademic_ID.getText();
                        String courseName_NAcademic = txt_NAcademic_Name.getText();
                        String prerequisite_NAcademic = txt_NAcademic_Prerequiste.getText();
                        String duration_NAcademic_temp = txt_NAcademic_Duration.getText();
                        int duration_NAcademic = Integer.parseInt(duration_NAcademic_temp);
                        if (courseID_NAcademic.isEmpty() || courseName_NAcademic.isEmpty()
|| prerequisite_NAcademic.isEmpty())
```

```
                    {
                        JOptionPane.showMessageDialog(jf,"The text field is empty, please fill
it up.");
                    }
                    else
                    {
                        for (Course w: nonAcademicCourseList)
                        {
                            if (courseID_NAcademic.equals(w.getCourseID()))
                            {
                                JOptionPane.showMessageDialog(jf,"The given CourseID is
already used. Please enter a different one.");
                                return;
                            }
                        }
                        NonAcademicCourse a = new
NonAcademicCourse(courseID_NAcademic, courseName_NAcademic,
duration_NAcademic, prerequisite_NAcademic);
                        nonAcademicCourseList.add(a);
                        JOptionPane.showMessageDialog(jf,"All of your records have been
added.");

                    }
                }
            }
        );

        //Action listener for Register button of Academic Course
        btn_Academic_Register.addActionListener(new ActionListener()
            {
                public void actionPerformed(ActionEvent e)
                    {
                        String courseLeader_Academic = txt_Academic_Leader.getText();
                        String lecturerName_Academic = txt_Academic_Lecturer.getText();
                        String startingDate_Academic = txt_Academic_SDate.getText();
                        String completionDate_Academic = txt_Academic_CDate.getText();

                        if (courseLeader_Academic.isEmpty() ||
lecturerName_Academic.isEmpty() || startingDate_Academic.isEmpty() ||
completionDate_Academic.isEmpty())
                        {
                            JOptionPane.showMessageDialog(jf,"The text field is empty, please fill
it up.");
                        }
```

```
                else
                {
                    for (int i=0; i<academicCourseList.size(); i++)
                    {
                        if
(academicCourseList.get(i).getCourseID().equals(txt_Academic_ID.getText()))
                        {
                            AcademicCourse AC = (AcademicCourse)
academicCourseList.get(i);
                            if (!AC.getIsRegistered())
                            {
                                AC.register(courseLeader_Academic,
lecturerName_Academic, startingDate_Academic, completionDate_Academic);
                                JOptionPane.showMessageDialog(jf,"Academic Course is
registered.");
                            }
                            else if (AC.getIsRegistered())
                            {
                                JOptionPane.showMessageDialog(jf,"Academic Course is
already registered.");
                            }
                            else
                            {
                                JOptionPane.showMessageDialog(jf,"The Academic course ID
doesn't match.");
                            }
                        }
                    }
                }
            }
        }
    );

    //Action listener for Register button of Non Academic Course
    btn_NAcademic_Register.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
                {
                    String courseLeader_NAcademic = txt_NAcademic_Leader.getText();
                    String instructorName_NAcademic = txt_NAcademic_Instructor.getText();
                    String startingDate_NAcademic = txt_NAcademic_SDate.getText();
                    String completionDate_NAcademic = txt_NAcademic_CDate.getText();
                    String examDate_NAcademic = txt_NAcademic_EDate.getText();
```

```
                    if (courseLeader_NAcademic.isEmpty() ||
instructorName_NAcademic.isEmpty() || startingDate_NAcademic.isEmpty() ||
completionDate_NAcademic.isEmpty() || examDate_NAcademic.isEmpty())
                    {
                        JOptionPane.showMessageDialog(jf,"The text field is empty, please fill
it up.");
                    }
                    else
                    {
                        for (int i=0; i<nonAcademicCourseList.size(); i++)
                        {
                            if
(nonAcademicCourseList.get(i).getCourseID().equals(txt_NAcademic_ID.getText()))
                            {
                                NonAcademicCourse NC = (NonAcademicCourse)
nonAcademicCourseList.get(i);
                                if (!NC.getIsRegistered())
                                {
                                    NC.register(courseLeader_NAcademic,
instructorName_NAcademic, startingDate_NAcademic, completionDate_NAcademic,
examDate_NAcademic);
                                    JOptionPane.showMessageDialog(jf,"Non academic Course is
registered.");
                                }
                                else if (NC.getIsRegistered())
                                {
                                    JOptionPane.showMessageDialog(jf,"Non academic Course is
already registered.");
                                }
                                else
                                {
                                    JOptionPane.showMessageDialog(jf,"The Non academic
course ID doesn't match.");
                                }
                            }
                        }
                    }
                }
            }
        }
    );

    //Action listener for Remove button of Non Academic Course
    btn_NAcademic_Remove.addActionListener(new ActionListener()
```

```
        {
            public void actionPerformed(ActionEvent e)
                {
                    String courseLeader_NAcademic = txt_NAcademic_Leader.getText();
                    String instructorName_NAcademic = txt_NAcademic_Instructor.getText();
                    String startingDate_NAcademic = txt_NAcademic_SDate.getText();
                    String completionDate_NAcademic = txt_NAcademic_CDate.getText();
                    String examDate_NAcademic = txt_NAcademic_EDate.getText();

                    if (courseLeader_NAcademic.isEmpty() ||
instructorName_NAcademic.isEmpty() || startingDate_NAcademic.isEmpty() ||
completionDate_NAcademic.isEmpty() || examDate_NAcademic.isEmpty())
                    {
                        JOptionPane.showMessageDialog(jf,"The text field is empty, please fill
it up.");
                    }
                    else
                    {
                        for (int i=0; i<nonAcademicCourseList.size(); i++)
                        {
                            if
(nonAcademicCourseList.get(i).getCourseID().equals(txt_NAcademic_ID.getText()))
                            {
                                NonAcademicCourse NC = (NonAcademicCourse)
nonAcademicCourseList.get(i);
                                if (!NC.getIsRemoved())
                                {
                                    NC.remove();
                                    JOptionPane.showMessageDialog(jf,"Non academic Course is
removed.");
                                }
                                else if (NC.getIsRemoved())
                                {
                                    JOptionPane.showMessageDialog(jf,"Non academic Course is
already removed.");
                                }
                                else
                                {
                                    JOptionPane.showMessageDialog(jf,"The Non academic
course ID doesn't match.");
                                }
                            }
                        }
```

```
                }
            }
        }
    );

    //Action listener for Clear button of Academic Course
    btn_Academic_Clear.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
                {
                    txt_Academic_ID.setText("");
                    txt_Academic_Name.setText("");
                    txt_Academic_Level.setText("");
                    txt_Academic_Credit.setText("");
                    txt_Academic_Duration.setText("");
                    txt_Academic_NOA.setText("");
                    txt_Academic_Leader.setText("");
                    txt_Academic_Lecturer.setText("");
                    txt_Academic_SDate.setText("");
                    txt_Academic_CDate.setText("");
                    JOptionPane.showMessageDialog(jf,"The entered values of text field are
cleared.");
                }
        }
    );

    //Action listener for Clear button of Non Academic Course
    btn_NAcademic_Clear.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
                {
                    txt_NAcademic_ID.setText("");
                    txt_NAcademic_Name.setText("");
                    txt_NAcademic_Prerequiste.setText("");
                    txt_NAcademic_Duration.setText("");
                    txt_NAcademic_Leader.setText("");
                    txt_NAcademic_Instructor.setText("");
                    txt_NAcademic_SDate.setText("");
                    txt_NAcademic_CDate.setText("");
                    txt_NAcademic_EDate.setText("");
                    JOptionPane.showMessageDialog(jf,"The entered values of text field are
cleared.");
                }
        }
```

```
        );

        //Action listener for Display button of Academic Course
        btn_Academic_Display.addActionListener(new ActionListener()
            {
                public void actionPerformed(ActionEvent e)
                    {
                        Academic_display_jf= new JFrame("Academic Course");
                        Academic_display_jf.setBounds(10,10,1400,250);

                        DefaultTableModel Academic_table_model= new DefaultTableModel();
                        //Creating table
                        Academic_table = new JTable(Academic_table_model);
                        //Columns in table
                        Academic_table_model.addColumn("Course ID");
                        Academic_table_model.addColumn("Course Name");
                        Academic_table_model.addColumn("Level");
                        Academic_table_model.addColumn("Credit");
                        Academic_table_model.addColumn("Duration");
                        Academic_table_model.addColumn("Number of Asssesments");
                        Academic_table_model.addColumn("Course Leader");
                        Academic_table_model.addColumn("Lecturer Name");
                        Academic_table_model.addColumn("Starting Date");
                        Academic_table_model.addColumn("Completion Date");

                        String rowTitleList[] = {"Course ID","Course
Name","Level","Credit","Duration","Number of Assesments","Course Leader","Lecturer
Name","Starting Date","Completion Date"};

                        Academic_table_model.addRow(rowTitleList);

                        //Rows of the table
                        for(int i = 0; i < academicCourseList.size(); i++)
                            {
                                AcademicCourse AC = (AcademicCourse)
(academicCourseList.get(i));
                                String courseID_Academic = AC.getCourseID();
                                String courseName_Academic = AC.getCourseName();
                                String level_Academic = AC.getLevel();
                                String credit_Academic = AC.getCredit();
                                int duration_Academic_temp = AC.getDuration();
                                String duration_Academic =
Integer.toString(duration_Academic_temp);
```

```
                        int numberOfAssesments_Academic_temp =
AC.getNumberOfAssessments();
                        String numberOfAssesments_Academic =
Integer.toString(numberOfAssesments_Academic_temp);
                        String courseLeader_Academic= AC.getCourseLeader();
                        String lecturerName_Academic= AC.getLecturerName();
                        String startingDate_Academic= AC.getStartingDate();
                        String completionDate_Academic= AC.getCompletionDate();

                        String tableRow[] =
{courseID_Academic,courseName_Academic,level_Academic,credit_Academic,duration
_Academic,numberOfAssesments_Academic,courseLeader_Academic,lecturerName_A
cademic,startingDate_Academic,completionDate_Academic};
                        Academic_table_model.addRow(tableRow);
                    }

                Academic_display_jf.add(Academic_table);
                Academic_display_jf.setVisible(true);
            }
        }
    );

    //Action listener for Display button of Non Academic Course
    btn_NAcademic_Display.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
                {
                    NAcademic_display_jf= new JFrame("Non Academic Course");
                    NAcademic_display_jf.setBounds(10,10,1300,250);

                    DefaultTableModel NAcademic_table_model= new DefaultTableModel();
                    //Creating table
                    NAcademic_table = new JTable(NAcademic_table_model);
                    //Columns in table
                    NAcademic_table_model.addColumn("Course ID");
                    NAcademic_table_model.addColumn("Course Name");
                    NAcademic_table_model.addColumn("Prerequisite");
                    NAcademic_table_model.addColumn("Duration");
                    NAcademic_table_model.addColumn("Course Leader");
                    NAcademic_table_model.addColumn("Instructor Name");
                    NAcademic_table_model.addColumn("Starting Date");
                    NAcademic_table_model.addColumn("Completion Date");
                    NAcademic_table_model.addColumn("Exam Date");
```

```
                String rowTitleList[] = {"Course ID","Course
Name","Prerequisite","Duration","Course Leader","Instructor Name","Starting
Date","Completion Date","Exam Date"};

                NAcademic_table_model.addRow(rowTitleList);

                //Rows of the table
                for(int i = 0; i < nonAcademicCourseList.size(); i++)
                  {
                    NonAcademicCourse NAC = (NonAcademicCourse)
(nonAcademicCourseList.get(i));
                    String courseID_NAcademic = NAC.getCourseID();
                    String courseName_NAcademic = NAC.getCourseName();
                    String prerequisite_NAcademic = NAC.getPrerequisite();
                    int duration_NAcademic_temp = NAC.getDuration();
                    String duration_NAcademic =
Integer.toString(duration_NAcademic_temp);
                    String courseLeader_NAcademic= NAC.getCourseLeader();
                    String instructorName_NAcademic= NAC.getInstructorName();
                    String startingDate_NAcademic= NAC.getStartingDate();
                    String completionDate_NAcademic= NAC.getCompletionDate();
                    String examDate_NAcademic= NAC.getExamDate();

                    String tableRow[] =
{courseID_NAcademic,courseName_NAcademic,prerequisite_NAcademic,duration_NAc
ademic,courseLeader_NAcademic,instructorName_NAcademic,startingDate_NAcademi
c,completionDate_NAcademic,examDate_NAcademic};
                    NAcademic_table_model.addRow(tableRow);
                  }

                NAcademic_display_jf.add(NAcademic_table);
                NAcademic_display_jf.setVisible(true);
              }
          }
      );

      //Set Visible
      jf.setVisible(true);
      WPjpane.setVisible(true);
      ACjpane.setVisible(false);
      NCjpane.setVisible(false);
  }
```

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

- **Course Class:**

```
//Course is a parent class.
public class Course{
    //Creating instance variables.
private String courseID;     private
String courseName;     private
String courseLeader;     private
int duration;


    //Creating a Constructor which accepts three instance variables and the courseLeader
is initialized with empty string ("")
    public Course(String courseID, String courseName, int duration){
this.courseID = courseID;         this.courseName = courseName;
this.duration = duration;
        this.courseLeader = "";
    }
    //Getter and Setter methods to return and initialize of a variable.
    //Getter method for all instance variable starts from here.
public String getCourseID(){
        return this.courseID;
    }

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

    public String getCourseLeader(){
return this.courseLeader;
    }

    public int getDuration(){
        return this.duration;
    }
    //Getter method ends here.

    //Setter method starts from here.
```

```
   //Setter method for courseLeader which puts a new value of courseLeader.
public void setCourseLeader(String courseLeader){
       this.courseLeader = courseLeader;
   }
   //Setter method ends here.

   //The display method is established where all the instance variables gives certain
string output.     public void display(){
       String toDisplay = "Course ID = " + getCourseID() + "\nCourse Name = " +
getCourseName() + "\nDuration = " + getDuration();
if(getCourseLeader().equals(""))            System.out.print(toDisplay);        else
           System.out.print(toDisplay + "\nCourse Leader = " + getCourseLeader());
   }
}
```

- **AcademicCourse Class:**

```
//*The AcademiCourse class is a child class of the Course class.
public class AcademicCourse extends Course{     //Creating
instance variables.     private String lecturerName;     private
String level;     private String credit;     private String
startingDate;     private String completionDate;     private int
numberOfAssessments;
   private boolean isRegistered;

   /*Creating a Constructor which calls the super class then accepts six instance
variables. The parent class and parameter variable are assigned to the instance variable
and the remaining variables are set to "" or False. */
   public AcademicCourse(String courseID, String courseName, int duration, String level,
String credit, int numberOfAssesments){

       //A call to the parent class is formed with arguments.
super(courseID, courseName, duration);

       //Assigning instance variables
       this.level = level;
       this.credit = credit;
       this.numberOfAssessments = numberOfAssesments;

       //The default values are declared ("") or False.
this.startingDate = "";        this.completionDate =
"";        this.lecturerName = "";
       this.isRegistered = false;
```

```
    }

    //Getter and Setter methods to return and initialize of a variable
//Getter method for all instance variable starts from here     public
String getLecturerName(){
        return this.lecturerName;
    }

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

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

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

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

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

    public boolean getIsRegistered() {
        return this.isRegistered;
    }
    //Getter methods ends here.

    //Setter method starts from here.
    //Setter method for lecturerName which puts a new value of lecturerName.
public void setLecturerName(String lecturerName) {        this.lecturerName
= lecturerName;
    }

    //Setter method for numberOfAssesments ehich puts a new value of
numberOfAssesments.
    public void setNumberOfAssessments(int numberOfAssessments) {
this.numberOfAssessments = numberOfAssessments;
    }
```

```
   //Setter method ends here.

   //If it is not registered, this method creates a new course, and if it is registered, it
displays correct information.     public void register(String courseLeader, String
lecturerName, String startingDate, String completionDate) {        if
(getIsRegistered()){
           System.out.println("The course is already registered.");
}else
       //courseLeader in parent class is set.
super.setCourseLeader(courseLeader);
this.lecturerName = lecturerName;            this.startingDate
= startingDate;            this.completionDate =
completionDate;
           this.isRegistered = true;
   }

   //The display method is established where all the instance variables gives certain
string output.     public void display(){
       String toDisplay = "\nLecturer Name = " + getLecturerName() + "\nLevel = " +
getLevel() + "\nCredit = " + getCredit() + "\nStarting Date = " + getStartingDate() +
"\nCompletion Date = " + getCompletionDate() + "\nTotal Assessments = " +
getNumberOfAssessments();
       if (getIsRegistered()){
super.display();
           System.out.print(toDisplay);
       }else
           super.display();
   }
}
```

- **Non-AcademicCourse Class:**

```
//*The NonAcademiCourse class is a child class of the Course class. public

class NonAcademicCourse extends Course {

   //Creating instance variables

private String instructorName;

private String startingDate;

private String completionDate;
```

```
private String examDate;     private

String prerequisite;     private

boolean isRegistered;     private

boolean isRemoved;



   /*Creating a Constructor which calls the super class then accepts four instance
variables. The parent class and parameter variable are assigned to the instance variable
and the remaining variables are set to "" or False. */

   public NonAcademicCourse(String courseID, String courseName, int duration, String
prerequisite) {



       //A call to the parent class is formed with arguments.

       super(courseID, courseName, duration);



       //Assigning instance variables.

this.prerequisite = prerequisite;



       //The default values are declared ("") or False.

this.startingDate = "";        this.completionDate =

"";        this.examDate = "";        this.isRegistered

= false;        this.isRemoved = false;

   }

   //Getter and Setter methods to return and initialize of a variable.

   //Getter method for all instance variable starts from here.

   public String getInstructorName() {
       return this.instructorName;

   }
```

```
    public String getStartingDate() {
return this.startingDate;
    }

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

    public String getExamDate() {
return this.examDate;
    }

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

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

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

```
    //Getter methods ends here.


    //Setter method starts from here.

    //Setter method for instructorName which puts a new value of instructorName.

public void setInstructorName(String instructorName) {

        if (getIsRegistered())

            System.out.println("The instructor name is already registered and cannot be
changed.");

        else

            this.instructorName = instructorName;

    }

    //Setter method ends here.


    //This methods takes four arguments that helps to register the particular course.

    public void register(String courseLeader, String instructorName, String startingDate,
String completionDate, String examDate){

        if (getIsRegistered()){

            System.out.println("The course is already registered.");

        }

else {

            this.setInstructorName(instructorName);

this.isRegistered = true;            this.startingDate

= startingDate;

super.setCourseLeader(courseLeader);

this.completionDate = completionDate;

this.examDate = examDate;
```

```
        }
    }


    //If it is not removed, this method removes a new course, and if it is removed, it
displays correct information.     public void remove(){         if (getIsRemoved()){
            System.out.println("The course is already removed.");
        }
        else {
            super.setCourseLeader("");
this.instructorName = "";
this.startingDate = "";
this.completionDate = "";
this.examDate = "";             this.isRegistered
= false;             this.isRemoved = true;
        }
    }


    //The display method is established where all the instance variables gives certain
string output.
    public void display(){
        String toDisplay = "\nInstructor Name = " +getInstructorName() + "\nStarting Date =
" + getStartingDate() + "\nCompletion Date" + getCompletionDate() + "\nExamination
Date = " + getExamDate();
        if (getIsRegistered()){
super.display();
            System.out.print(toDisplay);
```

```
        }else
super.display();
    }
}
```

References

TechTarget Contributor, 2005. *WhatIs.* [Online]
Available at: https://whatis.techtarget.com/definition/pseudocode
[Accessed 15 May 2021].

MicroTool, 2020. *microtool.* [Online]
Available at: https://www.microtool.de/en/knowledge-base/what-is-a-class-diagram/
[Accessed 14 May 2021].

Guru99, 2021. *Guru99.* [Online]
Available at: https://www.guru99.com/java-platform.html
[Accessed 14 May 2021].

TechTerms, 2012. *Logic Error.* [Online]
Available at: https://techterms.com/definition/logic_error [Accessed
17 May 2021].

TechTerms, 2012. *Syntax Error.* [Online]
Available at: https://techterms.com/definition/syntax_error
[Accessed 17 May 2021].

thinkpython21, 2020. *Semantic Error in a program.* [Online]
Available at: https://www.greenteapress.com/thinkpython/html/thinkpython021.html
[Accessed 17 May 2021].